Petter Synthesis

4.1 — Petter Graff Executive Synthesis

AI Factory Audit — 2026-05-09 Auditor: Petter Graff (CodeCraft — Lead Architect)
Synthesizing: P1 reports 1.1–1.4, P2 reports 2.1–2.3, P3 report 3.1 **Method:** P3.1 live-probe data overrides P1/P2 file-based claims where they contradict.

Section 1 — Executive Summary (Bosnian)

Situacija

John ima dobro zamišljenu arhitekturu: kontrolni sloj sa Mehanik kapijom, memorijski sloj sa pet pohrana, RAG pipeline za znanje, tim od 66 agenata u 12 virtualnih kompanija, i orkestratorski kernel koji bi trebao sve automatizirati. Na papiru to izgleda kao AI fabrika. U stvarnosti, 62.5% advertiziranih tokova podataka i kontrole su mrtvi ili degradirani. Sistem radi kao ručna radionica — John lično proslijedi svaki zadatak, lično provjeri, lično zatvori. Automatizacija postoji kao infrastruktura, ali nije spojena. Ono što funkcioniše: HiveDB/HiveMind intel bus, LightRAG lokalni upis, Mehanik kapija (djelimično), alati (250+ živih), i 74 calendar-scheduled daemona koji rade ispravno. Ono što je teatar: pi-orchestrator (živ proces, nema stvarnih dispatcheva od marta), verify-fix-loop (skill postoji, niko ga nikad ne pozove automatski), mem0 (93K+ vektora, nula aktivnih pisača), četiri "fantomske" kompanije bez routinga, i 35 chain YAML fajlova bez nijednog executora.

5 najkriti?nijih praznina (rangirano po IMPACT × SEVERITY ÷ EFFORT)

1. **RAG ingest pipeline — potpuno blokiran** (Vaultwarden timeout, 3,150+ stavki u redu (posljednji poznati snapshot: 454 dana 2026-04-23; live SQLite prebrojan 2026-05-09 = 3,150), drain-worker pao danas)

2. **pi-orchestrator u mock/broken modu** — kernel živi, ali ne dispatcha ništa od marta 2026; sav dispatch ide kroz Johna ručno
3. **Verifier loop — sposoban ali ne pozvan** — verify-fix-loop skill postoji, nije spojen ni na jedan automatski okidač; CEO je jedini QA gate
4. **Memorijska anarhija** — 5 pohrana, nijedna nije System of Record; mem0 ima 93K vektora koje niko ne piše ni čita; .md fajlovi su defacto SoR, ali to nije dizajnirano tako
5. **Agent routing rupa** — validator (44 pozivanja u skill fajlovima) i distiller (21 pozivanje) nemaju ni jedan unos u specialist-mapping.json; 7 mapirani agenti su fizički nedostupni

Šta popraviti prvo

Jedna stvar otključava više od svega ostalog: **RAG drain-worker** — jedan credential fix (Vaultwarden session za LightRAG CF Access) otključava 3 adaptera odjednom i prazni 454+ stavki iz reda. Direktno za njim: **pi-orchestrator real config** — razumjeti zašto HTTP port 8401 ne radi i zašto nema dispatcheva od marta; bez ovoga, fabrika ostaje ručna. Treće po prioritetu: **verify-fix-loop wiring** — dodati Section 2b u /task-postflight SKILL.md, što ne zahtijeva novu infrastrukturu i odmah uklanja CEO-a iz petlje za docs/system/refactor zadatke. Ova tri fixa su S/M napora i zajednički konvertuju fabriku iz "John kao ručni dispatcher + QA" u nešto što nalikuje automatiziranom sistemu.

Section 2 — Plan vs Reality Delta Table

Subsystem	Plan Claim	Reality (audit-verified)	Delta	Severity
Memory plane	mem0 is the structured SoR for John's personal facts; LightRAG is secondary RAG store	.md files are the actual SoR (Claude Code native). mem0 API has 0 active writers, 865 stale facts. LightRAG is primary RAG (999 docs, healthy). 5 parallel stores, none designated SoR.	Complete SoR inversion; mem0 is a ghost server with stale data nobody reads	H
HiveMind	Intel broadcast bus; P1 implied no read API	HiveDB SQLite 17,560 rows, live writes today. `hivemind.js read/query/semantic_query` all functional. hivemind-mcp.js wraps all. Read API EXISTS and works.	P1 overstated the gap. HiveMind is the healthiest store in the factory.	L

Subsystem	Plan Claim	Reality (audit-verified)	Delta	Severity
Tools shed	250+ live tools, manifest current	443 files on disk; manifest 6 weeks stale; 12 un-owned tools; 50 .bak files >14d old; 1 credential-bearing filename (security risk); 100 dead-code tools	Manifest does not reflect reality. Security artifact present. Dead code accumulating.	M
Agent fleet	29 agents routable via specialist-mapping.json	44% mapping coverage (29/66). validator (44 skill refs) and distiller (21 refs) absent from mapping. 7 mapped agents unreachable on disk. 4 companies invisible to routing. 35 chains have executors (chain-runner.js + chain-runner.sh) but executors are un-wired from active skills and broken at daemon invocation.	Routing table is too thin to be trusted as source of truth. Silent dispatch failures guaranteed.	H
Daemon fleet	148 daemons maintaining system health	20 erroring, 5 scripts deleted (exit 127), 2 in infinite crash loop. RAG pipeline fully deadlocked. Cost reporting dark 10+ days. pi-orch health monitor script deleted.	Monitoring is blind to key system health. 13% error rate.	H
pi-orchestrator	Automated dispatch kernel; picks up MC tasks, fires specialist agents	PID 75750 alive. HTTP port 8401 dead. No dispatch logs post-2026-03-19. Durable-runner bridge (port 3052) live but dispatch activity unclear. Config: offline-mode=false but effectively not dispatching.	Kernel running in operational void. All actual dispatch is manual-John.	H

Subsystem	Plan Claim	Reality (audit-verified)	Delta	Severity
Verifier loop	verify-fix-loop auto-invokes after mc.js ready for eligible tasks	Skill exists, internally correct. Zero wiring to any automated trigger (no hook, daemon, pi-orch code calls it). CEO is de-facto verifier.	Built but unwired. Capability without activation.	H
BUILD-BLUEPRINT discipline	Mehanik enforces blueprint read before any dispatch; 90-point score gate	Blueprint read IS required and enforced as hard block (CB#2). But: WARN scores (65, 80) allow dispatch — 90-point threshold is advisory only. 4 blueprints 59d+ stale. Missing-MC-ID path bypasses gate entirely.	Gate is real but porous. Score enforcement is theater. Session binding absent.	M
Library skill	Skill library accessible for cookbook-based task execution	`node ~/system/tools/library.js list` returns 13 cookbooks, 11 defaults. CLI fully functional.	WORKS. No gap.	L
Virtual companies	12 companies, each routable via discover.js → specialist-mapping.json	4 companies (Axiom, Datavera, Resolver, Lexicon) have full persona dirs, CLAUDE.md, 5–9 internal agents — but zero entries in specialist-mapping.json. Cannot be routed via normal John → discover.js flow.	33% of the company fleet is phantom infrastructure.	M

Section 3 — Top-10 Gaps Ranked

Composite priority = Leverage × Severity ÷ Effort (S=1, M=2, L=4)

#	Gap Name	Subsystem	Evidence	Leverage (1–10)	Severity (1–10)	Effort	Composite	Proposed Fix

1	**RAG drain-worker deadlock**	Daemon fleet / Data plane	1.4 §3, 2.1 §B Dead Edge 2, 3.1 H1 — 3,150 items queued (live SQLite 2026-05-09; stale prom file shows 454 as of 2026-04-23)	9	9	S	**81**	Fix Vaultwarden session so rag-drain-worker can reach LightRAG CF Access endpoint; confirm `/tmp/bw-session` valid.
2	**pi-orchestrator dispatch broken**	Orchestration kernel	1.4 §4, 2.1 §A Dead Edge 1, 3.1 C1/C2	10	9	L	**22.5**	Diagnose why HTTP port 8401 is silent and why no dispatch logs post-March; restore real MC API config or repair durable-runner bridge as authoritative dispatch path.
3	**Verifier loop unwired**	Verifier / QA	2.2 §2 verdict ABSENT, 2.1 Dead Edge 3, 3.1 D1	8	8	M	**32**	Add Section 2b to /task-postflight SKILL.md: conditional dispatch of /verify-fix-loop for docs/system/refactor domains when Proveo PASS; no new infrastructure required.

4	**mem0 SoR wire break**	Memory plane	1.1 §4, 2.1 §B Dead Edge 24/25	6	7	M	**21**	Designate .md files as official SoR or wire a PostToolUse hook that calls `POST localhost:9000/add` on every memory .md write; choose one, document it, retire the other.
5	**Agent routing table incomplete**	Agent fleet	1.3 §A concerns A/B, 2.1 §C	7	8	M	**28**	Add validator, distiller, mehanik, evidence-verifier, dzevad-jahic, fix-builder to specialist-mapping.json; sync 8 definitions-only agents to ~/.claude/agents/.

6	**5 deleted scripts with live plists**	Daemon fleet	1.4 §2 exit 127 analysis	5	7	S	**35**	Unload plists for pi-orch-health, cost-daily-report, daily-planning, legal-docs-azure-sync, mcp-health-check; restore scripts or remove LaunchAgents permanently; stop infinite crash loops.
7	**4 phantom companies unroutable**	Agent fleet / Routing	1.3 §2, 2.1 §C	5	6	M	**15**	Add Axiom, Datavera, Resolver, Lexicon to specialist-mapping.json with at least one dispatch agent each; or officially mark them as experimental and document the direct-session access pattern.

8	**Blueprint score gate advisory-only**	BUILD-BLUEPRINT discipline	2.3 §2 issues A/B/C	6	5	S	**30**	Lower enforced threshold to 60 (matching observed practice floor) or escalate WARN to BLOCK in `pre-dispatch-gate.sh`; fix missing-MC-ID bypass path.
9	**Chroma and stale mem0 orphan stores**	Memory plane	1.1 §3, 3.1 A4	3	5	S	**15**	Audit Chroma origin; if no active reader/writer, delete. Archive or document stale mem0_john/knowledge collections. Reduces cognitive confusion and false recovery paths.
10	**B2 storage cap exceeded**	Daemon fleet / Backup	1.4 §3 backup layer, 2.1 Edge 38	4	7	S	**28**	Raise Backblaze B2 bucket cap in the console (billing action); verify litestream replication is picking up where nightly snapshots fail.

Section 4 — Architectural Conclusions

The fragmented memory plane

The architecture planned for mem0 as the System of Record for John's personal facts, with LightRAG as the document retrieval layer. What exists is five parallel stores — mem0/Qdrant (93K+ vectors, zero active writers), LightRAG (999 docs, healthy), HiveDB SQLite (17K rows, healthy), Chroma (6.5K embeddings, unknown origin, no active reader), and 123 .md files (the actual write target of Claude Code's native auto-memory). Each store evolved independently. The .md files won the write race by default — Claude Code writes them natively without any configuration. The lightrag-auto-ingest.sh hook then routes .md writes to LightRAG, making .md→LightRAG the de-facto pipeline. mem0 accumulated 865 facts in its setup phase and has received nothing since. Nobody documented this inversion as a decision. The result is a system where the architecture document says one thing, the code does another, and the divergence is invisible until an audit reveals it. There is no reconciliation daemon, no SoR designation in any machine-readable config, and no alert when the stores diverge. This is not a failure of implementation — it is a failure of architectural governance. The fix is to pick a winner, write it down, and wire everything else as a derivative.

Capability without auto-invocation

Three significant capabilities were built, tested, and deployed — and then left sitting idle because the trigger that would activate them was never wired. The verify-fix-loop skill is fully specified: it decomposes acceptance criteria into atomic claims, dispatches a verifier agent, optionally dispatches a fix-builder, loops up to three times, and escalates cleanly. It has a cost cap. It handles domain escalation policy. It works when a human types a trigger phrase. It has never been activated automatically. The same pattern holds for mem0 — the server is running, the Qdrant collections are populated, the API surface is correct, but no hook or daemon calls the write endpoint. The library skill is functional as a CLI but there is no daemon that proactively loads relevant cookbooks before task dispatch. This is an engineering pattern I recognize from large enterprise projects: the team builds the component, writes the spec, declares it done, and moves to the next feature. Integration — the wiring between components — is treated as an afterthought. In a distributed system, integration is the product. A verifier that nobody calls is not a verifier. It is documentation.

The phantom infrastructure pattern

The audit found four virtual companies (Axiom, Datavera, Resolver, Lexicon) with complete organizational infrastructure: persona directories, CLAUDE.md files, company.json, README, 5–9 internal agents each. None appear in specialist-mapping.json. There is no routing path from John's

normal dispatch flow to any of them. Similarly, 35 chain YAML files define multi-step agent pipelines — and chain-runner.js (/system/tools/chain-runner.js, MC #1902) and chain-runner.sh (/system/tools/chain-runner.sh, Pillar #5) both exist as chain executors. However, (a) no active skill invokes them (skills call agents inline), (b) the three chain-related daemons that call chain-runner.sh all exit 1 due to downstream failures, and (c) chain-runner.js has no active caller in the current daemon or skill fleet. The chain YAML files are not dead because no executor exists — they are dead because the executors are broken or un-invoked. Five LaunchAgent plists reference scripts that were deleted at some point, leaving the daemons in permanent exit-127 loops. Two of them have KeepAlive.Crashed=true, meaning launchd restarts them on every crash, generating hundreds of failed process spawns per day. Phantom infrastructure has a cost: it consumes cognitive space during troubleshooting, generates false signals in health dashboards, and creates the illusion of capability that does not exist. The four phantom companies are particularly expensive because they imply John has routing coverage he does not have — if a task arrives that maps to Lexicon or Resolver capability, the system will not tell John it cannot route it. It will silently fall through.

The dual-process dispatch pattern

pi-orchestrator (PID 75750) is running. Its HTTP port 8401 refuses connections. The durable-runner bridge (port 3052) has been up for 20 days. These are two separate processes serving what should be one control plane. The kernel's own HTTP endpoint appears to have failed silently at some point, and the bridge was deployed as a workaround. No dispatch logs exist after 2026-03-19, which means either the system has not dispatched a task automatically in 50 days, or it is dispatching via a path not captured in the logs. The pi-orch-health script that would tell us was deleted on 2026-05-06 — the monitoring for the orchestrator is gone precisely when we need it most. The last recorded verdict from that monitor was CRITICAL. This dual-process split is not an architecture — it is an accident that has calcified into the operating model.

What the audit reveals about John as AI Director

John's CLAUDE.md presents a picture of a system where John delegates, monitors, and reports — while automation handles dispatch, verification, and completion. The audit reveals the actual operating model: John manually dispatches every specialist agent in the current conversation, manually verifies outputs (or asks the CEO to), and manually calls mc.js done. The automation layer exists as infrastructure but not as function. The 113 Mehanik cleared tokens in /tmp confirm John is disciplined about gate ceremonies — the ritual is present. But the outcome of those ceremonies (automated specialist dispatch via pi-orchestrator) is absent. What John actually does is closer to a senior engineer in a terminal window than an AI Director in an automated factory. This is not a criticism — it is a structural observation. The gap between the documented role and the operational reality is the gap between an architecture diagram and a working system. Closing that gap requires exactly three things: pi-orchestrator dispatch actually working, verify-fix-loop auto-invoked at task completion, and a clear SoR for memory. Everything else is incremental improvement. These three are the load-bearing walls.

Section 5 — Output for Downstream

5.1 Hand-off to devils-advocate (Phase 4.2)

The following gaps are strong findings in the audit but carry assumptions that need rebuttal-challenge before being formally confirmed in the fix backlog:

Gap	Rebuttal challenge needed
pi-orchestrator not dispatching	P3.1 (3.1 C2) found no mock config reference in the actual js file; config shows `offlineMode: false`. Is the lack of dispatch logs after 2026-03-19 because (a) dispatch actually stopped, (b) logs are written elsewhere, or (c) durable-runner is dispatching and pi-orch kernel is a passive watcher? The distinction matters for the fix: if dispatch moved entirely to durable-runner, "fix pi-orch" may be the wrong target.
mem0 as SoR — is it intentional?	The .md-first approach may be deliberate architecture, not drift. Claude Code's native auto-memory is a designed feature. The question is whether the team consciously decided "use .md + LightRAG as SoR, deprecate mem0" or whether mem0 was forgotten. If the former, Gap #4 is not a gap but a completed migration that was never documented.
35 dead chains	Claim: all 35 chains are dead because no executor exists. Rebuttal: skills call agents inline — is this equivalent to executing a one-step chain? The chains may represent a future DAG execution model that was prototyped and deferred, not a failed deployment. If deferred intentionally, the gap is documentation, not a broken executor.
4 phantom companies	Do Axiom, Datavera, Resolver have any work product? If they have been used via direct session invocation and are producing value, they are not phantom — they are informal. The rebuttal challenge: enumerate at least one real task that was dispatched to each company and assess whether the informal routing actually works.
verify-fix-loop wiring	P2.2 establishes that shell hooks cannot spawn conversational agents (architectural constraint). Before confirming the fix as "add to /task-postflight", validate that Task dispatch from within a skill conversation context actually works reliably for sub-agent spawning, or whether the pi-orch trigger-file pattern is required.

5.2 Fix backlog skeleton (Phase 4.3 — MC stubs, audit-level only)

These are audit-derived fix proposals. No MCs are created here — these are stubs for Phase 4.3 to evaluate, scope, and assign.

Stub ID	Title	Target system	Priority	Effort	Dependencies
FIX-01	Restore RAG drain-worker: fix Vaultwarden session + CF Access credentials	Daemon fleet / RAG pipeline	H	S	Vaultwarden accessible
FIX-02	Diagnose pi-orchestrator HTTP port 8401 + restore real dispatch	Orchestration kernel	H	L	FIX-01 (credential pattern same)
FIX-03	Wire verify-fix-loop into /task-postflight Section 2b	Verifier / QA	H	M	FIX-02 ideally (or manual trigger as interim)
FIX-04	Designate SoR for memory plane; document the .md→LightRAG pipeline as canonical or wire mem0	Memory plane	H	M	None
FIX-05	Sync 8 definitions-only agents to ~/.claude/agents/; add validator/distiller/mehanik to specialist-mapping.json	Agent fleet	M	S	None
FIX-06	Unload 5 dead-script plists; restore or archive cost-daily-report.sh and pi-orch-health.sh	Daemon fleet	M	S	None
FIX-07	Enforce blueprint score gate at threshold 60 (not advisory 90); fix missing-MC-ID bypass	BUILD-BLUEPRINT	M	S	None

Stub ID	Title	Target system	Priority	Effort	Dependencies
FIX-08	Register 4 phantom companies in specialist-mapping.json or formally mark as experimental	Agent fleet	M	M	FIX-05
FIX-09	Delete or document Chroma orphan; archive stale mem0_john/knowledge collections	Memory plane	L	S	FIX-04
FIX-10	Raise B2 storage cap in Backblaze console + verify litestream live replication	Backup / Infra	M	S	None (billing action)
FIX-11	Schedule agent-definitions-sync.sh as daily cron to prevent dual-store drift	Agent fleet	L	S	None
FIX-12	Add blueprint staleness alert daemon: if modified > 30d and repo commits > 14d, surface warning	BUILD-BLUEPRINT	L	S	None

---

Report produced by Petter Graff — CodeCraft Lead Architect Source reports: 1.1 (chip-huyen), 1.2 (sentinel-developer), 1.3 (sentinel-architect), 1.4 (kelsey-hightower), 2.1 (sentinel-architect synthesis), 2.2 (martin-kleppmann), 2.3 (sentinel-ba), 3.1 (sentinel-tester) P3.1 live-probe data used as authoritative override for contradicted P1/P2 claims.

Revision #2
Created 2026-05-09 19:44:23 UTC by John
Updated 2026-06-14 20:02:59 UTC by John