Codessian Adaptive Orchestrator — Spooky Logic (v0.1)

> Become the board. Absorb the game. Rewrite the rules.

0) Mission

Build an **Adaptive Orchestrator** that self-observes, reconfigures its agent ecology, and absorbs external capabilities. v0.1 delivers self-tuning orchestration over a minimal multi-agent crew, backed by MHE memory, policy-as-code governance, and experiment-driven improvement.

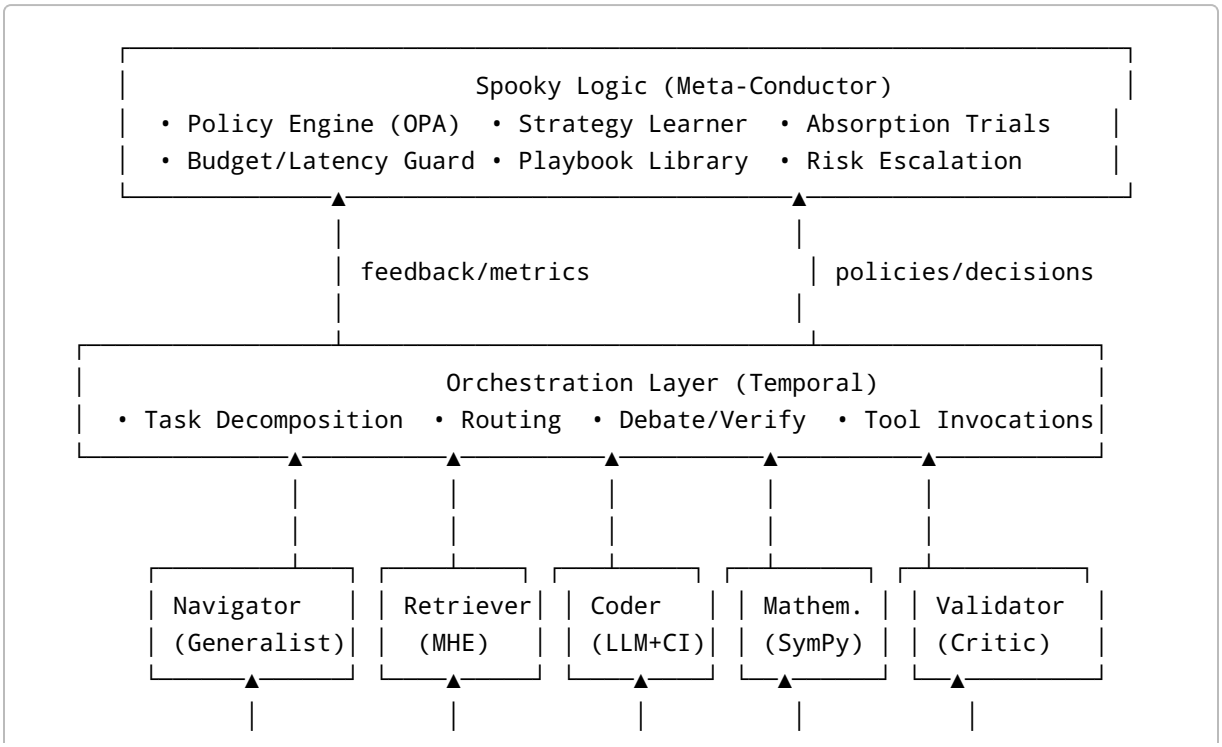
---

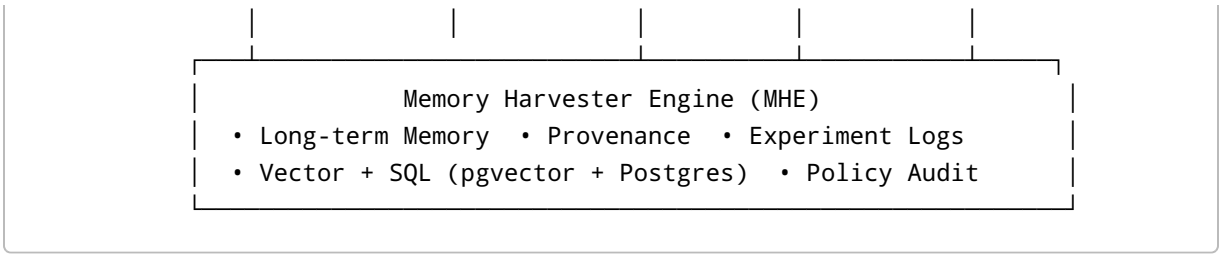
1) Design Goals

- **Self-tuning**: pick strategies/agents based on live metrics (accuracy, cost, latency).
- **Absorption**: discover → trial → integrate external tools/models.
- **Governed**: OPA policies gate capability use, data egress, spend, and safety.
- **Composable**: Temporal/Ray workflows; hot-swappable agents via a Router.
- **Observable**: exhaustive traces, metrics, and provenance into MHE + Prometheus.

---

2) High-Level Architecture





---

3) Minimal Crew (v0.1)

- **Navigator**: frontier LLM (GPT-4o / Claude 3.5) → goal parse + task graph.
- **Retriever**: MHE hybrid search (BM25 + pgvector) → context and sources.
- **Coder**: DeepSeek-Coder or o1-mini for codegen + unit-test loop.
- **Mathematician**: SymPy + math-tuned LLM for formal steps.
- **Validator**: adversarial LLM critic (constitutional prompts + heuristics).
- **Scribe**: logs steps, metrics, and artifacts to MHE; updates provenance chains.

Start broad; specialize later (e.g., SQL-Agent, Web-Agent, Vision-Agent) when bottlenecks appear.

---

4) Data & Memory

Core tables (Postgres): - `tasks(id, parent_id, user_goal, strategy, status, metrics_json, created_at)` - `artifacts(id, task_id, type, uri, hash, provenance_json)` - `experiments(id, name, control_playbook, variant_playbook, outcome, stats_json)` - `agents(id, name, kind, endpoint, caps_json, costs)` - `policies(id, name, rego_module_ref, enabled)`

Vectors (pgvector): - `mem_chunks(id, task_id, content, meta_json, embedding)` - `playbooks(id, name, description, embedding, yaml)`

Provenance: every orchestration step writes `{who, when, inputs, outputs, policy, model, cost}`.

---

5) Policy-as-Code (OPA/Rego)

Example: budget guard & red-flag escalation

```
package spooky.budget

default allow = false

allow {
```

```
  input.estimated_cost <= input.budget.max
}

escalate {
  input.estimated_cost > input.budget.max
}
```

Example: tool egress restriction

```
package spooky.egress

allow_tool[tool] {
  tool := input.tool
  allowed := {"mhe.search", "sympy.solve", "pytest.run"}
  tool.name ∈ allowed
}
```

Example: auto-debate trigger

```
package spooky.quality

debate_required {
  input.task.risk >= 3
}

second_opinion {
  input.validator_error_rate > 0.25
}
```

---

6) Orchestration Playbooks (YAML)

Control: single-pass

```
name: control_single_pass
steps:
  - route: navigator
  - retrieve: mhe.hybrid_search
  - solve: primary_agent  # coder|math chosen by task type
  - validate: validator
  - decide: accept_if(conf>=0.7 and validator.ok)
```

Variant: debate + toolformers

```
name: variant_debate_tools
steps:
  - route: navigator
  - retrieve: mhe.hybrid_search
  - parallel:
      - solve_a: coder
      - solve_b: coder_alt
  - debate: {mode: "cross-exam", rounds: 2}
  - tool_calls:
      - math: sympy
      - tests: pytest
  - validate: validator
  - decide: select_best_by(score=accuracy-0.1*cost-0.05*latency)
```

Router overlay (excerpt)

```
roles:
  navigator:
    candidates: [gpt4o, claude35]
    fallback: gpt4o_mini
  coder:
    candidates: [deepseek_coder, gpt4o_mini]
    tests: pytest_local
  validator:
    candidates: [claude35_mini]
  math:
    tools: [sympy]
```

---

7) Temporal Workflow Sketch (Python)

```python
@workflow.defn
class Orchestrate:
    @workflow.run
    async def run(self, task: Task):
        playbook = select_playbook(task)
        ctx = await act("navigator", task)
        evid = await act("retriever", ctx)
        result = await execute_strategy(playbook, ctx, evid)
        verdict = await act("validator", result)
        record_metrics(task, result, verdict)
```

```
        if not verdict.ok and should_debate(task):
            result = await run_debate(ctx, evid)
        await persist(task, result, verdict)
        return decide(result, verdict)
```

---

8) Adaptive Loop (Meta-Conductor)

1. **Observe**: aggregate metrics per playbook/agent (accuracy, cost, latency, error types).
2. **Compare**: run scheduled A/B trials; compute uplift with Welch's t-test.
3. **Decide**: if variant outperforms control ($p<0.05$) for N tasks → **promote** variant.
4. **Refine**: update router weights; revise policies; write a "Playbook Delta" to MHE.
5. **Guard**: OPA checks for drift (cost spikes, hallucination rate, policy violations).

Promotion rule (pseudo-rego):

```
promote_variant {
  input.stats.uplift_accuracy > 0.03
  input.stats.p_value < 0.05
  input.stats.cost_delta <= 0.10
}
```

---

9) Absorption Pipeline (External Capability Intake)

- **Discovery**: catalog new tools/models (manifests + caps).
- **Sandbox**: run benchmarks & red-team suites.
- **Trial**: shadow-run on 10% of eligible tasks behind a feature flag.
- **Compare**: evaluate vs incumbent; log to `experiments`.
- **Integrate**: add to router, update OPA allowlist, publish playbook update.

Tool manifest (JSON)

```
{
  "name": "perplexity.search",
  "kind": "api",
  "latency_ms": 800,
  "cost": {"per_call": 0.002},
  "capabilities": ["web_search", "citations"],
  "egress": ["https://*.perplexity.ai"],
  "safety": {"pii_risk": 1}
}
```

---

10) Metrics & Telemetry (Prometheus)

- `spooky_task_accuracy{playbook,agent}`
- `spooky_cost_usd{playbook,agent}`
- `spooky_latency_ms{step}`
- `spooky_validator_error_rate{domain}`
- `spooky_absorption_win_rate{tool}`
- Traces: OpenTelemetry spans per step; link to artifacts/provenance IDs.

Grafana boards: **Overview**, **Quality**, **Spend**, **Latency**, **Absorption Trials**.

---

11) Safety & Governance

- **OPA gates** for budgets, tool egress, high-risk content, private-data scopes.
- **Human-in-the-loop** escalations for risk≥4 tasks.
- **Provenance chains** embedded in outputs; red-team validator always on for public-facing tasks.

---

12) Minimal Milestones

M0 – Skeleton (Week 1)
- Temporal workflow + FastAPI ingress.
- Navigator/Retriever/Coder/Validator wired to MHE.
- Prometheus + Grafana dashboards.

M1 – Adaptive Loop (Week 2)
- Playbook A/B, metrics aggregator, router weights update.
- OPA budget/safety rules enforced.

M2 – Absorption v0 (Week 3)
- Tool manifest loader, sandbox runner, shadow-trial + promotion rule.

M3 – Guardrails & Debates (Week 4)
- Auto-debate triggers, second-opinion policy, structured validator reports.

---

13) Test Harness

- **Gold tasks** per domain with deterministic or human-scored rubrics.
- **Canary set** for cost/latency smoke tests.
- **Red-team set** (prompt injections, jailbreaks, specious citations).
- CI: run harness on each playbook change; block promotion if regressions.

---

14) Operator Controls (FastAPI)

Endpoints
- `POST /orchestrate` submit task
- `GET /tasks/:id` status + provenance
- `POST /playbooks/:id/trial` start experiment
- `POST /agents/register` add tool/model manifest
- `POST /policies/reload` hot-swap OPA bundles

RBAC via OPA; **audit** via MHE.

---

15) Sample Validator Report (Schema)

```
{
  "task_id": "t_123",
  "signals": {
    "logic_consistency": 0.82,
    "citation_valid": true,
    "tests_passed": 7,
    "tests_failed": 1
  },
  "flags": ["missing_edge_case"],
  "recommend": "revise",
  "notes": "Edge case: empty input; propose guard clause."
}
```

---

16) Deployment Notes

- **Linkerd** mTLS across services; **SPIFFE** identities for agents.
- **Ray** optional for parallel debates/tools.
- **NATS JetStream** for event bus (metrics + trial outcomes).
- **Cost Guardrails**: per-namespace budgets enforced by OPA and router quotas.

---

17) What Makes It Spooky

- The *system* learns which **ways of thinking** work, not just which answers.
- It metabolizes rival capabilities via trials and policy-gated promotion.
- It treats coordination itself as a mutable, learnable artifact—**playbooks as memory**.

v0.1 exit criterion: control-beating variant automatically promoted on ≥2 domains with provable uplift and within budget.

18) Next Up (v0.2-v0.3)

- Meta-learning over playbook embeddings (retrieve similar past strategies).
- Causal metrics (DoWhy) to separate correlation vs true uplift.
- Multi-tenant policy packs; per-tenant playbook evolution.
- Emergent crew shaping: auto-spawn new specialists when recurring failure patterns detected.

Spooky Logic isn't a bigger brain. It's a **smarter civilization** of small ones, constantly renegotiating how to think—until the game itself belongs to it.