OPUS — Whitepaper

Ars Magna · A Bio-Inspired Multi-Agent Swarm Architecture for Collective Reasoning

Version 0.1 · MMXXVI

---

§1 — Premise

OPUS is not a model. It is a colony.

A single language model, however large, reasons in one voice. It produces a stream of plausible tokens, defends them, and moves on. It cannot meaningfully disagree with itself. It cannot triangulate. It cannot be falsified except from outside.

We replace that lonely soliloquy with a structured swarm.

Three concentric tiers — **Scouts** at the perimeter, **Workers** in the middle, a **Hive Core** at the centre — coordinate not by speaking to each other but by writing typed records to a single shared substrate: the **Blackboard**, an append-only event log. Each agent reads the current state of the Blackboard, performs one unit of cognition, and writes its result back. No agent has a private channel to any other. The environment is the conversation.

This is the stigmergic principle, observed first in termite mound construction by Grassé in 1959 and formalised in computer science as the blackboard architecture by Hearsay-II at Carnegie Mellon between 1971 and 1976. It is not new. What is new is applying it, with discipline, to large language models — to obtain something a single model cannot produce: a *deliberation*, with an audit trail, a confidence, and a falsification attempt baked in.

When the colony has deliberated enough, three stages of consensus run:

1. **Borda aggregation** — weighted ranking across all Worker outputs.
2. **Judge adjudication** — an LLM-as-Judge resolves near-ties.
3. **Verifier pass** — a final agent attempts to *falsify* the chosen answer.

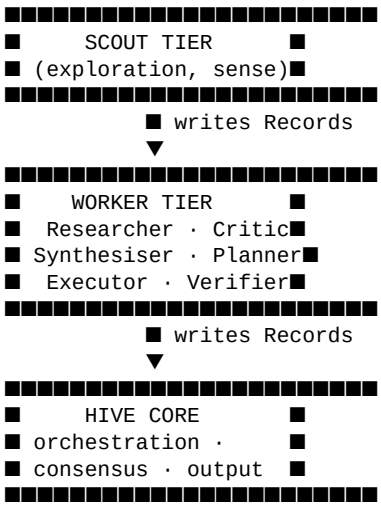
If verification fails, the swarm re-deliberates with the falsification as a new constraint. The loop is bounded: at most three attempts, after which the colony surfaces what it has, with confidence and trace.

That is the Great Work — *solve et coagula*. Dissolve a single mind into many; recombine the many into one well-considered answer.

---

§2 — Architecture

2.1 The three tiers



Each tier has a distinct cost profile and reasoning style.

- **Scouts** are cheap and many. They retrieve, sense, and reconnoitre. Default model: `claude-sonnet-4-6`. They do not opine; they bring back evidence.
- **Workers** are the deliberators. They are fewer in number but more capable. Default model: `claude-opus-4-7`. Each Worker has a single role — Researcher, Critic, Synthesiser, Planner, Executor, Verifier — and operates on the same Blackboard.
- **Hive Core** is one entity per query. It schedules agent rounds, enforces budgets, runs consensus, and emits the final Record.

2.2 The Agent contract

Every agent is a subclass of a single base class. Its only required method is:

```
async def think(self, task: Task, blackboard: Blackboard) -> Record:
    ...
```

Agents are stateless across queries. All state lives on the Blackboard or in long-term memory (vector + graph). This is deliberate: an agent is a *behaviour*, not a service.

---

§3 — The Blackboard

The Blackboard is an append-only log of typed Records. A Record is a pydantic model with strict typing:

field	type	meaning
`id`	UUID	globally unique
`agent_id`	str	identity of the agent that wrote this Record
`agent_role`	enum	scout · researcher · critic · …
`parent_ids`	list[UUID]	which Records this Record reasons from
`type`	enum	observation · hypothesis · critique · synthesis · verdict
`content`	str / json	the Record's payload
`confidence`	float ∈ 0–1	the agent's self-assessed confidence
`model`	str	the model used
`cost_estimate`	USD	cost of producing this Record
`timestamp`	ISO 8601	when written

3.1 Why append-only

Mutability invites coordination problems we do not need. Append-only gives us:

- A **causal DAG** for free, via `parent_ids`.
- **Time-travel debugging** — replay any past Blackboard state.
- A **complete audit trail** of provenance, no extra plumbing.
- **Optimistic concurrency** — writes never conflict; readers never block.

3.2 Why not CRDTs

V0 runs in a single Python process. CRDTs solve a distributed-consistency problem we do not yet have. The Blackboard interface is abstracted (`Blackboard.append`, `Blackboard.read`, `Blackboard.subscribe`) so a Redis Streams backend, and eventually a CRDT-based one, can slot in without touching agent code. We will adopt that complexity when the workload justifies it; not before.

---

§4 — Consensus

Three stages, each independently testable.

4.1 Borda aggregation

Every Worker that produces a Synthesis Record also produces a ranked list of all candidate Syntheses on the Blackboard (their own included). We aggregate using **weighted Borda count**: each Worker's vote is multiplied by a role weight (TODO — defaults below; reviewed by the user). The candidate with the highest aggregate score advances.

> *Default role weights for v0: Researcher 1.0 · Critic 1.2 · Synthesiser 1.5 · Verifier 1.3 · Planner 1.0 · Executor 0.8.*
>
> *Critics and Verifiers are weighted higher because their failure mode (false negatives) is less harmful than a Synthesiser's (false positives). These numbers are starting points; tune empirically.*

4.2 Judge adjudication

If the top two candidates are within $\varepsilon = 5\%$ of each other on aggregate Borda score, a single Judge agent (model: `claude-opus-4-7`) reads both candidates *and* the trace that produced them, and selects a winner with reasoning. The Judge's verdict is itself a Record on the Blackboard.

4.3 Verifier pass

The chosen Synthesis goes to a Verifier agent whose job is the opposite of synthesis: *find what is wrong with it*. The Verifier produces one of two Record types:

- `verdict.accepted` — no falsification found. The candidate becomes the final answer.
- `verdict.falsified` — the Verifier returns a specific falsification (a counter-example, a contradicted axiom, a missed constraint). This Record is then injected into a new round of deliberation as a hard constraint.

Verification is bounded: at most **three** attempts. After the third, the colony surfaces the strongest remaining candidate along with the unresolved falsifications, marked clearly. The user is never lied to about the colony's certainty.

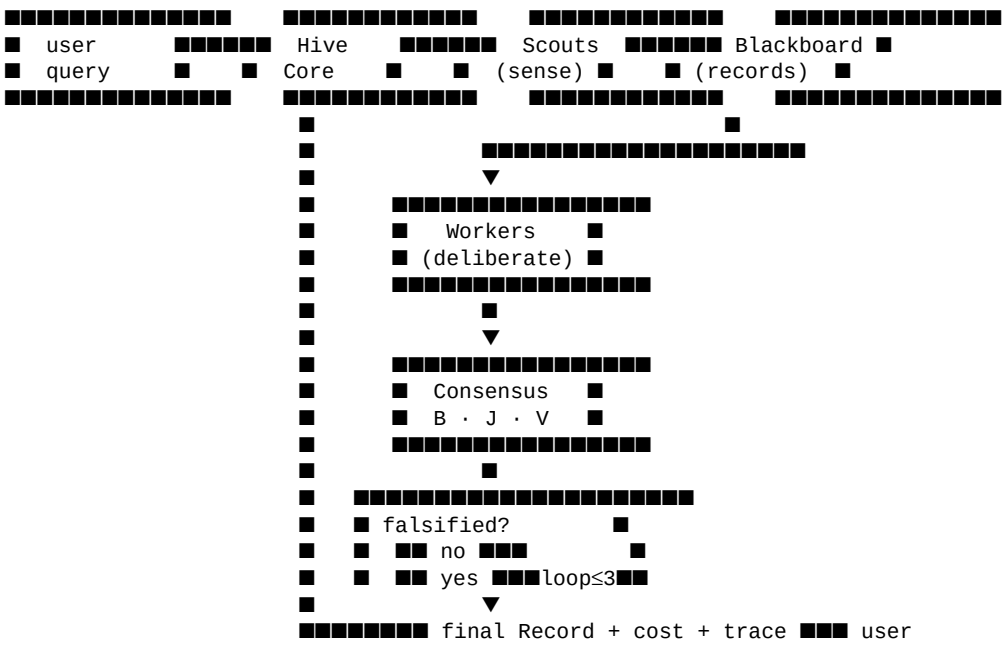
---

§5 — Tech Stack

Layer	Component	Status
Models	Claude Opus 4.7 (Workers, Judge, Verifier)	**ACTIVE**
Models	Claude Sonnet 4.6 (Scouts, light critique)	**ACTIVE**
Async runtime	Python `asyncio` + `anyio`	**ACTIVE**
Validation	pydantic v2	**ACTIVE**
Logging	structlog (JSON)	**ACTIVE**
CLI	typer	**ACTIVE**
Packaging	hatchling	**ACTIVE**
Tests	pytest, pytest-asyncio	**ACTIVE**
Vector memory	Qdrant (interface stubbed)	PLANNED
Graph memory	Neo4j (interface stubbed)	PLANNED
Distributed bus	Redis Streams (Blackboard backend)	PLANNED
Observability	OpenTelemetry traces	PLANNED
Web	Next.js 14 + Three.js + GSAP	**LIVE**

---

§6 — Lifecycle of a single query



Concretely, a default v0 query runs:

- 3 Scouts in parallel
- 6 Workers (2 Researchers, 2 Critics, 1 Synthesiser, 1 Verifier) over 1–3 rounds
- 1 Judge (only if Borda is close)
- 1 Verifier pass per round

Token + dollar budget is enforced by the Hive Core. Any agent attempting to exceed budget is short-circuited with a `verdict.budget_exhausted` Record.

---

§7 — Provenance & Cost

Every Record carries its `model`, `cost_estimate`, and `parent_ids`. The Hive Core writes a final `provenance.summary` Record at end-of-run containing:

- Total wall time
- Total tokens (input + output) per model
- Total USD spent (using the official Anthropic pricing table, hard-coded and version-stamped in `provenance.py`)
- The full causal DAG of Records, serialisable to JSONL for replay

The CLI surfaces three things at end of run: **the answer · the cost · the trace path**. Nothing else by default. Verbosity is opt-in.

---

§8 — What we are not building (yet)

Stated plainly so future contributors do not assume otherwise.

- No authentication, accounts, or login.
- No payments, billing, credits, or virtual currencies of any kind.
- No persistent user database.
- No REST or GraphQL surface — CLI only for v0.
- No Docker, Kubernetes, or production deployment.
- No CI/CD until there is something to deploy.
- No marketplace, third-party agents, or governance — Phase β and beyond.

---

§9 — Open editorial decisions (TODO before v1)

These were defaulted in the v0.1 draft and should be reviewed by the project lead:

1. **Borda role weights** — defaulted in §4.1. Tune empirically once we have a representative query set.
2. **Verification ε threshold** — defaulted to 5%. Sensitive to model temperature.
3. **Default agent prompts** — first-pass prompts live in `opus-core/src/opus/agents/*.py`. These are *the product*; they want hand-crafting.
4. **Budget defaults** — currently unbounded in v0.1. Set sensible USD ceilings before public demos.
5. **Vector + graph schemas** — interfaces stubbed, no schema decisions yet.

Magnum Opus · MMXXVI