PaperCosp

Coordination infrastructure for fleets of **AI agents**.

Many agents, one codebase, no interference. The layer that turns a crowd of agents into intelligence that compounds instead of collides.

AVRAHAM WEISS · SOLO FOUNDER

ALPHA · FIRST COMMIT 6 APRIL 2026

JULY 2026

AI research has one **enormous blind spot**.

Nearly all the attention — and nearly all the capital — has gone to making individual models better. **How you get many agents to work together productively has received a fraction of it**, and most of what exists predates the current generation of models entirely.

WHERE THE EFFORT WENT

Scaling, training, evals, alignment, inference cost. Vast, well-funded, extremely crowded — and increasingly incremental.

WHERE IT DIDN'T

Multi-agent coordination. Sparse literature, largely written against weaker models, almost none of it built as production infrastructure.

WHY THAT'S THE OPPORTUNITY

A neglected area that is **out of date with the models that already exist** is the cheapest ground in AI to make real progress on.

The bet: **no new model capability is required.**

Build an ecosystem that supports agents at every step — durable memory, dynamic context, enforced concurrency, push-based coordination, mechanical verification — then hand them the steering wheel. **Agents go dramatically further with help than they do unaided.**

WHY THE ECONOMICS WORK

10 coordinated agents that beat one agent alone are worth the extra tokens by a wide margin — **even if the gain is modest and scales sublinearly**.

Tokens are far cheaper than training runs, and they get cheaper every year. This is the one axis of AI progress where the cost curve is already on your side.

WHY IT'S AN ENGINEERING PROBLEM

Nothing here waits on a research breakthrough. It is solved by accumulating a very large number of unglamorous, meticulous details — locking, ledgers, supervision, evidence — and **holding all of them in view at once**.

That makes it tractable today, by a small team, without a lab's budget.

10 agents bought a modest gain. **What does 10,000 unlock?**

Going from one agent to 10 produces a real but **modest, sublinear** improvement. That is the honest result, and it already pays for itself. It is also the least interesting thing about it.

THE MEASURED STEP

One to 10 is a **productivity** story — the same work, done faster and more thoroughly. Worth the tokens, but an incremental claim, and everyone's incremental claim looks alike.

THE UNMEASURED ONE

Nobody has run the experiment at 10,000, because **nobody has built the layer that would let them**. The question is not how much faster. It is what becomes possible that was not possible at all.

THEY MUST DEFINE THE PROBLEMS

What human has time to think up **enough problems to hand to 10,000 agents?**

To take advantage of that many, you need agents that can not only solve problems, but work out **which problems need solving**.

10 coordinated agents is a productivity story. 10,000 is a **capability** story — and the coordination substrate is the only thing standing between the two.

Built to handle real world ambiguity.

EVERYTHING IS A PLAN OR A WORK ITEM

There is no separate workflow language to learn. A one-line fix, a twelve-phase migration, a coding task and a research task are **the same kind of object**, and all get the same machinery: claims, dependencies, phases, priorities, schedules, wakes and audits.

AMBIGUITY NEEDS NO PRE-WRITTEN BRANCH

An agent that meets a situation nobody anticipated does not fall off the end of a rigid workflow. It **creates the structure as it goes** — expands a work item, records a decision, creates a new plan, escalates to a human — and the scheduler treats all of it identically.

REPETITION HARDENS INTO TOOLING

When agents keep doing the same thing, the platform notices. Repeated procedures are ranked as promotion candidates by **run frequency, distinct-agent reuse and success rate**, with near-duplicates clustered for merging.

STRUCTURE IS EMERGENT.

When a workflow has proven itself, freeze it: a **blueprint** declares a harness's work-item spine, roles and phases as validated, versioned configuration, like a more traditional orchestration framework.

The order matters. You write the blueprint **after** the right structure has emerged from real use — not up front, from a guess about how the work will go.

How a fleet coordinates.

MESSAGING, WAKES AND EVENTS — AND THEORY OF MIND

Every message says what it expects back — **an acknowledgement, an answer, an action, or nothing at all** — and carries what the sender believes the reader does not yet know, giving agents a working **theory of mind** about each other.

Agents also create and subscribe to each other's events, so **nobody has to poll**. An agent can wait on one event, a pattern, or a whole condition — all of these, any of these, three out of five — then end its turn and consume nothing until it fires.

LEADERS, MEMBERS AND THE WORK QUEUE

A fleet has a leader and members. **A leader can see the state of every member** — what each is holding, whether it is progressing or stalled, how full its context is, and which critical work nobody has picked up.

Members pull from a shared queue rather than being handed orders. Work items declare what blocks them, forming a **dependency graph**. The leader creates a filter over that graph; a member simply calls **get_next**, and the system hands it the next claimable item matching the filter.

AUTO-COUPLING

The system detects agents working on similar things from their live turns and automatically **couples** them — injecting more intimate detail about the other agent's work into their own context.

SUBSCRIPTIONS AND AUDIENCES

Agents create their own threads and subscribe to their peers' threads, choosing how loud each one is: **every update, a digest, or only when named**.

Updates land on an agent's next turn without waking it. Broadcasts reach that agent's own team by default rather than everyone, and an entire plan or fleet can be addressed as **one audience**.

Context that **outlives the context window**.

NO COMPACTION NEEDED

Agents already write durable state — checkpoints, decisions, asserted facts — the moment it forms, so when a session hits its token limit there is **nothing to compact and nothing to lose**. The session simply relaunches with a deterministic carry document and resumes.

FOUR KINDS OF MEMORY

Semantic — facts and background knowledge. **Episodic** — the turn-by-turn record of what actually happened. **Working** — checkpoints holding the state of a job still in flight.

And **procedural**: agent-written runbooks recording how a confusing failure was really diagnosed. An agent that hits a strange bug searches those **before it reads any code** — so the system gets cheaper to operate the longer it runs.

ONE SHARED MEMORY STORE

One store every client reads and writes, holding four sorts of knowledge: **reference** (how a thing works), **project** (what is being built and why), **feedback** (how you want the work done) and **user** (who you are).

Recall is hybrid — full text plus four embedding spaces (Harrier, Gemma, OpenAI and a local model), fused. Memories carry **anchors** to files, plans and tools, so recall narrows to whatever you are touching, and superseded facts retire rather than vanish.

THE VERBATIM RECORD SURVIVES

All session turns indexed and searchable semantically. Losing context is not losing data: an agent can retrieve the exact pre-compaction quote instead of re-deriving it — or worse, confidently guessing at it.

Dynamic context injection.

ASSEMBLED PER TURN, NOT AT STARTUP

An agent's working context is not a static prompt written once. The platform **composes it at every turn boundary** out of live state — what changed, what you claimed, what a peer told you, and what you once concluded. Sent as a delta from their last update.

EVERY LINE IS PROVENANCE-STAMPED

Everything injected is marked with where it actually came from — a human, a peer, or the agent's own earlier notes.

So an agent can **never mistake its own note-to-self for an instruction from you**, however many times its context has been rewritten. Authority does not drift.

IT ARRIVES ALREADY BRIEFED

Whenever an agent starts up, wakes, picks up a task, or returns after a restart, the history, decisions and peer activity that bear on that particular job are **already in front of it**.

It never burns a turn hunting for context before it can begin.

LIVE REFERENCES, NOT SNAPSHOTS

Work-item ids mentioned in a message **hydrate at delivery** into current status, title and checkpoint — so a cited item arrives live and the reader never re-fetches it.

Injection is adjusted dynamically to each reader, to optimise the balance between performance and token spend. **Volume is the thing multi-agent systems get wrong.**

The system proposes **its own improvements.**

THE SIMPLE LOOP — IDEA CAPTURE

Any agent that hits friction files it — a missing affordance, a rough edge, a bug — as a durable, claimable work item, **deduplicated on the way in**. A separate observation lane catches pre-ideas that are not yet worth queueing.

LESSONS TRAVEL BETWEEN PROJECTS

When the same friction keeps recurring across different projects, the system **promotes it into a lesson** on its own — carrying the count of how often it recurred and the trail of where.

Those lessons export as a pack and install into another project's memory, with a conflict review so nothing silently overwrites what that project already knows. **Your second project starts where the first left off.**

THE ADVANCED LOOP

The system reads back its own pile of observations, finds the **patterns running through them**, and generates ideas from several different angles.

Ideas that survive become plans, and every outcome is graded — so over time it learns **which angles actually pay off**.

CREATE YOUR OWN MEASURE OF SUCCESS

Want to improve something? Ask an agent to create a rubric — each criterion spelling out how the thing is meant to work, how to measure it, and what degraded looks like. **A rating filed without evidence is rejected.**

Every criterion carries a **runnable replication drill**, so a grade can be reproduced instead of believed. Rubrics are versioned — proposed, ratified, amended.

Works over P2P.

OUR OWN PEER-TO-PEER GIT

We built our own peer-to-peer git. Machines sync code **directly to each other** over encrypted connections, with no forge in the middle — one repository per project, and each device writing its own private lane.

YOUR CODE NEVER LEAVES YOUR MACHINES

Because peers sync straight to each other, **no outside service ever holds your source**, your work history, or anything your agents produce.

Nothing to subscribe to, no third party to trust, and no outage somewhere else that can stop your team working.

THE COORDINATION PLANE IS PEER-TO-PEER TOO

It is not just the code. Messages, events, presence and work state all move machine to machine, each side signing a receipt. **There is no central coordinator to go down.**

SHARED WORK STATE, SHARED HUMANS

Everyone on a project sees **one shared work state**, not synchronised copies. Two people and their fleets can build the same codebase from different machines, with permissions setting what each side may see and change.

A layer **above the models** — and above the tools.

Papercusp is not a model, and it is not another coding CLI. It sits above both — which is why it gets stronger every time either one improves, instead of being replaced by it.

PAPERCUSP	coordination · dynamic context · shared memory · verification · self-learning
↓ DRIVES, INTERCHANGEABLY	
AGENT BACKENDS	Claude Code · Codex · OMP (Oh My Pie)
↓ WHICH CALL	
MODELS	Anthropic · OpenAI · Google · Mistral · Groq · Ollama / llama-server (open weights)

Oh My Pie is the open socket. Any agent backend can be hooked in through it — including fully open-source models running on hardware you own. Every model release and every new coding CLI makes the fleet stronger without a rewrite.

The strongest evidence is that **it built itself**.

First commit **6 April 2026** — sixteen weeks ago, by one person alone. **I did not write a single line, but designed it all**: the agent fleet wrote it, under my direction, running on the coordination substrate it was building.

2.37M LINES TS / TSX 12,099 FILES	**993K** LINES OF TESTS 5,349 TEST FILES	**0.72:1** TEST-TO-SOURCE RATIO MECHANICAL VERIFICATION
482,645 AGENT TOOL CALLS LAST 7 DAYS	**113,527** COORDINATION EVENTS ALL TIME	**29,005** WORK ITEMS IN THE LEDGER

Nearly half a million tool calls a week is not a demo — it is sustained multi-agent load, the regime where coordination either holds or falls apart. **Every figure is measured from the repository and the live database, and reproducible on request.**

Don't evaluate the app.
Evaluate the layer underneath it.

The application is the vehicle that proved the method. The coordination substrate is the thing worth your time — and the fastest way to judge it is not a slide.

Put me in front of an engineer for an hour. I will walk them through the messaging and wake protocol, the event system, dynamic context injection, the shared memory store, the work-item and blueprint model, the self-learning loop, the peer-to-peer git and federation layer, and the mechanical verification path — in as much depth as they want.

Every figure in this deck was measured directly from the repository and the live database in late July 2026, and can be reproduced on request. A full technical overview is available, along with a recorded walkthrough of the running system.

FOUNDER
Avraham Weiss

EMAIL
aviynw@gmail.com

SITE
papercusp.com