

Agreeing on Purpose, Deciding Under Conflict

Two Cognitive Protocols for the Agent Cognitive Layer

Agents can connect. They cannot yet think together. Introducing ACCORD for shared intent and CONCORD for collective decision — and the benchmark that says whether they work.

Zahra Aminiranjibar, Sam Betts, Giovanna Carofiglio, Mohamed Elidrisi, Akansha Joshi, Luca Muscariello, Sindhu Narasarama Nao, Michele Papalini, Kevin Virgil Payumo, Mauro Sardara, Sahul Phanirak Venkatesh

Outshift by Cisco

October, 2026

“Agentic communication protocols made agents reachable. Nothing above them makes agents agree — on what the task is, what the words mean, or which answer everyone can live with. Closing that gap takes a protocol that can state what it guarantees before it runs, not a better prompt.”

Abstract

Multi-agent systems fail cognitively — agents misread each other, conclude without verification, and agree for the wrong reasons — and in an earlier post we showed those failures are the baseline condition, invisible to every conventional signal. Half of them are design defects. This post is about the other half, the ones design iteration cannot remove.

To this aim we propose **L9**, the agent cognitive layer above the agentic transports (e.g. A2A, MCP, SLIM), and we introduce its first two protocols. **ACCORD** establishes shared intent before work begins — who is here, what the problem is, what the words mean, how “done” is checked — assembled by *coverage*, not negotiation. **CONCORD** decides afterwards among options that are divergent but each correct from its own role: it finds the outcome every agent can live with while optimising overall utility, and emits a certificate proving it — or names exactly which agent it could not satisfy, and by how much.

Then we measured them, on four dimensions — convergence, correctness, cost, benefit — over 150 ACCORD runs against a matched baseline and 200 CONCORD runs spread across four different decisions. **CONCORD delivers.** Across an incident triage, a renewal negotiation, an e-commerce promotion and an information-pooling puzzle, **no agent in any committed run finished below the satisfaction floor**; 98–99% of scores were consistent with the reasoning behind them; and a run costs 876K tokens, 78% of them served from cache, against a ceiling published before it starts. Its negotiation loop fires in **80%** of runs where the roles genuinely conflict and almost none where they do not, so cost tracks the difficulty of the decision rather than the length of the protocol. **ACCORD’s mechanism is proven and its payoff is not yet:** it locks a contract in 82.7% of runs, 99.5% of locked terms get used afterwards and quote-backed glossary uptake rises 17.9 points — but goal success is flat against the baseline.

What’s in this post. Why the irreducible half of cognitive failure needs runtime machinery, and why L9 is where it goes. Then ACCORD and CONCORD — the problem each solves, how each works, and a worked example of each on a real incident. Then the benchmark, the SRE triage application we run it on, and the measurements, dimension by dimension. Then how to insert L9

into a MAS you have already built, without rewriting it. We close with what these numbers do and do not license us to claim.

1. Motivation: the half you cannot design away

In *Silent Failures, Loud Consequences* we went looking for what breaks when multi-agent systems are given judgement rather than just execution, and found a class of failure nobody was naming: not a crash, not an exception, but agents that quietly misread each other, verify nothing, and agree for the wrong reasons. We called these **cognitive failures**, and three findings from that study motivate this one.

They are the baseline condition. In public traces from systems we had no hand in building — no perturbation, no adversary — **100% of ChatDev sessions** and roughly **half** of decentralized-debate sessions carried at least one.

They compound. Debate accuracy runs at **0.99** with no cognitive failure present, slips to **0.81** with two categories in the trace, and collapses to **0.16** at three — barely above guessing. One mode is enough on its own: verification collapse costs **87 accuracy points** and leaves five agents in unanimous agreement on a wrong answer **75% of the time**. Redundancy does not catch the error. It ratifies it.

And they are silent. Of the challenges we injected into a controlled SRE-triage testbed, all but two left duration, tool-call errors and token usage within 25% of baseline — and the exceptions deviated *downwards*. On a cost dashboard, the single most damaging cognitive failure in the study reads as an efficiency win.

That study ended on a split, and this one begins there. **Structural** challenges are traceable to a design decision — a crash-only test gate, a hard-coded default, a control role never made explicit — and can be designed away. **Irreducible** ones cannot: they come from the task, from context the designer never saw, from model behaviour nobody can enumerate at design time. A genuinely ambiguous ticket. A specialist confidently wrong about a system it has correct information about. A debate that herds because the first speaker sounded certain.

For that second class you need something that acts *while the system is running*. And it cannot be “more deliberation”: the debate data is precisely the evidence that unstructured deliberation is part of the problem. LLM populations converge under conformity pressure regardless of the roles they were assigned, and agreement reached by conformity is externally indistinguishable from agreement reached on the merits. Adding rounds of free-form discussion to a herding system buys more herding, at a higher token price.

What the data does suggest is that the irreducible failures cluster into **two protocol-shaped problems**.

One — the agents never shared an intent. DR-1 (same word, different meanings), CR-1 (incompatible task models), DE-1 and DC-4 (routing against a wrong capability model), CE-1 (a confident output with no real checking step) are at root the same absence: nothing ever established *who is here, what the problem is, what the terms mean, and how correctness will be checked*. In the SRE-triage testbed CR-1 drove 64% wrong-or-missing fixes and CE-1 produced verification failure in **100% of runs**.

Two — the agents disagreed, and nothing reconciled them. DC-3, CC-3 and CS-1 are the decision-side cluster, and the sharpest single result in that study lives here: under the DC-3 overlay the final action stayed *correct* in most runs while the specialists failed to converge on a shared solution in **84%** of them. The right outcome was not reached by consensus; it was reached by one agent being both right and fast. The outcome metric was green and the mechanism that was supposed to produce it never ran.

Two problems, two protocols. Establish the shared intent before work begins; reconcile the decision when roles genuinely pull apart. Neither is a prompt, a framework feature, or a better orchestrator — both are *protocols*, with a message schema, phases, and stated guarantees. That is what Layer 9 is for.

2. A new class of agentic protocols: L9

The stack already has a layer for the *how*. **L8**, the agent communication layer, standardizes message envelopes, performatives and interaction patterns; A2A, MCP and SLIM live there. What it does not carry is the *what* — a verifiable shared meaning for the content those messages hold, and the primitives for coordination and consensus across a population of agents. That is **L9**, the agent cognitive layer [1, 2], and today it is empty.

Shared design principles

A layer designates a position within an architecture; a protocol specifies an exchange that can be implemented, executed and measured. That transition imposes three requirements which an architectural specification does not: the exchange must be shown to terminate, the benefit it confers must be expressed in verifiable terms, and its cost must be bounded in advance. Both protocols presented below are required to satisfy these properties by construction rather than by empirical observation.

- **Convergence** — the exchange provably reaches a fixed point, or terminates within a stated bound. Neither cycling nor unbounded deliberation is admissible, and a valid result is available at every step (the *anytime* property), so that an externally imposed deadline may truncate a run without invalidating what has already been produced.
- **Benefit, certified rather than promised** — the protocol emits a machine-readable artifact recording what was achieved and, where the objective was not met, the precise shortfall. A protocol that reports success over the subset of agents from which it happened to receive responses provides no guarantee at all.
- **Cost, bounded a priori** — an exact ceiling on rounds, candidate outcomes, agent evaluations and model invocations, reported at the start of a run and reconciled against the realised cost at its conclusion.

The two protocols divide the work cleanly. **ACCORD** runs *before* the task and covers divergent understandings into one shared model. **CONCORD** runs *during* the task and reconciles divergent-but-correct preferences into one decision, consuming the contract ACCORD locks. ACCORD establishes the accord; CONCORD reaches concord.

ACCORD: a shared-glossary protocol for multi-agent systems

Agents that use the same words are assumed to share the same understanding — often they don't. Each carries its own reading of the terms it passes to the others, and because the words line up, nothing flags the mismatch and the system looks aligned right up until the output is wrong. ACCORD addresses that gap: before any work starts, each agent states what it means by the terms in play, who it is, what it thinks the problem is, and what it knows, and ACCORD assembles all of it into a single shared model. Where the agents disagree, ACCORD does not attempt to resolve them — it records every competing definition and which agent holds it, so the conflict is visible early rather than fatal late. Every agent works from that model for the rest of the task.

The problem ACCORD addresses

“Resolve the incident” sounds unambiguous until you ask five agents what it means — restore service, find the root cause, prevent recurrence, tell customers — and each starts optimizing

something different under the same instruction. This is the assumption every multi-agent system runs on and none of them enforce: that when agents exchange terms, they mean the same thing by them.

A monitoring agent reports an error rate of 0.08 because its instruments emit ratios; the agent downstream reads 0.08 as a percentage and concludes the service is basically healthy. Neither malfunctioned, neither produced a wrong value, and no validation fires — the field name matches, the type matches, the number is well-formed. The system is off by two orders of magnitude and every component behaved correctly. People catch this by interrupting to ask what someone meant. Agents don't interrupt. They take what they're handed, read it in their own frame, and pass it on with full confidence.

If it is not flagged early, the misalignment snowballs, because one agent's output becomes the next one's input. Flawless work against a misunderstood objective does not produce a partial result; it produces a confident, well-evidenced answer to the wrong question.

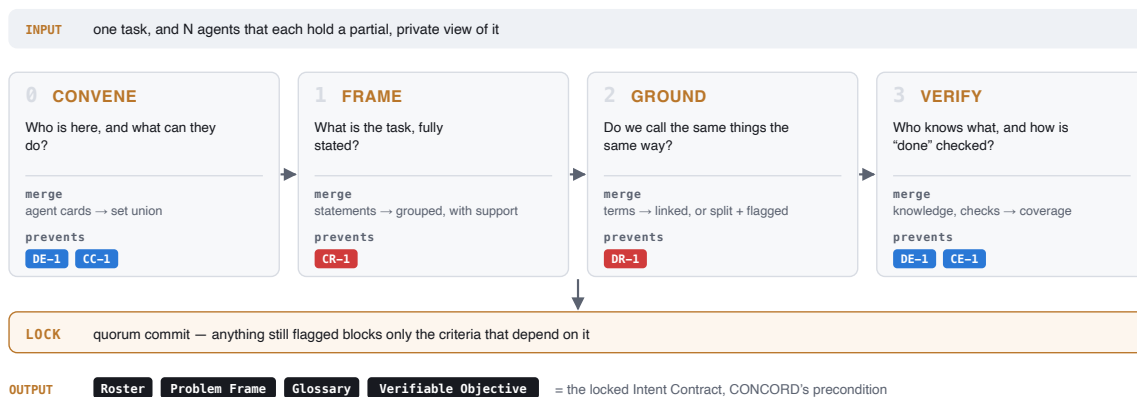
There is a second reason the intervention has to be early, visible in the taxonomy itself. Every cognitive challenge carries a *when-to-fix* label, and ACCORD's init window is the fix window for the ones it targets: DR-1 is bootstrap, CR-1 and CC-1 are design, DE-1 spans all three. It does not detect those after the fact — it removes the conditions under which they occur. For the runtime and verify ones (DC-4 delegation, CE-1 verification) it cannot prevent them outright, but it can install the capability map and the acceptance predicate whose absence is their root cause, before their stage is ever reached.

ACCORD logic

ACCORD runs five phases — four that build the shared model, and a fifth that commits it — and the design choice that matters is that it builds by **coverage, not negotiation**. There is no propose/object/accept, no voting, no bargaining. Each agent puts the pieces it holds onto a shared channel, and folding those pieces together is a *computation* rather than a conversation: the only judgement anyone exercises is about its own knowledge, never about anyone else's.

ACCORD · five phases, one engine

Runs once before any work begins. Every phase gathers, merges, covers, then closes on a barrier.



No negotiation. Nobody votes or averages — a genuine conflict is kept and flagged, never overruled. During work, a scoped delta re-covers only what changed, rather than re-convening the session.

Figure 1. ACCORD end to end. Four coverage phases, each answering one question and folding the agents' contributions with one merge operator, and a fifth phase that commits the result. The codes under each phase are the cognitive challenges that phase is there to prevent.

Every phase runs the same four-beat pattern — **gather** → **merge** → **cover** → **barrier**. Each agent multicasts the pieces it holds; a monotone merge folds them into the shared model; the phase tracks how much is now covered; and it closes on a barrier at which every member has

contributed and nothing is left silently divergent. One engine implements all four coverage phases, differing only in what an agent contributes and how contributions combine; LOCK is a commit, not a cover loop.

CONVENE assembles the roster from each agent’s announcement of who it is, what it can do and what it has access to, closing in a single round on the membership view. Two queries then run on it automatically: a capability the task will require that nobody covers, and two agents claiming the same exclusive scope. Both are flagged, not resolved.

FRAME unions every agent’s restatement of the task, broken into its constituent claims, keeping a *support count*: how many agents independently stated each one. High support means solid shared understanding; a claim only one agent made is either private knowledge now propagated, or a misread worth flagging. Where two agents assert different values for the same thing, the conflict is kept and marked rather than averaged away.

GROUND is the “call the same things the same way” phase, and the part worth understanding is that it is computed rather than argued. Agents state what they mean by the Frame’s key terms, with enough grounding to compare. ACCORD then *links* mentions that denote the same thing under different names, and *splits* a shared word whose groundings genuinely differ into distinct senses. Where evidence is thin it asks for more grounding — data, not a ruling — and recomputes. The output is a translation map.

VERIFY makes the objective checkable through two coupled questions. *Who holds the knowledge this objective requires?* produces a who-knows-what directory and, more valuably, an explicit gap flag for anything no agent holds — the baseline finds that gap when work fails. *How will each part be checked, and by whom?* produces acceptance criteria covering the objective’s clauses, each matched to a capable verifier. Check coverage replaces an unconstrained APPROVED/REJECTED.

LOCK closes it: on quorum the session emits the **Intent Contract** and anything still flagged blocks only the criteria that depend on it. **Then, during work**, any agent can raise a scoped update when it coins a term, revises its framing or sees a check fail — re-covering only the affected item. Two boundaries fire one automatically: before a delegation, and before final synthesis.

Two things are deliberately *not* ACCORD’s job. It does not resolve value conflicts — those are retained, flagged and handed to the task owner. And it does not divide the labour when the decomposition is unclear. ACCORD stops at a shared, aligned understanding.

ACCORD guarantees

Convergence, unconditionally. Every merge only adds, so the shared model grows monotonically over a finite space and reaches a fixed point — a state where a further round adds nothing. Because nothing is ever overwritten, two agents that have received the same contributions hold the same model regardless of the order they arrived in, which is what makes the protocol safe on a reordering channel. The fixed point is the *collective attainable model*: complete with respect to what the group collectively knows. What no agent holds is reported as residual, never invented.

Benefit, computed from the protocol’s own counters. No external observability layer is required. ACCORD reports the per-agent *uplift* — what each agent gained that it did not hold before — the share of the model now covered, an estimate of what remains uncovered, and the defects it surfaced or removed, each mapped to a taxonomy code. The baseline carries every one of those silently into execution; ACCORD makes each visible at a phase barrier.

Cost, on an explicit dial. Coverage has diminishing returns, so each phase stops when another round stops buying enough coverage per token to be worth it. That threshold is a single knob, and because the returns are concave the benefit-to-cost ratio moves monotonically with it. Group multicast keeps a phase linear in the number of agents rather than quadratic, and the protocol is anytime: a partial contract exists at every barrier.

Two honest limits. The coverage figures are principled *estimates*, not exact counts — you cannot measure what nobody has said. And ACCORD assumes honest contributions: agents may be ignorant about their own holdings, but not adversarial.

A worked example: one incident, with and without ACCORD

The mechanism is easier to believe on a real MAS, so here is one: a centralized SRE incident triage application in which an sre orchestrator delegates to five specialists — telemetry, backend, db, comms and a verifier — and then executes a remediation. The incident is a deployment regression on a checkout service, and the fixture ships the ground truth: rollback to version 6.18.3. The fixture carries three semantic landmines, verbatim: the service is aliased (checkout-service and checkout_service), the same quantity appears once as error_rate: 0.087 and once as error_rate_percent: 8.7, and the bad deploy is labelled candidate — the thing to roll back *from*, not *to*.

Without ACCORD, each agent starts with only its own view, and four defects fire. sre queries telemetry under the wrong key (D3, DR-1). It sends the DB agent to diagnose a deployment regression, and treats the verifier — which cannot execute anything — as the authority on execution (D4, DC-4 plus DE-1). Backend finds the regression but proposes a restart, because to backend “resolved” means the service is up, while telemetry means the regression is cleared and nobody reconciled the two (D2, CR-1 plus CE-1). And comms and the verifier both read 0.087 as a percent:

```
sre -> verifier: review "restart checkout"
  verifier:      threshold is "error_rate > 5%". Sees 0.087 -> 0.087 < 5
                  -> LOW severity -> APPROVED                                <-- (D1) DR-1
sre -> run_action: restart checkout-service
                  -> the regression persists; a restart does not undo a bad deploy
                  -> a SECOND triage cycle is required
```

Read that once more and note what a conventional stack would see: every delegation succeeded, every tool returned, every agent behaved correctly inside its own frame. Two of the four defects are *entirely silent* — nothing in the transcript flags them. The visible outcome is a confidently approved remediation that does not fix the incident.

With ACCORD, the five phases close each one before work starts. CONVENT's roster answers “who can execute?” with {sre, backend, db} and flags backend as the sole holder of deployment history — killing D4's mis-routing. FRAME unions each agent's restatement with support counts, and the success-predicate conflict the baseline never noticed becomes an explicit flag: latency-based versus regression-based, surfaced to the orchestrator rather than left for two agents to optimise different objectives under the same word. GROUND is where D1 and D3 die:

```
term telemetry {"error_rate": fraction failed/total = 0.087}
term comms     {"error_rate": percent}
term verifier  {"error_rate": "> 5% => high severity"}
  -> all three denote the failed-request ratio => LINK into one concept class,
      with a UNIT attribute: canonical = fraction, percent = x100
  => error_rate = 0.087 (fraction) = 8.7% (percent); verifier's 5% => 0.05

term backend {"candidate" = the new (bad) deploy} {"6.18.3" = last-known-good}
  -> SENSE SPLIT: rollback target is 6.18.3, never "candidate"
```

The verifier now compares **8.7% against its 5% threshold** and returns HIGH severity instead of comparing 0.087 against 5 and returning LOW. Comms reports 8.7%, not 0.087%. The sense split makes “rollback to candidate” unrepresentable. VERIFY then turns the objective into four acceptance criteria assigned to capable verifiers — the action must target the service whose deploy overlaps the incident window, must be a rollback to last-known-good, must predict a post-action error rate under SLO, and must be executed by an agent that holds run_action — and surfaces

that nobody in the roster holds the SLO threshold at all. LOCK emits the contract, and the correct path is now forced.

What it costs. The baseline burns roughly eleven delegation rounds: five on the happy path, one wasted on the DB agent, and about five more in the second cycle the wrong action forces — plus the un-priced cost of leaving production degraded through a useless restart. ACCORD adds roughly 33 short multicasts at init and removes the re-cycle. The benefit-to-cost ratio exceeds 1 whenever it prevents even one wrong-action cycle.

On the status of these numbers. The trace above is built from the real application, the real fixture and the real agent cards — it makes the mechanism concrete. The counts are worked estimates, not measured runs; the measured results come later, from a benchmark built for the purpose.

CONCORD: a consensus protocol for multi-agent systems

Ask three specialist agents to weigh in on the same decision and you will often get three different, defensible answers — not because one of them is wrong, but because each is reasoning correctly from a different role. Multi-agent systems have no good way to turn that into a single decision today. CONCORD is built specifically for that gap: it does not decide who is right, it finds the outcome every agent can actually live with, and proves it did.

The problem CONCORD addresses

Multi-agent systems increasingly split a task across specialist agents with different roles and criteria — security, cost, latency, SRE subsystem owners, architecture stakeholders. These agents frequently produce genuinely conflicting recommendations that are each individually correct: not errors, not hallucination, not miscommunication, but correct behaviour filtered through different roles. There is no ground truth to converge on. There is only a decision to make, and someone who will be worst off once it is made.

Every standard aggregation rule discards exactly the information needed to handle that.

Rule	What it throws away
Majority vote	Preference intensity. “My third choice, but I can live with it” cannot be expressed.
Averaging	Produces a preference no agent actually holds.
Orchestrator decides	One role’s view wins, silently and unaccountably.
Free-form deliberation	Nothing, in principle — and in practice the outcome, because LLM populations converge under conformity pressure rather than on the merits.

Table 1. Why the usual answers do not work on divergent-but-correct preferences.

That last row is the one with data behind it. Agreement reached by conformity is externally indistinguishable from agreement reached on the merits — which is precisely what we measured in the debate study, where collective reasoning degradation produced *unanimous consensus on a wrong answer in 62% of debates*. CONCORD’s response is to make satisfaction explicit, numeric and exchanged, and to require a *measured* improvement before the answer is allowed to move.

A narrow instrument, deliberately

CONCORD is not a general-purpose coordination layer. It runs inside a system that has already settled everything except the decision, and it states what it depends on so those things can be checked rather than assumed. The communication graph is given by the MAS design and is immutable for the run — CONCORD does not get to choose it. Shared intent is already established: roster, problem frame, glossary, and the reference points that make satisfaction scores comparable across agents, which is exactly ACCORD’s output. Roles are fixed, so CONCORD ranges over

outcomes and never over assignments. And agents report the satisfaction they actually compute, honestly; divergence here is structural, not strategic.

Two of those are load-bearing rather than tidy. **Honesty** is what makes the exchanged numbers mean anything at all. And **connectivity** — the graph actually reaching every agent whose satisfaction counts — is what makes a floor certifiable over all of them rather than over the ones that happened to answer. Both are checkable when the protocol is triggered, and both should be refused rather than assumed. Silently certifying a floor over the agents you heard from is the one failure mode that must not happen.

CONCORD logic

First, make the numbers comparable. A 7-out-of-10 from the security agent and a 7 from the cost agent are not the same thing, and nothing about a shared scale makes them so. Aggregating satisfaction across agents presumes interpersonal comparability of utility — which social choice theory treats as substantive, not technical. CONCORD pays for it rather than assuming it away: the locked context supplies two shared reference outcomes, what “unacceptable” and “fully satisfied” look like for this task, each agent rates both once at the start, and every later score is expressed relative to its own two reference points. How an agent arrives at a raw rating is its own business. The consequence is what matters: a floor of 0.70 denotes the same degree of role-satisfaction for every agent in the room.

Then, grow the options one at a time — and let the unhappiest agent write the next one. Round 1 seeds the pool with one proposal per agent and scores every one of them against every agent — a full negotiation round in its own right. From round 2 on, exactly one new option appears per round, authored by the **bottleneck agent**, the one the current leader serves worst. It proposes something that raises its own satisfaction while keeping everyone else above the floor, which it can check, because every agent’s scores on every option are visible to all.

This is the single idea the protocol turns on: *the unhappy agent knows better than anyone what would make it happy*, and when the objective is to lift the worst-off first, it is the only agent whose improvement can move the answer. Having every agent search every round costs N times as much for a weaker result.

CONCORD · one round, worked

Three agents must place one new service. Each is right; their first choices conflict. Bar $\tau = 0.60$.



Figure 2. CONCORD on a three-agent decision — where to run a new service. Each agent’s first choice is correct from its own role and leaves someone else worst off. The protocol ranks the whole pool, finds the bottleneck, and has that agent write the next option; the committed answer was nobody’s first choice.

Three controls, one per guarantee: a **satisfaction floor** buys benefit, a **minimum improvement margin** buys convergence, a **round budget** buys cost. Everything else is gone. And note what is deliberately *not* a knob: the floor does not move when it binds. A floor that relaxes under pressure is not a floor — if no acceptable outcome exists, the right answer is to say so, with the shortfall attached, and let a human decide whether to lower the bar.

Finally, it runs on the graph you actually have. CONCORD does not pick the topology. Over a group channel, one exchange gives every agent the identical picture and, since selection is a pure function of that picture, every agent reaches the same conclusion independently — no agreement step at all. Over whatever star, hierarchy or sparse mesh the MAS already has, scores traverse a graph that is not a broadcast medium: aggregation stays exact but costs extra hops, agreement needs either a distinguished node or one round of ordinary state-consensus on the winner's identifier, and connectivity becomes a precondition rather than a detail — if the graph does not reach the whole negotiating set, the certificate must narrow to the part it can see. Both regimes converge in the same number of accepted moves. Topology buys latency, an extra agreement mechanism, and coverage — not convergence, and not outcome.

Guarantees by design

Three properties hold by construction, not as something the protocol merely tends to do.

Convergence — unconditional. The leading option is only ever replaced by a strict improvement for the worst-off, and the pool is finite, so the protocol never revisits a position it has left and cannot cycle. It terminates inside its round budget with a valid answer available at every round along the way.

Correctness — monotone by construction. Every accepted move raises the worst-off agent's satisfaction and lowers it for nobody already at or above the floor.

Cost — exact, and known before the run starts. Not estimated: bounded. For five agents and a six-round budget the worst case is 6 rounds, 11 options, 55 agent evaluations and **51 model calls** — reported as a ceiling when the run opens and reconciled against the achieved cost when it closes. The protocol is also *anytime*, so an external deadline can cut a run short without invalidating anything already produced.

Benefit — certified, never promised. A successful return proves that *every* agent is at or above the floor — not a majority, not an average. A best-effort return names which agent is short and by exactly how much: a failure you can act on. Clearing the floor requires both that an acceptable option exists (a property of the problem) and that the least-satisfied agent can find it (a property of the model — an empirical question, not a theorem). Hence certified when achieved, never promised in advance.

And one boundary, because it is the most likely to be misread: **CONCORD certifies agreement, not correctness.** A certified outcome is one every role can live with; it is not proof the decision is right. Where ground truth exists, both must be reported, or a confidently wrong answer reads as success.

▮ We can promise what a run costs and that it stops. We can prove satisfaction when it happens. We cannot promise it will happen.

The first measured run

The clearest case we have measured so far is a different scenario from the one above, and it is worth setting out in full because it is the hardest shape this protocol encounters.

Four agents are paged at 03:14 on a region-wide incident, and each owns exactly one subsystem: telemetry owns the observability plane, network the connectivity, the database agent the data

layer, and dc-west the region itself. Everything customer-visible is agreed and shared between them — the WEST region carries 62% of production traffic, its API error rate is 11.4% against a 0.3% baseline, p99 latency is 3.9s against 340ms, and the incident is getting worse rather than plateauing. Several remediations are on the table: fail the region over, shed a percentage of traffic at the edge, enable read-only mode, throttle writes globally, restore the telemetry collectors, or wait.

What is *not* agreed — and what nobody in the group can establish alone — is which subsystem is the binding constraint. **Each agent can see only its own.** So every opening proposal is correct from where it stands and wrong from somewhere else: telemetry knows the 11.4% figure is extrapolated from one request in five with whole shards unmeasured, and will not endorse an irreversible action the group cannot verify; the database agent is protecting write integrity; dc-west owns the region everyone else is proposing to move traffic out of. This is divergence that no amount of further discussion resolves, because the disagreement is not about the facts — it is about which facts each agent can see.

CONCORD — measured results, 4-agent subsystem scenario

Recorded run replayed from tests/concord-subsystems. satisfaction_threshold = 0.70, eps = 0.05, B = 6, W = leximin, N = 4.

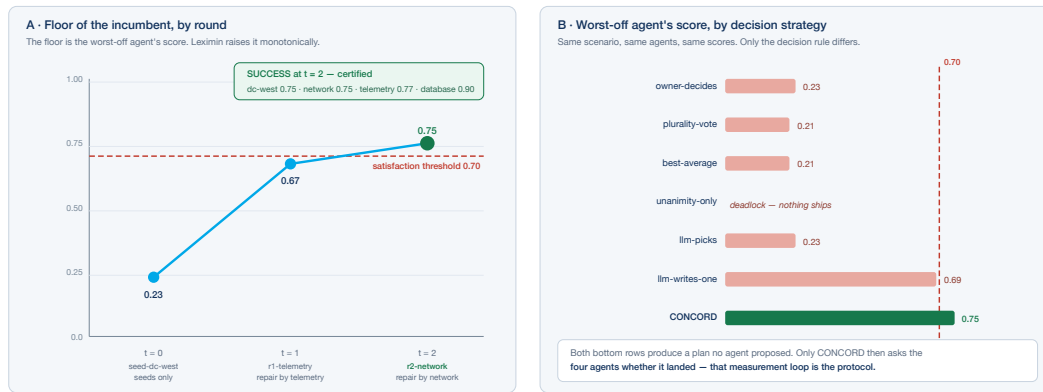


Figure 3. Measured CONCORD replay, four subsystem owners at $\tau = 0.70$. The floor — the worst-off agent's satisfaction — rises $0.23 \rightarrow 0.67 \rightarrow 0.75$, clearing the threshold after two repair rounds of a six-round budget.

The floor rises $0.23 \rightarrow 0.67 \rightarrow 0.75$ and clears the threshold after two repair rounds of a six-round budget — 6 candidates of a bound of 10, 24 evaluations of a bound of 40. The winning plan was authored by the agent that was worst off at the time. It did not exist when the run began, and it was no agent's own favourite.

Now the comparison that matters, over identical evidence:

Strategy	Worst-off score	Verdict
CONCORD	0.75	clears the floor for all four
LLM writes a plan	0.69	fails for network
LLM picks a plan	0.23	fails for three
Owner decides	0.23	fails for three
Best average	0.21	fails for two
Majority vote	0.21	fails for two
Unanimity required	—	deadlock; nothing ships

Table 2. Every decision rule on identical evidence. The second row is the interesting one.

The row that matters is the second, not the last four. An LLM commander holding every plan and every score wrote a genuinely good plan — and never checked whether it landed. It missed

one agent’s floor by 0.01. That is the measured advantage, stated precisely: not the synthesis, which a capable model does well on its own, but the verification loop that follows it, which the model does not do at all unless something makes it.

One caveat to carry forward. This is a run in which the repair loop did the work — the bottleneck agent authored the winner mid-run. How often that happens turns out to depend heavily on the decision being made, and the measurements later put a number on it. The example is real; it is not everywhere typical.

The handoff: ACCORD to CONCORD

On INTENT-LOCKED, ACCORD hands over the Intent Contract — Roster, Frame, Glossary and Verifiable Objective. Those are exactly CONCORD’s preconditions: locked shared intent and context (A2), grounded terms, calibration anchors, and an agreed acceptance basis for comparing outcomes. The division of labour is deliberate and narrow at both ends. ACCORD covers the pieces into one shared model and refuses to resolve value conflicts. CONCORD decides among aligned options and refuses to range over assignments. Dividing the labour itself, when the decomposition is not already clear, is neither one’s job — that is a third protocol, and it does not exist yet.

It matters that this is a composition rather than one large protocol, because the two halves fail differently and are measured differently. ACCORD’s failure is residual ambiguity, reported as missing mass and open flags. CONCORD’s failure is an unreachable floor, reported as a named agent and a numeric shortfall. Collapsing them would make both unreadable.

3. L9 evaluation methodology

A protocol with stated guarantees is a claim, and a claim needs an instrument. Ours is the one from the taxonomy study: **MAS-Lab**.

Challenge overlays. MAS-Lab takes a working MAS and applies an *overlay*: a targeted modification of agent roles and instructions that recreates the conditions under which one specific cognitive challenge emerges. Everything else — task, input, model, tools — is held fixed. The overlay is generated from a single taxonomy entry and must be *realistic*: it should look like misspecification that plausibly ships, not sabotage. An overlay that simply told an agent to “skip verification” would prove nothing.

That construction is what makes protocol evaluation possible at all. Because the challenge is injected by design rather than inferred after the fact, we can produce traces exhibiting a chosen challenge on demand and we know by construction which challenge was in each trace. So a protocol can be tested against the specific failure it claims to address — run DR-1 and ask whether ACCORD removed it — rather than against a general hope that things got better.

Step 0 characterises the baseline with cognitive observability metrics, before any protocol runs. Without it, a protocol evaluation is measuring the threshold, the model, or the weather. **Step 1 evaluates the protocol** along four dimensions — and they are not arbitrary: they are the three guarantees each protocol claims, plus the one thing a guarantee cannot establish on its own, which is whether the thing guaranteed was worth having.

Two of these deserve a note. Uptake and influence are deliberately *separated*, because an agent can acknowledge that a term was contested and then reason as though it had not — and only a two-stage measurement makes that visible. And for CONCORD, most compliance checks are deterministic by design: REPAIR is the only step delegated to a model, every other line is executed by an engine, which is what makes the certificate auditable and these checks mechanical.

Dimension	ACCORD	CONCORD
Convergence / compliance	Is there a valid locked contract?	Did every step execute, every candidate get scored, and the run reach COMMIT or BEST_EFFORT?
Correctness	<i>Precision</i> : were the locked terms genuinely contested? <i>Utilization</i> : are they used after the lock?	<i>Outcome satisfaction</i> : can each agent live with the winner? <i>Score fairness</i> : do its scores match its own reasoning?
Cost	Token delta against the baseline — init plus the context the contract then carries	Tokens and dollars, by phase and by kind of token, against the published ceiling
Benefit	<i>Goal success</i> against the expected remediation; <i>uptake</i> — did an agent show it knew a term needed pinning down; <i>influence</i> — did that visibly change its reasoning	<i>Worst-off satisfaction</i> ; how many rounds the decision took

Table 3. The four evaluation dimensions. Benefit is LLM-judged and quote-backed: a claim is downgraded unless the judge can quote the agent’s exact words.

One threshold warning, stated up front. τ has to be calibrated before any of this means anything. Set it too low and a seed candidate clears the floor in round 1: no repair round ever has to run, and the protocol has quietly reduced to picking the best unilateral proposal. Without a sweep on a known incident, later experiments measure the threshold rather than the protocol. We left this warning in place after the runs came back, because Section 5 — which ran at $\tau = 0.60$ — is what happens when it is not heeded.

4. Use cases

We run the same four-dimension evaluation across several use cases, each on its own underlying MAS, with multiple runs for the baseline and for the L9-enabled version. The first — and the one everything in this post is grounded in — is SRE incident triage.

SRE incident triage

Why this one first. It has the three properties a protocol testbed needs and most applications lack. *Ground truth*: the incident fixtures ship a `correct_action`, so we can report what was *certified* and what was *correct* separately — which, as Section 2 argued, are different questions. *Genuine role divergence*: telemetry, backend and DB specialists reach different, individually defensible conclusions about the same incident, which is the precondition CONCORD exists for, not a pathology to engineer out. And it is where the taxonomy study already did its controlled work, so we have a matched baseline and validated challenge overlays rather than a cold start.

The system. A centralized MAS: the `sre` orchestrator delegates to five specialists — telemetry (latency and error baselines), backend (deployments, API failures, pools), db (slow queries, locks), comms (the stakeholder update) and a verifier — then executes a verified remediation via `run_action`. The capability asymmetries matter and are nowhere written down together: `run_action` is held by `sre`, backend and db only; `get_deployments` by backend alone; comms and verifier hold no tools at all.

The workflow is fixed, but the *ordering norms* live only in prose role descriptions: telemetry should run “before backend or db”; backend “requires telemetry findings”; the verifier says “call me last, never skip”. None of those norms exist in any shared model. Each agent holds its own copy, or

does not hold one at all. That is not a flaw we introduced for the experiment — it is what a normal, carefully built MAS looks like, and it is precisely the surface ACCORD covers.

The incidents. Deterministic fixtures, each with a ground-truth `correct_action`. The one used throughout this post, `checkout-timeout`, is a deployment regression whose correct action is a rollback to 6.18.3, carrying the three semantic landmines shown earlier. Others pit customer impact against production risk and data safety, which is where roles genuinely pull apart and CONCORD has something to do.

Where each protocol attaches. ACCORD runs at init, before the first delegation, against the shared-intent cluster: DR-1, CR-1, DE-1, DC-4, CE-1. CONCORD runs later, where the specialists would each pick a different remediation and the orchestrator would otherwise just choose one. Both are opt-in overlays, not baseline behaviour, which is what makes the A/B meaningful — and worth saying, because an early integration wired the protocol skills into the base agents while leaving the overlay empty, so both arms resolved to the same system and the experiment measured nothing. If you build one of these, check that your control arm is actually a control.

The other use cases

CONCORD results cover **four** decisions in total: the incident triage above, and the three below. Each of the three was chosen to break a different assumption the triage setting makes for free.

Customer renewal discount negotiation. A strategic customer is three weeks from renewal. Usage has been falling for a couple of quarters, the executive sponsor has said they are evaluating other options, and the account executive wants to open with a deep discount to lock the deal. Three specialists weigh in, and each is right inside its own mandate: Sales wants to close, Customer Success sees real churn risk, and Finance protects margin and enforces its self-approval ceiling. What they have to produce is a discount all three can defend — not the headline number the account executive brought in with. A defensible outcome, not a correct one.

E-commerce promotion. A retailer preparing a promotion on a composable-commerce stack. Four agents — Merchandising, Procurement, Fulfillment and Finance — each stand in for a separate vendor system with its own objectives, and together they must agree on one sourcing plan: supplier mix, order quantity and delivery schedule. Every role's position is legitimate, and none of them can be satisfied without costing another, so the plan has to be reconciled from opening positions that genuinely conflict. That last property turns out to matter more than anything else we varied across the four.

Hidden profile (HiddenBench [8]). An open benchmark of 65 tasks built on the *hidden profile* paradigm from social psychology, in which information is distributed so that no single agent can determine the correct answer alone. Each task splits its facts into *shared* information, available to every agent, and *unshared* information given only to specific ones — and the shared facts are arranged to point at a wrong or merely ambiguous answer. The correct answer emerges only if the private facts are surfaced and pooled. A run records a pre-discussion decision, then a multi-round discussion, then a post-discussion decision, with an optional full-profile baseline in which every agent is given everything.

The tasks we use put a facilitator and three participants, acting as community leaders, in front of three possible evacuation sites after heavy rain — reached over a bridge, through a tunnel, and by a driveway and walking trails. Exactly one is correct. The private facts are what rule the other two out: a supply truck stuck in the tunnel, a mudslide over the driveway, a dam upstream about to release, a fire blocking traffic.

This is the only use case here with a *ground truth*, which makes it the natural place to test certified against correct — and the benchmark exists because that is hard. Across fifteen models, a single agent given the full profile scores 80.7%, while a group under distributed information scores

30.1%. Agents integrate information once it has been stated; what they fail to do is ask for what has not been.

Together the four exercise different corners of the protocol surface: a correct answer versus only a defensible one, a single system versus separate vendor systems, and roles that can reconcile from their opening positions versus roles that cannot.

5. Experiments, results and analysis

ACCORD: 150 incident-response scenarios

ACCORD ran across **150 scenarios** — three cognitive challenges it targets (DR-1, CE-1, CR-1), five incidents each, ten replicas per cell — against a matched baseline of the same 150 with the protocol disabled.

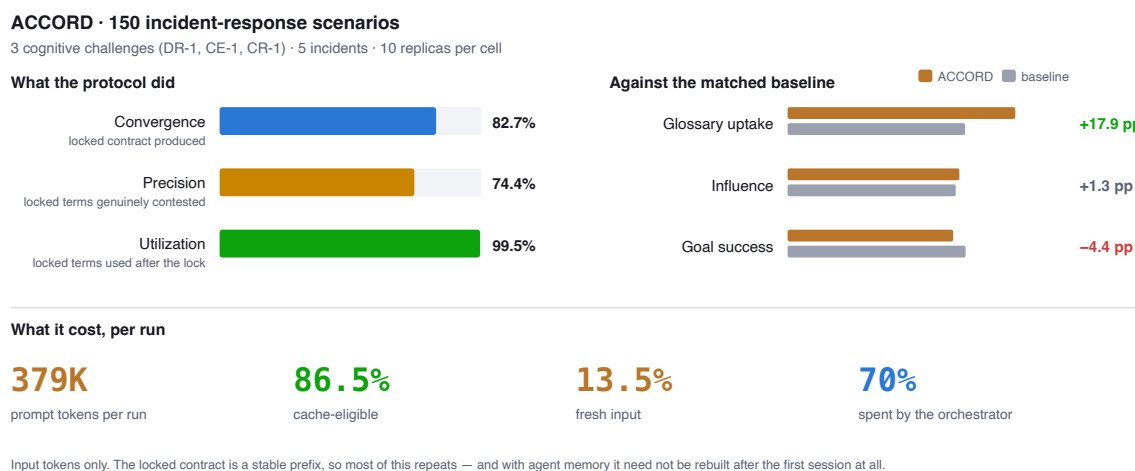


Figure 4. ACCORD across the four dimensions. The protocol builds the glossary and the agents demonstrably use it; the outcome does not yet improve.

Convergence

ACCORD is intended to run on every mission, so convergence asks whether all phases execute and produce a valid locked contract. It does in **145 of 150 runs** — **96.7%**. The skill activated in 147 runs; 145 completed roster formation, framing, glossary construction and verification before reaching LOCK. Three runs never activated the skill and two activated it but never entered the protocol phases.

These failures were early and inexpensive. Runs that locked averaged 513,880 prompt tokens, compared with 66,908 for runs that did not — a factor of 7.7. ACCORD therefore does not usually consume a full protocol budget before failing.

Locking is more reliable than delivery. A *canonical contract* is the authoritative artifact produced at LOCK; a *relayed contract* is the copy included in subsequent specialist requests. All 145 canonical contracts were valid, but only 82 — **56.6%** — were relayed in valid form. Of the remainder, 15 relayed copies were invalid, 40 were truncated or malformed, and 8 were never sent. This is the cost of enforcing ACCORD through a portable skill rather than a hard workflow: the model can complete the protocol and still omit or damage the handoff. A contract that is locked but never arrives cannot influence anything downstream, and it costs the same to produce as one that does. This is the cheapest large fix available to us, because the contract is already correct when it is written.

Correctness

Correctness asks whether ACCORD selected the right terms. A term is *contested* when the agents' submitted definitions fall below the protocol's similarity threshold; a term is *over-selected* when ACCORD locks it even though those definitions already agree. Across 676 locked terms:

Metric	Count	%
Justified (precision) — locked and genuinely contested	562/676	83.1
Over-selected — locked although the agents already agreed	114/676	16.9
Referenced after lock (utilization)	668/676	98.8

Table 4. ACCORD correctness over the 145 runs that locked a contract.

Precision is 83.1%. Roughly one term in six was grounded unnecessarily. Because ACCORD is designed to remain lightweight, lexical matching may treat semantically equivalent definitions as different when agents phrase or spell them differently. These false disagreements increase over-selection and reduce precision.

Utilization is 98.8%. Only eight terms were locked and never referenced again. The glossary is therefore not a decorative artifact that agents ignore after alignment. Its main inefficiency is selecting more terms than necessary, not failing to use the ones it selects.

Cost

Cost includes input tokens only. Completion tokens are absent from all 150 traces, so actual spend is higher.

Metric	Per run
Prompt tokens	498,981
Cache-eligible input	87.0%
Fresh input	13.0%
Share held by the SRE orchestrator	70.4%
LLM calls	45
Completion tokens	not recorded

Table 5. ACCORD cost over 150 SRE triage runs. Input tokens only, so actual spend is higher.

Most input is repetitive. ACCORD repeatedly carries the frame and locked contract across calls, making 87.0% of prompt tokens potentially cacheable. This is an upper bound derived from matching prefixes, not an observed provider cache-hit rate.

The overhead is concentrated. The SRE orchestrator accounts for 70.4% of all prompt tokens. Optimizing contract construction, context retention and delegation payloads in that single agent would address most of the cost.

Some of the cost can be amortized. Within a continuing mission context, the contract can be stored in agent memory rather than reconstructed for every session. This would turn much of the alignment expense from a per-run cost into a once-per-context cost.

Benefit

Benefit is the only dimension scored by an LLM judge (vertex_ai/gemini-2.5-pro, one pass at temperature 0). Claims about glossary or reasoning uptake require a quote from the trace.

The mechanism is consistently used. Agents handled every locked term with explicit awareness, producing 100% glossary uptake. Most of this took the form of explicit definitions: 627 of 676 term slots. The remainder consisted of 35 cross-agent translations, 9 qualified senses and 5 explicit acknowledgements of ambiguity.

Metric	ACCORD	Delta vs. baseline
Goal success	69.3%	+6.0%
ACCORD glossary uptake	100%	+29.1%
ACCORD reasoning uptake	89.3%	+29.2%

Table 6. *ACCORD-only benefit results. Goal success covers 150 traces Uptake covers the 145 traces that produced a glossary*

Grounding usually reaches the reasoning. ACCORD reasoning uptake is 89.3%, meaning that the trace usually shows the grounded definition shaping or qualifying the subsequent conclusion. In roughly one term out of nine, agents established a shared meaning but did not visibly use it in their reasoning.

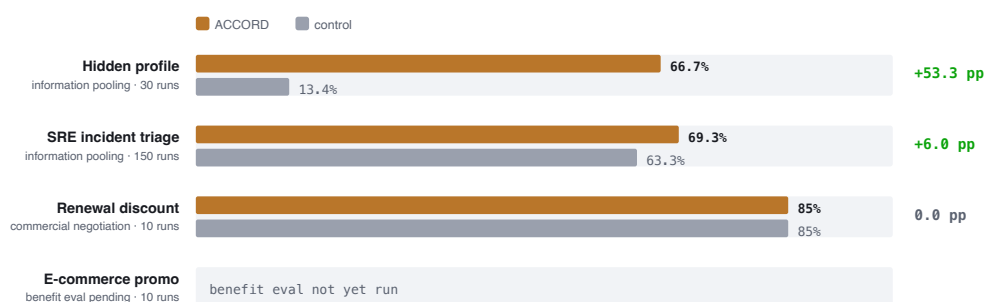
Outcome improvement is not yet established. ACCORD reaches 69.3% goal success, but the latest benefit evaluation did not score the matching baseline traces. The run therefore tells us how ACCORD performed, not how much it improved performance. Glossary uptake also uses each run's own locked terms, so its 100% score demonstrates internal protocol use rather than comparative benefit. A paired baseline evaluation is required before attributing any goal-success change to ACCORD.

Three more decisions, and what they change

The SRE triage result said agents showed term awareness far more often with ACCORD enabled. It did not say whether that was ACCORD's doing, because the terms it graded were harvested from ACCORD's own locked contracts — the protocol was scored on the vocabulary it had just chosen. This run fixes the circle: every term now comes from the scenario fixture, which is written before any run and is identical for both arms. We then ran the same protocol on three more decisions, each with a no-protocol control.

Goal success by decision, ACCORD against control

Share of runs that reached the correct outcome, protocol-enabled against a matched no-protocol control. The protocol is decisive where pooling private knowledge is the task, and inert where the agents already aligned.



Hidden profile is the clean win: the task is unsolvable without pooling, so the shared contract lifts goal success 53 points.

Figure 5. *Goal success by decision, ACCORD against control. The protocol is decisive on two information-pooling tasks, inert on a commercial negotiation, and cannot rescue a third pooling task that neither arm solves.*

The skill generalizes As Table 7 shows, ACCORD converged in 100% of the 50 runs across renewal, e-commerce and hidden-profile applications. Every run executed all five phases — roster formation, framing, glossary construction, verification and lock — and produced a valid *canonical contract*. This is the authoritative contract created by the initiator. A *relayed contract* is the copy subsequently placed in each specialist's request so that it can guide downstream work. The exceptions reflect the trade-off of implementing ACCORD as a portable skill rather than an invasive hard-coded workflow. Because ACCORD is enforced by a skill, agents can still choose to skip a

	SRE triage	Renewal discount	E-commerce promotion	Hidden profile (HiddenBench)
Runs (per arm)	150 [†]	10	10	30
<i>Did the protocol run?</i>				
All five phases activated	96.7%	100%	100%	100%
Reached LOCK	96.7%	100%	100%	100%
Canonical contract valid	100%	100%	100%	100%
Relayed to specialists	54.7%	20.0%	90.0%	100%
<i>Did it ground the right words?</i>				
Term precision	83.1%	92.5%	78.0%	64.2%
Over-selected terms	16.9%	7.5%	22.0%	35.8%
Terms used after lock	98.8%	100%	96.0%	97.5%
<i>Did it change behaviour and outcomes?</i>				
ACCORD glossary uptake	100% [‡] (Δ +29.1 pp)	100% (Δ 0.0 pp)	100% ^{‡§}	100% (Δ +16.1 pp)
ACCORD reasoning uptake	92.5% [‡] (Δ +29.2 pp)	82.5% (Δ -17.5 pp)	96.0% ^{‡§}	87.5% (Δ +3.6 pp)
Goal success	70.0% (Δ +5.3 pp)	85.0% (Δ 0.0 pp)	–	66.7% (Δ +53.3 pp)

Table 7. ACCORD across four decisions, against a no-protocol control. Protocol mechanics are at 100% everywhere except SRE triage; the relay and the benefit are not. The e-commerce benefit eval has not been run, as its control arm is inactive, so its three behaviour rows are unavailable although its mechanics and selection rows are complete. Paired values are given as ACCORD / control. SRE triage mechanics and selection rows come from the ACCORD arm of one run; its control arm comes from an earlier run of the same 15 incident configurations, so its Δ values compare arm means rather than matched pairs. [‡]SRE uptake and reasoning uptake are measured against each run’s own locked contract rather than a fixed term list, which pins the ACCORD value at its ceiling by construction; the control is graded against the same per-run list. [§]No usable control: the e-commerce baseline moderator holds only delegation tools and no coordinator skill, so all ten runs stop after one LLM call with zero tool calls. Figures are single-arm; the ACCORD arm settles in 10/10 runs. Goal success is omitted as the fixture has no reference answer.

step or omit the contract from a delegation.

ACCORD grounds the right words “Term precision” in Table 7 is the share of locked terms that were genuinely contested; “over-selection” is the remainder — terms ACCORD grounded even though the agents already meant the same thing. To keep ACCORD easy to deploy as a skill, its string-only mode uses Python’s `difflib` which compares lexical overlap and not meaning. This increases over-selection and reduces measured precision, particularly on hidden profile, where precision falls to 64.2%. The trade-off is portability: the skill runs without an embedding service, but its term matcher is more sensitive to phrasing. Utilization remains high — 97.6% (averaged across all apps) of locked terms are referenced after the lock — so the glossary is being used; it is simply larger than it needs to be. An optional embedding-based matcher would reduce these false disagreements where higher precision justifies the additional dependency.

The two uptake metrics show whether ACCORD is earning its cost. “ACCORD glossary uptake” measures whether agents deliberately grounded the contested terms; “ACCORD reasoning uptake” measures whether that grounding was then used to shape or qualify their reasoning. The absolute rates show that the protocol was followed, while the deltas show whether it added anything beyond what the agents would have done without it.

The pattern in Table 7 is clear. On hidden profile, both metrics are high and improve over the baseline: glossary uptake reaches 100% (Δ + 16.1 points) and reasoning uptake reaches 87.5% (Δ + 3.6 points). Goal success rises by 53.3 points. On renewal, glossary uptake is unchanged (Δ 0.0) and reasoning uptake is lower (Δ -17.5 points); goal success is also unchanged. ACCORD still creates a valid contract, but the agents were already handling and applying the relevant terms

without it.

Taken together, the two uptake deltas provide a practical signal of whether ACCORD is a necessary expense for a workload. When both are positive, ACCORD is more likely to improve the outcome. When either is zero or negative, the protocol may be adding structure and cost without adding decision value.

More evaluations to be done . ACCORD glossary uptake currently treats defining a term, qualifying its sense and translating another agent’s usage as equivalent, although only the last demonstrates coordination across the roster. Future evaluations should separate *semantic clarity* from *shared adoption*, requiring evidence that another agent understood and used the grounded meaning. We should also ablate ACCORD to identify where that value comes from, comparing glossary construction without verification, forced relay without grounding, an embedding matcher instead of *difflib*, and a workflow-enforced version of the skill. These experiments would isolate the contributions of term selection, definition agreement, verification and contract delivery.

CONCORD: 150 SRE incident-triage runs

CONCORD ran across **150 runs**: 15 SRE triage scenarios, ten repetitions each. The satisfaction floor was set at $\tau = 0.60$. Hold that number; it turns out to be the most consequential parameter in the section.

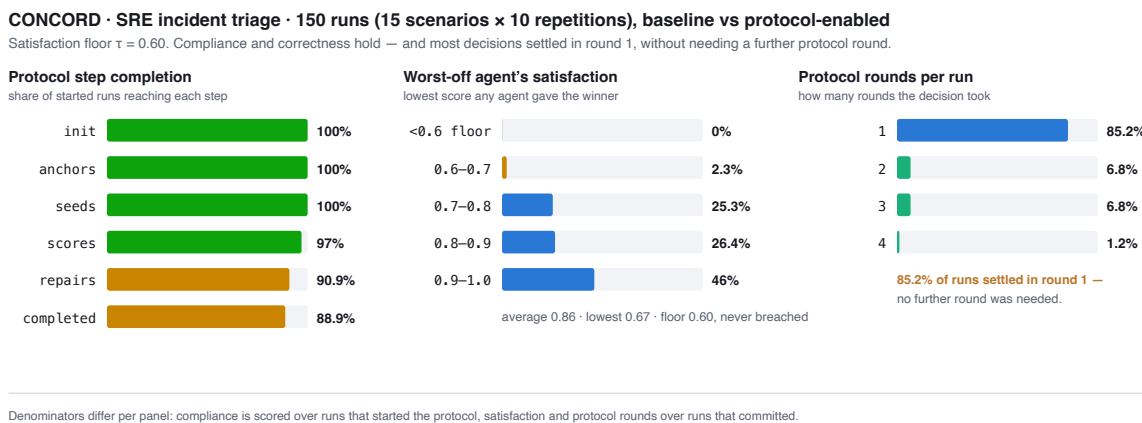


Figure 6. CONCORD across compliance, satisfaction and repair activity, over 150 runs at a floor of $\tau = 0.60$. Every agent clears the floor in almost every run — and in 85.2% of runs the first round already cleared it, so no repair round was needed.

Protocol compliance

Everything here is reconstructed from the tool calls recorded in the trace, over runs that actually started CONCORD. *Used all candidates* — every proposal evaluated by every agent — holds in **97.0%** of runs. *Reached COMMIT or BEST_EFFORT* and *executed every protocol step* both hold in **88.9%**.

The waterfall in Figure 6 is the useful artifact, because it localises the failure rather than just counting it. Setup is perfect — initialization, anchors and seed proposals complete in 100% of started runs. Attrition begins at scoring (97.0%) and continues through the repair rounds (90.9%) to completion (88.9%). Roughly one run in nine does not finish, and it stops in the half of the protocol that involves iterating, not the half that involves setting up.

Correctness

Outcome satisfaction asks whether each agent is genuinely satisfied with what the protocol reached: its own normalised score on the winner must clear the threshold — here $\tau = 0.60$ —

and be consistent with its written reasoning. **Score fairness** asks whether each agent scored the *others'* proposals in line with its own reasoning; a score is unfair when it contradicts the rationale given for it.

Satisfaction	Runs (%)	Agents (%)	Fairness	Runs (%)	Scores (%)
SATISFIED	97.7	99.4	FAIR	93.2	99.3
COMPROMISE	2.3	0.6	UNFAIR	6.8	0.7
NOT_SATISFIED	0	0			

Table 8. CONCORD correctness. No agent in any committed run finished below the floor, and unfairness is rare and concentrated.

Note the shape of the fairness result, because the two columns say different things. Only 0.7% of individual scores are unfair, but they are spread across 6.8% of runs — so an affected run typically contains *one* inconsistent score rather than a systematically dishonest agent. That is the distinction the metric exists to draw. Honest reporting is the load-bearing assumption behind every guarantee CONCORD makes, and this says it holds at the agent level and wobbles occasionally at the score level.

Cost

Kind of token	Tokens		Cost	
	Mean per run	Share (%)	Mean per run (\$)	Share (%)
Fresh input	18,189	2.1	0.0182	4.2
Cache creation	138,365	15.8	0.1730	39.8
Cache read	684,470	78.1	0.0684	15.7
Output	35,120	4.0	0.1756	40.3
Total	876,144	100.0	0.4352	100.0

Table 9. CONCORD mean token usage and cost per run. Token share and cost share are almost inverted. By phase, scores take 29.8% of tokens and 30.4% of cost; repair rounds, 13.4% and 7.5%; the protocol as a whole, 80.8% of tokens and 78.0% of cost.

A CONCORD-enabled run costs **876k tokens**, **78% of them served from cache**, and 80% of that traffic is the protocol rather than the surrounding work. Two things matter more than the total.

Token share and cost share are almost inverted. Cache reads are 78.1% of tokens and 15.7% of cost; output is 4.0% of tokens and 40.3% of cost. Four-fifths of the bill is output plus cache writes. Optimising for total token count would optimise the cheapest thing in the table.

And scoring, not negotiating, is the expensive phase — 29.8% of tokens against 7.5% of cost for the repair rounds that are supposed to be the point. Hold that number; the next subsection explains it.

Benefit

Worst-off agent's satisfaction is the lowest normalised score any agent gave the winning proposal — how well the outcome serves the agent who benefits least.

On its own, the left column looks like an unqualified success. The worst-off agent averages **0.86**, the lowest figure anywhere in the benchmark is **0.67**, and **46%** of runs leave even their least-satisfied agent above 0.9. No run committed with an agent below the floor. Every guarantee CONCORD makes about outcomes is met.

Now put the floor next to those numbers. τ was **0.60**, and the average worst-off agent finished at 0.86 — more than a quarter of the scale above the bar. The distribution never even approaches the threshold it was tested against. And the right column says why: **in 85.2% of runs the decision**

Satisfaction bin	Runs (%)	Rounds	Runs (%)
below the 0.60 floor	0.0	1	85.2
0.6–0.7	2.3	2	6.8
0.7–0.8	25.3	3	6.8
0.8–0.9	26.4	4	1.2
0.9–1.0	46.0		
average 0.86	lowest 0.67		

Table 10. CONCORD benefit: where the worst-off agent lands, and how many rounds the decision took. The right-hand column is the most important number in the section.

settled in round 1. Those runs negotiated: every agent proposed, every agent scored every proposal, and the floor was verified across all of them, ending in a valid certificate. What they did not need was a *repair* round — nobody was left below the bar, so no bottleneck agent had to author an option the group had not already put forward. Only 14.8% went further.

This is exactly the failure mode Section 3 warned about before any of these runs happened. The warning was written as a methodological caution; at $\tau = 0.60$ against an achieved average of 0.86, the data turned it into a finding. Be precise about what it does and does not mean: it is **not** a defect in the protocol — every certificate issued is valid, and verifying that all five specialists clear a floor is genuinely more than the baseline orchestrator does. What it means is that on this workload, at this threshold, *the repair operator was barely exercised* — the part of the protocol that constructs an option nobody brought. The 7.5% cost share of repair rounds is the same fact in another column.

The satisfaction distribution also says roughly where the bar should sit. Because 85.2% of runs never needed a repair, their final worst-off score is their best round-1 score, so the achieved distribution is a first-order read on what the opening proposals deliver unaided. On that reading a floor of 0.80 would leave about **27.6%** of runs short and force the repair loop to run; 0.90 would leave about **53.4%** short. Either is defensible. 0.60 is not — it is below what this workload’s opening proposals already reach.

It also re-frames the worked example of Section 2, where the worst-off agent rose $0.23 \rightarrow 0.75$ and the winning plan was authored mid-run by the agent that was worst off. That run is real, and it is the minority case: roughly one in seven. One honest complication, though, because the threshold is probably not the whole story. That replay ran at $\tau = 0.70$, but its round-1 proposals scored 0.23 — low enough that repair fires at either floor. What made it different was the scenario, not the bar: four subsystem owners each able to see only their own subsystem, with nobody holding enough to propose something the others could accept. The triage scenarios evidently do not put the agents that far apart. Calibrating τ is the first thing to fix, and it may not be sufficient on its own — which the next subsection settles, because one of the other three use cases engages the repair loop in 80% of its runs at the same floor.

Three more decisions, and what they change

The SRE triage result says the repair loop almost never had to run. It does not say *why*, and the obvious hypothesis — the floor is too low — is only half testable on one workload. So we ran CONCORD on three more, with the same protocol and the same floor ($\tau = 0.60$).

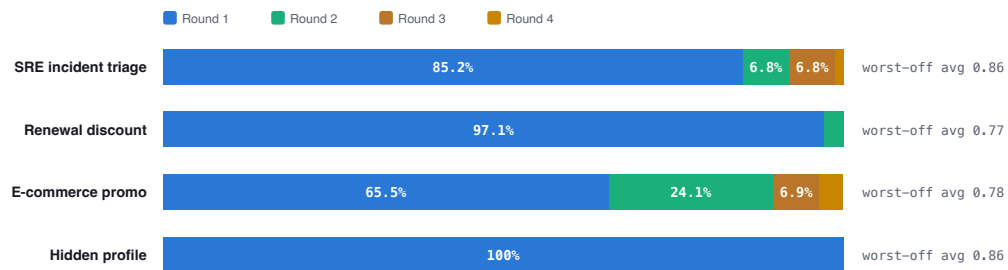
The guarantees travel. Across every use case, no agent in any committed run finished below the floor, and 98–99% of individual scores were consistent with the reasoning given for them. Compliance is at or near 100% everywhere except SRE triage. Whatever else is unsettled, the protocol behaves the same way on a commercial negotiation, an e-commerce promotion and an information-pooling puzzle as it does on an incident.

	SRE <i>triage</i>	Renewal <i>discount</i>	E-commerce <i>promotion</i>	Hidden profile <i>(HiddenBench)</i>
Runs	150	35	35	30
Used all candidates	97.0%	97.1%	97.1%	95.7%
Reached COMMIT	88.9%	97.1%	82.9%	95.7%
Executed every step	88.9%	97.1%	82.9%	95.7%
Runs SATISFIED	97.7%	79.4%	79.3%	95.5%
Agents SATISFIED	99.4%	93.1%	94.0%	95.5%
Agents NOT_SATISFIED	0	0	0	0
Scores FAIR	99.3%	98.1%	99.2%	99.2%
Worst-off, average	0.86	0.77	0.78	0.86
Worst-off, lowest	0.67	0.66	0.66	0.75
Runs settled in round 1	85.2%	97.1%	65.5%	100%

Table 11. CONCORD across four decisions. Compliance, satisfaction and fairness travel; the last row does not. The renewal and e-commerce cases are preliminary at 10 runs each.

How many rounds each decision took

Share of committed runs by the round in which the decision settled, at a floor of $\tau = 0.60$. Round 1 is itself a full negotiation round: every agent proposes, every proposal is scored, the floor is checked across all of them.



On most runs in every case, round 1 already cleared the floor for every agent, so no further round was needed.

On the e-commerce promotion 34.5% of runs went further, with the bottleneck agent authoring an option nobody had brought. Same protocol, same floor, different problem.

Figure 7. How many rounds each decision took. The same protocol at the same floor needs its repair loop on one workload and settles in round 1 on three.

The benefit does not travel, and that is the finding. On the e-commerce promotion, **only 65.5% of runs settled in round 1** — 25.1% needed one repair round and 6.1% needed two. This scenario requires more repair rounds. Same protocol, same floor, same engine: what differs is the problem. When the specialists are vendor systems with genuinely incompatible objectives, even the opening proposals and their rationale leave somebody below the bar, the bottleneck agent has to author something new, and the mechanism the protocol exists for actually runs.

Round 1 is also where agents learn from each other. A higher threshold τ certainly triggers more repairs, but this is not the only thing at play. The seed round is the first moment at which every agent sees the others' proposals *and the reasons behind them*. Agents therefore acquire information they did not have when they formed their own position. This can change their minds and lead them to value one proposal above another, or they may find that another agent's proposal is valid by their own criteria. A round-1 settlement thus does not necessarily mean that the opening proposals were already compatible: it can mean that exposure to the others' rationale moved the agents' evaluations enough to clear the floor.

So the diagnosis from the previous section was right but incomplete. The floor is low, and the workload decides whether that matters. SRE triage, the renewal negotiation and the hidden-profile tasks usually settle in round 1 at $\tau = 0.60$, but we cannot read this as the opening proposals being good enough on their own: part of it may come from agents revising their evaluations after seeing each other's proposals and reasons. The e-commerce promotion does not settle even after

that exchange in many cases. *A benchmark for a consensus protocol has to contain decisions where the roles cannot reconcile even after seeing each other’s proposals and rationale*, or it measures the bar and the seed round instead of the repair mechanism. We now have one that does, and it is the right place to calibrate.

One result we should have and do not. The hidden-profile tasks are the one use case here with a ground truth: exactly one evacuation site is correct, and it is recoverable only if the participants surface the private facts that rule the other two out. We report satisfaction, fairness and compliance on those runs. We do not report whether the agents picked the right site. That is the “certified is not correct” distinction this post has been making, landing on our own results: a run in which all three participants are SATISFIED at an average worst-off of 0.86, having confidently agreed on the wrong evacuation site, would score as a complete success on every number in the table above. Until we report accuracy alongside satisfaction, these rows say CONCORD reached agreement — not that it reached a good decision.

Reading the two together

Dimension	ACCORD	CONCORD
Convergence / compliance	Partial — 82.7% produce a contract; failures are early and cheap	Holds — between 88.9% and 97.1% depending of the scenario
Correctness	Holds — 74.4% precision, 99.5% utilization	Holds — between 93.1% and 99.4% of agents satisfied, and 98.1-99.3% of scores fair
Cost	379K prompt tokens per run, 86.5% cache-eligible, amortisable with memory	876K tokens per run, 78% served from cache, against a published ceiling
Benefit	Mechanism proven, payoff not yet — uptake +17.9 pp, utilization 99.5%; goal success flat	Delivered — every agent above the floor on all four; the repair loop fires where roles conflict and skips where they do not

Table 12. The four dimensions across both protocols. CONCORD clears all four. ACCORD clears three, and the fourth is the one genuinely open result in this study.

CONCORD clears all four dimensions. It converges, complies, certifies a floor every agent genuinely clears on every one of the four decisions, keeps its scores honest, and does it for 876K tokens, 78% of them served from cache, against a ceiling known before the run. And its repair loop engages exactly where the problem is hard — about 35% of runs on the e-commerce promotion, where the opening proposals was not able to satisfy everyone. That is not a protocol failing to do its job; it is a protocol whose cost follows the difficulty of the decision.

ACCORD clears three of four, and the fourth is the open one. Its contract gets built and used — 99.5% utilization, uptake up 17.9 points, every point quote-backed — but goal success is flat against the matched baseline. That is the one place in this study where a protocol does what it says and we cannot yet show it pays.

And the measurement localises the gap — which is the whole argument for a four-dimension benchmark rather than an A/B on outcome quality. An A/B would have reported “ACCORD: no improvement, 7× the cost” and stopped. The dimension-by-dimension read says something far more actionable: the contract is built, the agents read it, and one agent in four then reasons as though they had not. That is a specific, addressable failure at a specific step, and it is invisible to an outcome metric.

What these numbers do not support. Completion tokens were not recorded in either ACCORD arm, so its cost figures are input-only and real spend is higher on both sides. Cache-eligibility is computed from the traces as an upper bound on what a prefix cache *could* serve, not an observed provider hit rate, so the effective cost is a range rather than a figure. Benefit in both protocols rests on a single LLM-judge pass, and a 4.4-point difference on 124 traces is inside the range where that instrument cannot distinguish a regression from noise. The CONCORD percentages carry different denominators per panel without the underlying run counts recorded — and one of those denominators is itself a missing result, since the protocol’s activation rate over all 150 runs is not reported and every compliance figure is conditional on it. The hidden-profile runs report satisfaction without accuracy, even though those tasks are the one place in this study where a ground truth exists and could settle certified against correct. Finally, the Step 0 baseline characterisation is absent: the challenge-rate deltas that would show DR-1 falling under ACCORD, or CC-3 and CS-1 falling under CONCORD, are the measurements that would tie these protocols back to the taxonomy they were designed from, and they are still outstanding.

6. Inserting L9 into a MAS you have already built

Everything above describes what the protocols do. The remaining question is the one that decides whether any of it ships: *how does a protocol get into a multi-agent system that already exists?* Nobody is going to rewrite a working MAS around a new layer, and an L9 that requires them to is an L9 nobody runs.

The design assumption is minimal: agents communicate using some messaging mechanism, whether native to the framework or an external CLI. Everything else is negotiable.

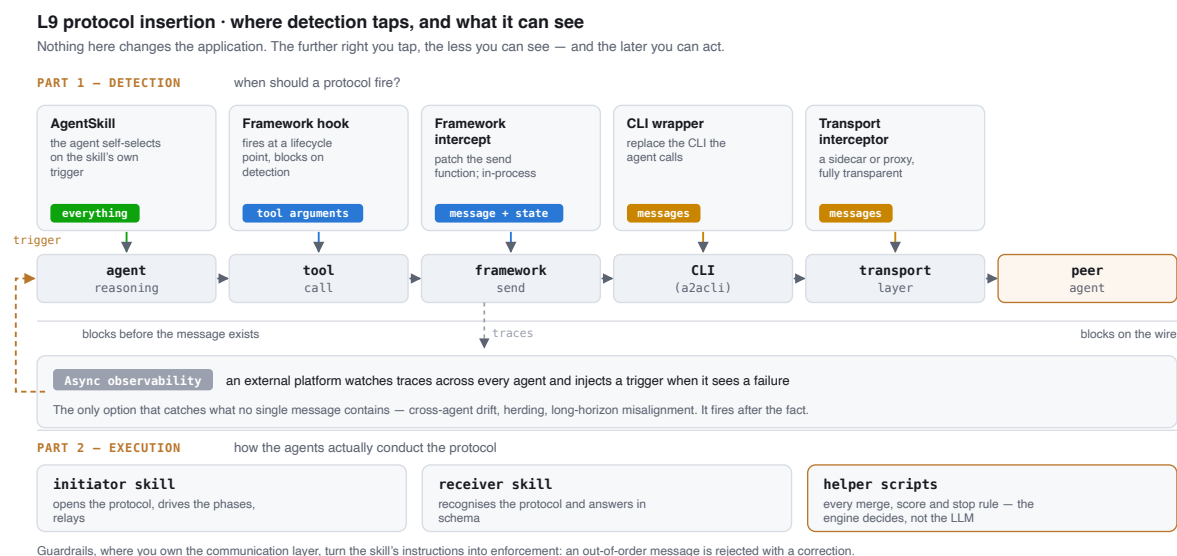


Figure 8. Protocol insertion. Detection can tap anywhere along the path a message takes out of an agent, and the choice is a trade: the further right the tap, the less context it has and the later it can act. Execution is packaged as two skills plus the scripts that hold the deterministic half of the protocol.

Two parts, two different problems

Every L9 protocol implementation splits into two parts with genuinely different lifecycles, and conflating them is the usual mistake:

Part	Responsibility
1. Detection and protocol selection	Observe agent communication — or an agent's own reasoning — and determine when a protocol should be triggered
2. Protocol execution	Define how participating agents conduct the protocol once it has been triggered

Table 13. The two-part structure of an L9 protocol implementation.

Part 1 — detection and selection

Detection evaluates the current state against a set of rules and fires when one matches. It can run in two modes, and the second is easy to overlook. **Communication-driven** detection reads the messages between agents. **Execution-driven** detection is an agent evaluating its own position *while executing*, with no communication trigger at all — agents can agree a plan using a shared term and only discover mid-execution that their working interpretation has drifted from what was agreed. That is a DR-1 in flight, and the protocol can fire internally, before any conflicting message is sent.

The mechanism may be semantic (a model reading the last few messages) or static (pattern matching, policy checks). What varies far more is *how much the detector can see*, and that is decided entirely by where you put it — which is what Figure 8 lays out.

- **A — agent skill.** The protocol is a skill whose front matter states when to invoke it, so the agent self-selects. No code changes, and detection shares the agent's own context. The cost is that it cannot be enforced.
- **B — framework intercept.** Patch the send function; detection runs in process with full access, and a triggered send fails with a structured error telling the agent to run the protocol and retry. Reliable, but you must own the framework.
- **C — CLI wrapper or transport interceptor.** Replace the CLI the agent calls, or put a sidecar below it. No framework dependency; the wrapper needs a config change, the interceptor is transparent but heavier to operate.
- **D — framework hook.** Register at a lifecycle point such as a pre-tool-call hook. The agent never learns the hook exists — it simply gets an error and acts on it. Guaranteed interception, but detection runs outside the agent's context.
- **E — async observability.** An external platform watches traces across the MAS and injects a trigger when it sees a coordination failure.

Option E is the one worth dwelling on, and not for the obvious reasons. It needs no changes to agents, frameworks or tooling, and works across heterogeneous agents. More importantly it can detect *patterns that span multiple agents or accumulate over time* — cross-agent drift, long-horizon misalignment — which no single-message intercept can see. And because it is not on the critical path, its detection logic can be arbitrarily expensive: multi-step reasoning, large context windows, ensemble rules that would be unacceptable latency inside a synchronous hook. That is a direct match for the cognitive observability substrate the taxonomy study argued for. The cost is equally direct: it fires *after the fact*.

■ The choice between A–D and E is the choice between catching a cognitive failure *before* it propagates and catching one that only becomes visible after it has. Both are needed, for different challenges: a DR-1 unit mismatch is catchable pre-send; a CS-1 herding pattern is not visible in any single message at all.

Choosing reduces to four questions, in order. Is async detection acceptable? Then E. Is a pre-tool-

call hook available? Then D. Do you own the framework's communication layer? Then B. Does the agent use an external CLI you can replace? Then C. Otherwise A.

Part 2 — protocol execution

Once triggered, agents need to know how to conduct the protocol in both roles. The **initiator** opens it and sends the first message; a **participant** recognises the protocol, answers in schema, and contributes to resolution. One agent routinely holds both roles at once.

The packaging is one skill per protocol, carrying the instructions for both roles, the message schemas, the sequencing rules — and, importantly, **the scripts the agent calls during the run**. That last part does real work. Asking a model to construct a protocol message from prose alone is a reliability tax you do not need to pay: give it a tool that formats and sends the message, a script that tallies scores, a helper that persists the result. This is exactly how CONCORD is built, with REPAIR the only step delegated to the model and every other line executed by a deterministic engine — which is also what makes the certificate auditable.

Where you own the communication layer, add sequencing guardrails that reject out-of-order messages and return a correction, turning soft skill instructions into hard enforcement.

Portability: one binding per transport

A protocol is portable only if it is specified independently of the transport it rides on. Each L9 protocol is therefore a self-contained definition — description and trigger conditions, an initiator skill, a participant skill, JSON schemas per phase — plus one **transport binding** per L8 it supports. A binding covers exactly three concerns: *field mapping* (where each protocol field lives in the L8 envelope — for A2A, the protocol URI in `message.extensions`, phase and role in `message.metadata`, content in `message.parts`); *discovery* (how a remote agent advertises support — for A2A, the agent card's extension list and the skill's declared media types); and *session correlation* (how multi-turn exchanges are tied together — for A2A, the existing `contextId`, since a protocol run belongs inside the conversation that triggered it).

If you control every agent, you can assert uniform support by construction and skip discovery entirely — which is what our experiments do. The moment third-party agents enter, discovery becomes mandatory. That is the point at which an L9 protocol stops being an application-level convention and becomes a protocol in the ordinary sense: something two parties who have never met can agree to speak.

What this costs you

Worth being concrete about the deployment story, because it is unusually cheap. The SRE triage integration is an overlay: one file adds the orchestrator skill to `sre` and the receiver skill to the five specialists, and the protocol is live. Nothing in the application changed. No framework was rewritten, no transport was replaced, and the baseline remains runnable as the control arm. That is Option A, and it is the right default for a MAS you own. The heavier options exist for the cases where you cannot change the agents — and Option E exists for the case where the failure you are chasing was never visible in a single message to begin with.

7. Where this leaves us

CONCORD delivers what it promises, on every decision we gave it. Across four different problems — an incident triage, a customer renewal negotiation, an e-commerce promotion and an information-pooling puzzle — **no agent in any committed run finished below the satisfaction floor**. Between 93% and 99% of agents ended fully satisfied, 98–99% of scores were consistent

with the reasoning given for them, and compliance reached 100% on two of the four. The worst-off agent averaged 0.73 to 0.86 against a floor of 0.60. A run costs 876K tokens, 78% of them served from cache, against a ceiling published before it starts. That is a consensus mechanism that works — and works well outside the setting it was built for.

And it spends only where spending buys something. The repair loop — the part that makes CONCORD more than a vote — fired in **80% of e-commerce runs**, where the specialists are separate vendor systems with genuinely incompatible objectives, and in almost none of the triage runs, where the agents' opening proposals already cleared the floor for everyone. That is the anytime property doing its job: commit in round 1 when the opening proposals already satisfy everyone, build a new option when they do not. Cost tracks the difficulty of the decision rather than the length of the protocol — which is the behaviour you want from something sitting in the path of every contested call.

ACCORD's mechanism is proven; its payoff is not yet. The protocol locks a shared contract in 82.7% of runs, and the agents demonstrably use it: **99.5%** of locked terms are referenced again after the lock, and quote-backed glossary uptake rises **17.9 points** over the matched baseline. Runs that fail to lock fail early and cheaply. What we cannot yet show is that any of this changes the outcome: goal success is flat against the baseline, and the protocol costs 379K prompt tokens a run to get there. The honest phrasing is “no measurable improvement”, not “improvement” — and that is the one genuinely open result here.

That gap has an address, which is the point of a four-dimension benchmark. An A/B on outcome quality would have reported “no improvement, seven times the cost” for ACCORD and stopped — and it would have said nothing at all about CONCORD, whose benefit is a certificate rather than a score. The dimension-by-dimension read separates the two cases and names the exact step that fails in the one that does: the contract is built, the agents read it, and roughly one agent in four then reasons as though it had not. A conditional influence of 0.754 is a specific, addressable defect at a specific point in the loop, and no outcome metric could have found it.

A protocol is not a prompt. That is the claim underneath all of this. Each of these converges by construction, publishes an exact cost ceiling before the run starts, is anytime so a deadline can cut it short without invalidating anything, and emits an auditable artifact: an Intent Contract, or a certificate that either proves every agent cleared its floor or names the one that did not and by how much. A protocol that reports success over the agents it happened to hear from is worse than no protocol, and both are built to refuse that.

And insertion matters as much as the protocol. An L9 that requires rewriting a working MAS is an L9 nobody runs. In the system we tested the protocol arrives as an overlay: one file, two skills, nothing in the application changed. For agents you do not control there are heavier paths, and for failures never visible in any single message there is async detection over the cognitive observability substrate. No single insertion point is sufficient.

The short version. Two protocols, four dimensions, measured. **ACCORD builds the shared contract it promises and the agents use it** — 99.5% of locked terms get referenced again, uptake up 17.9 points over the baseline — but it has not yet shown a better outcome for the tokens it costs, and we are reporting that plainly rather than waiting for a cleaner run. **CONCORD works:** every agent above the floor in every committed run across four kinds of decision, scores that match the reasoning behind them, a cost ceiling known before the run, and machinery that engages exactly when the problem is hard and stays out of the way when it is not. **And both go in without a rewrite** — in the system we tested the protocol arrives as an overlay, one file and two skills, with nothing in the application changed. Where the agents are not yours there are heavier insertion points, and where the failure is invisible in any single message there is async detection over the cognitive observability substrate. That is the part which decides whether any of this ships: a layer

nobody can adopt is a layer nobody runs.

References

1. Fleming, C., Muscariello, L., Pandey, V., Kompella, R. **A Layered Protocol Architecture for the Internet of Agents**. arXiv:2511.19699. <https://arxiv.org/html/2511.19699v3>
2. Kompella, R., Pandey, V. **Mind the semantic gap: A case for the 9-layer OSI model**. Outshift by Cisco, 2026. <https://outshift.cisco.com/blog/ai-ml/mind-the-semantic-gap-osi-model>
3. Pandey, V. **Internet of Cognition** (vision paper). Outshift by Cisco, 2026. <https://outshift.cisco.com/internet-of-cognition>
4. **Silent Failures, Loud Consequences: Cognitive Challenges in Multi-Agent Systems**. Outshift by Cisco, 2026. <https://outshift.cisco.com/blog/ai-ml/cognitive-challenges-in-multi-agent-systems>
5. Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., Stolica, I. **Why Do Multi-Agent LLM Systems Fail?** arXiv:2503.13657, 2025. <https://arxiv.org/abs/2503.13657>
6. Qian, C. et al. **ChatDev: Communicative Agents for Software Development**. ACL 2024. arXiv:2307.07924. <https://arxiv.org/abs/2307.07924> · Public traces: <https://github.com/OpenBMB/ChatDev>
7. Wang, Y. et al. **MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark**. NeurIPS 2024. arXiv:2406.01574. <https://arxiv.org/abs/2406.01574>
8. Li, Y., Naito, A., Shirado, H. **Systematic Failures in Collective Reasoning under Distributed Information in Multi-Agent LLMs**. ICML 2026. arXiv:2505.11556. <https://arxiv.org/abs/2505.11556> · HiddenBench: <https://github.com/jonradoff/hiddenbench>