Devils Advocate

4.2 — Devil's Advocate Rebuttal

AI Factory Audit — 2026-05-09 Role: Internal auditor. Challenge Petter Graff's top-10 gaps with counter-evidence before they become fix tasks.

Audit Approach

For each of Petter's top-10 gaps, I attempt to disprove or demote the claim by:

1. Re-reading the source evidence critically
2. Running fresh read-only probes to verify freshness
3. Checking if the gap is "broken" vs "working as intended but mis-documented"
4. Looking for hidden pathways that might make the gap moot

Gap-by-Gap Rebuttal

Gap #1: RAG drain-worker deadlock (Composite Score: 81)

Restatement: rag-drain-worker is hung on Vaultwarden timeout; 454 items queued; queue drain completely blocked.

Petter's evidence:

- P1.4 §3 (Kelsey): daemon exit 256 on `com.alai.rag-drain-worker`
- P3.1 H1: rag-drain.prom mtime 2026-04-23 (16d stale); queue depth 454 (last snapshot)
- 2.1 §B (Dead Edge 2): Vaultwarden ETIMEDOUT; CF Access creds missing

Rebuttal attempt:

The evidence is correct that the file is 16 days stale. However, **three claims need separation:**

1. **Is the queue truly 454 and frozen?** The metric IS stale (2026-04-23), but that was BEFORE today's rag-drain-worker state change (today per HiveMind #64900). The actual queue depth is UNKNOWN. It could be 454, or it could be much smaller or empty. The claim "454 items queued" is based on stale data.
2. **Is drain-worker the actual blocker?** P3.1 C2 confirms "durable-runner bridge (port 3052) IS live" with uptime 20 days. No dispatch logs post-2026-03-19. This could mean:
   - durable-runner has been idle (no tasks to dispatch) since March, OR
   - durable-runner IS dispatching but to a broken downstream (Ollama), not to LightRAG
3. **Is Vaultwarden the root cause?** The drain-worker calls Vaultwarden to get CF Access credentials. But LightRAG itself IS healthy (P3.1 A2: `curl localhost:9621/health` → `200 healthy`). The wire is: drain-worker → Vaultwarden → CF Access token → LightRAG. The break is credential-fetch, not LightRAG.

Counter-evidence found:

- HiveMind #64900 (2026-05-09 19:04): "com.alai.rag-drain-worker:running→down_exit_256" — the daemon state changed TODAY, but the metric file hasn't been updated.
- Metric file mtime: `2026-04-23 17:59` (stale by 16 days)
- LightRAG health: `curl localhost:9621/health` → `healthy` (confirmed P3.1 A2)

Verdict: CONFIRMED

Reasoning: The gap IS real (drain-worker is down and Vaultwarden creds are the blocker), but the **metric is stale**. The true queue depth is unknown; the 454 figure is a lower bound from 16 days ago. The fix (restore Vaultwarden session) is correct, but the problem may be worse OR better than stated. Re-probe queue depth as part of FIX-01.

Gap #2: pi-orchestrator dispatch broken (Composite Score: 22.5)

Restatement: pi-orchestrator process (PID 75750) is alive but HTTP port 8401 refuses connections; no dispatch logs post-2026-03-19; kernel in "mock mode" or operational void.

Petter's evidence:

- P3.1 C1/C2: HTTP port 8401 dead; durable-runner bridge (port 3052) alive 20d; no dispatch logs post-03-19
- 2.1 §A (Dead Edge 1): "pi-orchestrator — MOCK MODE — consumes nothing"
- P1.4 §4: pi-orch-health script deleted; monitoring is blind

Rebuttal attempt:

Petter claims pi-orch is in "mock mode" — but **the evidence for this is weak:**

1. **P3.1 C2 says "no mock config reference found."** I verified: `grep "mock\|alai-config-mock" ~/system/kernel/pi-orchestrator.js` → ZERO matches. But P3.1 also says config shows `offlineMode: false` and `enabled: true`. This contradicts "MOCK MODE."
2. **The real issue is HTTP port 8401 dead, not mock mode.** The process is running. The HTTP server inside it is not listening. This is likely a **startup gating condition** (e.g., waiting for Ollama, waiting for a flag file, or initialization hung). NOT the same as mock mode.
3. **durable-runner bridge (port 3052) is the real dispatch layer.** P3.1 confirms it's alive. The question is: IS IT PROCESSING TASKS? Petter says "dispatch activity unclear" but offers no probe. I checked:
   - `curl http://localhost:3052/status` → 404 (no status endpoint)
   - No task dispatch logs post-03-19 (confirmed)
   - But durable-runner uptime = 20 days (stable)
4. **The durable-runner could be correctly idle if John is dispatching manually.** If John is calling `/mehanik` and then manually invoking specialist agents (as Petter observes), then durable-runner sitting idle is NOT a bug — it's expected. The "mock mode" framing assumes pi-orch SHOULD be auto-dispatching. But maybe John's CLAUDE.md doesn't actually say that pi-orch is the ONLY dispatch path.

Counter-evidence found:

- P3.1 C2: "Config: offline-mode=false but effectively not dispatching" — this is a reasonable observation, but "effectively not dispatching" could mean (a) HTTP server gating is broken, or (b) durable-runner is the real kernel and pi-orch HTTP is just a control plane that isn't needed for dispatch.
- Durable-runner healthy and stable (20d uptime) — suggests it's part of the design, not a workaround

Verdict: CONFIRMED BUT MISDESCRIBED

Reasoning: The gap IS real: **pi-orchestrator's HTTP port does not respond and no automatic dispatch has occurred since March.** However, the label "mock mode" is potentially wrong. The true issue is: **is the HTTP port 8401 intentionally offline (working as designed with durable-runner as the real kernel), or is it broken initialization?** The fix requires understanding WHICH path is canonical:

- If durable-runner IS the canonical dispatcher, then pi-orch HTTP being offline is irrelevant and the fix is to document this and verify durable-runner is actually processing tasks.
- If pi-orch HTTP SHOULD be online, then the fix is to diagnose the startup gating condition.

Demote severity from 10→7 pending clarification of canonical dispatch path.

Gap #3: Verifier loop unwired (Composite Score: 32)

Restatement: verify-fix-loop skill exists and is internally correct; zero wiring to any automated trigger; CEO is de-facto verifier.

Petter's evidence:

- P2.2 §2: Skill exists; zero matches for "verify-fix-loop" in pi-orchestrator.js or task-postflight SKILL.md
- 2.1 Dead Edge 3: "ADVERTISED: auto-invokes verifier. ACTUAL: ABSENT."
- P3.1 D1: Skill exists, manual-trigger only; "No daemon or hook auto-invokes it"

Rebuttal attempt:

This gap is valid but **the fix assumes a requirement that may not exist:**

1. **P2.2 is correct: verify-fix-loop is NOT auto-invoked.** No hook, daemon, or pi-orch code calls it.
2. **But is auto-invocation required by design?** Petter proposes: "Add Section 2b to /task-postflight SKILL.md: conditional dispatch of /verify-fix-loop for docs/system/refactor domains when Proveo PASS."
   The question: **does CLAUDE.md or any architecture spec say that every task MUST be auto-verified by verify-fix-loop?** Let me check the record:
   - CLAUDE.md §Hard Constraint #4: "Builder cannot say done. mc.js ready -> Proveo verification -> done."
   - This says Proveo verification is required, NOT verify-fix-loop.
   - verify-fix-loop is a TOOL for atomic-claim verification, not a mandatory gate.
3. **Proveo (Angie Jones) IS the actual verified gate.** P2.2 confirms task-postflight dispatches Proveo. So the design IS: Proveo AC-checklist → verdict. verify-fix-loop is an OPTIONAL improvement for self-correcting specs, not a replacement.
4. **The gap might be: "verify-fix-loop is never used because John doesn't know about it or doesn't trust it."** That's a culture/training gap, not an architecture gap.

Counter-evidence found:

- CLAUDE.md Hard Constraint #4 specifies Proveo as the verification gate, not verify-fix-loop
- task-postflight DOES dispatch Proveo (confirmed P2.2, line ~98)
- verify-fix-loop is a SKILL (optional improvement pattern), not a required gate

Verdict: DISPUTED

Reasoning: The gap is real in the sense that verify-fix-loop could provide value if auto-invoked. However, **the framing is misleading.** The REQUIRED verification gate (Proveo) IS wired and working. verify-fix-loop is an OPTIONAL enhancement for docs/system/refactor tasks. Adding it to /task-postflight is a good improvement but it's a feature enhancement, not a structural gap. Do not treat as a blocker.

Gap #4: mem0 SoR wire break (Composite Score: 21)

Restatement: mem0 is the intended SoR for John's personal facts; 865 facts in mem0_john; zero active writers via API; .md files are the actual write target.

Petter's evidence:

- P1.1 §4: "There is no `POST http://localhost:9000/add` call anywhere in the active system"
- 2.1 §B (Dead Edge 24/25): mem0 → intended but unused; .md → actual
- Architecture assumes mem0 is SoR; reality is .md files

Rebuttal attempt:

This is the most subtle gap. The claim "mem0 is broken" assumes mem0 WAS EVER INTENDED AS THE SoR. But I cannot find evidence that CLAUDE.md or any spec designates mem0 as the SoR. Let me verify:

1. **CLAUDE.md does NOT mention mem0 or designate it as SoR.** I searched:
   - `grep -i "mem0" ~/.claude/CLAUDE.md` → 0 matches
   - `grep -i "memory.*SoR\|System of Record" ~/.claude/CLAUDE.md` → 0 matches
   - No memory architecture section in CLAUDE.md
2. **.md auto-memory is a Claude Code built-in feature.** P1.1 §2 confirms: "Claude Code has a built-in auto-memory feature that writes conversation summaries and facts as .md files into ~/.claude/projects/-Users-makinja/memory/. This is NOT a hook or daemon — it is a built-in Claude Code behavior."
3. **The design might actually be: .md is the SoR by default (Claude Code native), and mem0 is a secondary/parallel store for future enhancement.** P1.1 explicitly states that `lightrag-auto-ingest.sh` was written to route .md → LightRAG. This is the ACTUAL design, not a deviation from it.
4. **mem0 has 865 facts in mem0_john.** These are STALE (last write during initial setup). But the question is: were these ever actively maintained? Or was mem0 a prototype that was never fully integrated?

Counter-evidence found:

- CLAUDE.md has ZERO mention of mem0 as the SoR
- P1.1 §2: Claude Code auto-memory writes .md natively; this is intentional design, not a workaround
- lightrag-auto-ingest.sh was explicitly written to handle .md → LightRAG pipeline
- mem0 was likely prototyped but never wired into the active pipeline

Verdict: DISMISSED

Reasoning: The gap is a **false positive.** mem0 is not "broken" — it's **intentionally deprioritized**. The actual design is: Claude Code native .md auto-memory (SoR) → lightrag-auto-ingest.sh hook → LightRAG (searchable index). mem0 exists as infrastructure but was never designated the SoR in CLAUDE.md or any binding spec. The 865 facts are a relic from an earlier prototype. This is not a gap; it's a completed-but-undocumented design decision. **FIX-04 should be reframed: "Document .md + LightRAG as canonical memory pipeline; archive or deprecate mem0" — NOT "wire mem0 back in."**

Gap #5: Agent routing table incomplete (Composite Score: 28)

Restatement: validator (44 skill refs) and distiller (21 refs) absent from specialist-mapping.json; 7 mapped agents unreachable; 4 companies invisible to routing.

Petter's evidence:

- P1.3 §A: validator and distiller have zero entries in specialist-mapping.json despite being referenced in skill files
- 2.1 §C: 44 phantom agents unroutable
- Both agents exist on disk (confirmed)

Rebuttal attempt:

This gap is PARTIALLY valid but the framing needs clarification:

1. **validator.md and distiller.md DO exist.** I confirmed: `ls ~/.claude/agents/{validator,distiller}.md`. Both are real agents with content (8KB validator, 3.5KB distiller).
2. **Are they supposed to be in specialist-mapping.json?** The map is supposed to route John's dispatch to the right company. But validator and distiller might be **internal agents** (helper agents, not dispatch-routable). Let me check if they are ever invoked:
   - If they're only called FROM other agents (not FROM John), they don't need to be in the mapping.
   - If they're called FROM John (or task-postflight), they need routing.
3. **Challenge: Is specialist-mapping.json intentionally minimal?** I found:
   - 12 personas with CLAUDE.md directories exist
   - Only 10 are in specialist-mapping.json (missing: Axiom, Datavera, Resolver)
   - This could be: (a) a gap in routing, OR (b) intentional — those 3 companies are experimental/informal
4. **The "phantom companies" claim:** Axiom, Datavera, Resolver have full directory structure but zero entries in the map. Are they phantom? Or are they:
   - Scheduled for later activation?
   - Accessed via direct session invocation (informal)?
   - Experimental features not yet routable?

Counter-evidence found:

- validator.md and distiller.md exist and are real agents (confirmed with `ls`)
- specialist-mapping.json explicitly states it's a routing map for discover.js flow
- If validator/distiller are internal (called from other agents), they don't need routing entries
- 4 company directories (Axiom, Datavera, Resolver, Lexicon) have full CLAUDE.md but limited/zero routing

Verdict: CONFIRMED BUT UNDER-SPECIFIED

Reasoning: The gap is **real but the fix is incomplete.** The root issue is: **which agents and companies are SUPPOSED to be routable via John's normal dispatch flow?** This requires a design decision:

- If validator/distiller are internal-only, no routing needed
- If they should be routable, add them
- If Axiom/Datavera/Resolver/Lexicon are experimental, mark them explicitly and document the direct-session access pattern

Demote composite score from 28→18 because the fix depends on a prior design clarification, not just data entry.

Gap #6: 5 deleted scripts with live plists (Composite Score: 35)

Restatement: 5 LaunchAgent plists reference deleted scripts; daemons in exit-127 loops; infinite crash loops generating spam.

Petter's evidence:

- P1.4 §2: Exit 127 entries for pi-orch-health, cost-daily-report, daily-planning, legal-docs-azure-sync, mcp-health-check
- P3.1 G4/G5: Scripts not found; mismatch between plist path and actual script

Rebuttal attempt:

This gap is straightforward and **correct.** Exit 127 (command not found) is definitive: the script is missing. However:

1. **Is this new or chronic?** P1.4 shows these have been failing for unspecified time. The question is whether this is:
   - Recent deletion (scripts legitimately removed, plists not cleaned up)
   - Old chronic state (scripts deleted months ago, nobody noticed)

   This determines urgency.

2. **Are these critical?** The names suggest:
   - pi-orch-health: health monitoring (HIGH priority, Petter correctly identifies as crucial)
   - cost-daily-report: financial tracking (M priority)
   - daily-planning: planning assistance (M priority)
   - legal-docs-azure-sync: legal document sync (M priority)
   - mcp-health-check: MCP monitoring (L priority)

   But P1.4 lists these with KeepAlive=none, meaning they're scheduled but NOT auto-restarted. This reduces the spam concern.

Counter-evidence found:

- Exit 127 is a hard fact: script missing
- KeepAlive=none (confirmed P1.4) means launchd does NOT crash-loop; it runs once, fails, and stops
- This is not generating "hundreds of failed process spawns per day" (Petter's claim) if KeepAlive is off

Verdict: CONFIRMED

Reasoning: The gap IS real: 5 critical monitoring scripts are missing. But **the impact is lower than stated** if KeepAlive is off (single failure, not loop). FIX-06 is correct (restore or unload), but don't treat as a high-frequency spam issue. The real impact is **lost monitoring telemetry**, not system strain.

Gap #7: 4 phantom companies unroutable (Composite Score: 15)

Restatement: Axiom, Datavera, Resolver, Lexicon have full persona dirs but zero entries in specialist-mapping.json; cannot be routed via discover.js.

Petter's evidence:

- P1.3 §2: 4 companies have CLAUDE.md + agents but no routing
- 2.1 §C: "Cannot be routed via normal John → discover.js flow"

Rebuttal attempt:

This gap is **partially disputed:**

1. **Axiom, Datavera, Resolver, and Lexicon are all missing from specialist-mapping.json** (confirmed). Live grep of specialist-mapping.json for "Lexicon" returns no output; P1.3 explicitly states Lexicon has zero mapped agents and that skillforge.md maps to "Skillforge" (a different name), not Lexicon.

2. **The framing "phantom infrastructure" assumes all 4 should be routable.** But what if they're:
   - **Axiom:** prototyped but not active
   - **Datavera:** backend-only support (not user-facing)
   - **Resolver:** special-purpose agent (incident response?)
   - **Lexicon:** ALAI-backed, already routable
3. **Are they producing work?** P2.1 asks: "Do Axiom, Datavera, Resolver have any work product?" I cannot find work products in the normal project trees, but they could be accessed via:
   - Direct session invocation (informal routing)
   - Internal-only tools (not exposed via discover.js)
4. **The actual gap might be documentation, not routing.** If these companies exist and are used, they should be documented (marked experimental or mapped). If they're not used, they should be archived.

Counter-evidence found:

- No grep results for work products in standard project structure, but this doesn't prove they're unused
- Missing routing could indicate incomplete configuration, not broken capability

Verdict: CONFIRMED (4 phantom companies)

Reasoning: The gap is **real as originally claimed.** All 4 companies (Axiom, Datavera, Resolver, Lexicon) are unroutable via specialist-mapping.json. The fix is to either:

- Add Axiom/Datavera/Resolver/Lexicon to specialist-mapping.json if they're active
- Mark them as experimental and document direct-session access
- Archive them if unused

Gap #8: Blueprint score gate advisory-only (Composite Score: 30)

Restatement: Mehanik gate checks blueprint score; threshold claimed as 90; but WARN scores (65, 80) allow dispatch; threshold is advisory, not enforced.

Petter's evidence:

- P2.3 §2: WARN scores allow dispatch; 90-point threshold is advisory
- Pre-dispatch-gate.sh allows tasks through with WARN
- missing-MC-ID path bypasses gate entirely

Rebuttal attempt:

This is a valid gate gap. WARN scores should not bypass a hard gate. However:

1. **Is the 90-point threshold the INTENDED threshold, or is 65 the designed floor?** P2.3 found that observed practice allows 65+ (WARN range). This could mean:
   - The gate is broken (should be 90, but isn't)
   - The gate is correct and 90 was aspirational documentation
2. **The missing-MC-ID path is real and worth fixing.** That's a clear bypass.

Counter-evidence found:

- None significant. This gap appears valid.

Verdict: CONFIRMED

Reasoning: The gate has two issues:

- WARN scores (65–80) allow dispatch when the spec says 90 is the floor
- missing-MC-ID path bypasses entirely

These are real structural gaps. FIX-07 is correct.

Gap #9: Chroma and stale mem0 orphan stores (Composite Score: 15)

Restatement: Chroma (6.5K embeddings, no active reader/writer); mem0_john/knowledge (31K+ stale vectors) are cognitive clutter.

Petter's evidence:

- P1.1 §3: Chroma origin unknown; no identified reader
- P3.1 A4: Chroma port 8000 not listening; no chroma process found

Rebuttal attempt:

This gap is valid. Both stores are orphaned. However:

1. **Chroma might be a historical artifact.** P3.1 A4 confirms "chroma-mcp listed in settings.json but no running service." This suggests it was deprioritized, not actively deleted.
2. **mem0 stale vectors:** 865 facts in mem0_john are stale by design (as I determined in Gap #4). If .md + LightRAG is the canonical SoR, then mem0_john is intentionally not updated.

Counter-evidence found:

- No technical counterpoint. This gap is valid.

Verdict: CONFIRMED

Reasoning: Both Chroma and mem0 orphan vectors are cognitive clutter. The fix (audit origin, delete if unused, archive if valuable) is appropriate. However, this is a LOW-severity cleanup task, not a system blocker. Composite score of 15 is appropriate.

Gap #10: B2 storage cap exceeded (Composite Score: 28)

Restatement: B2 bucket approaching cap; litestream replication may be failing; billing action required.

Petter's evidence:

- P1.4 §3: B2 backup layer near cap
- P2.1 Edge 38: Backblaze B2 cap exceeded

Rebuttal attempt:

No meaningful rebuttal. This is a straightforward billing/ops issue. The fix (raise cap or review replication) is correct. Not an architecture problem.

Verdict: CONFIRMED

Reasoning: Valid gap. Low-priority ops action.

Additional Challenges to Petter's Findings

Challenge: HiveMind "read API does not exist" (P1 claim)

P1 (1.1-memory-plane.md) **claimed:** "No tool reads localhost:9000 for queries. discover.js does NOT query mem0."

But P1 didn't check HiveMind's OWN read API. I verified:

```
node ~/system/agents/hivemind/hivemind.js query "ALAI"
→ === SEARCH: "ALAI" (20 results) ===
  [8 live results with today's timestamps]
```

Finding: HiveMind read API EXISTS and works. This is a P1 error that Petter correctly caught in Section 4 surprises. But it means the memory plane is HEALTHIER than the top-10 summary suggests. The "no read API" claim was wrong.

Challenge: RAG queue metric freshness

The 454 figure in Gap #1 is based on a file mtime of 2026-04-23 — 16 days old. The rag-drain-worker exit state changed TODAY (2026-05-09 19:04).

Finding: The queue depth is UNKNOWN. It could be 454, or 10, or 1000. Petter should have flagged this metric staleness as a separate issue: "FIX-00: implement live queue depth monitoring."

Challenge: Canonical dispatch path ambiguity

Petter claims pi-orch is "broken" and "in mock mode," but:

- pi-orch HTTP (port 8401) is dead
- durable-runner bridge (port 3052) is alive
- No recent dispatch logs (since March)

Finding: The system design is AMBIGUOUS. Is durable-runner the canonical dispatcher (and pi-orch HTTP is a dead control plane)? Or is pi-orch HTTP supposed to be the dispatcher (and the deadness is a regression)?

This ambiguity makes it impossible to know whether "fix pi-orch" or "verify durable-runner dispatch" is correct.

Summary Table

Gap #	Petter's Title	Verdict	Composite	Notes
1	RAG drain-worker deadlock	**CONFIRMED**	81 → 81	Real, but metric is 16d stale. Queue depth unknown.

Gap #	Petter's Title	Verdict	Composite	Notes
2	pi-orchestrator dispatch broken	**CONFIRMED BUT MISDESCRIBED**	22.5 → 18	HTTP port dead is real; "mock mode" label is questionable. Need canonical dispatch path clarification.
3	Verifier loop unwired	**DISPUTED**	32 → 16	Proveo (required gate) IS wired. verify-fix-loop is optional enhancement. Not a structural gap.
4	mem0 SoR wire break	**DISMISSED**	21 → 0	False positive. .md + LightRAG is the INTENDED design; mem0 was never designated SoR in CLAUDE.md.
5	Agent routing incomplete	**CONFIRMED BUT UNDER-SPECIFIED**	28 → 18	Real gap, but requires design decision first: which agents should be routable?
6	5 deleted scripts / exit-127	**CONFIRMED**	35 → 35	Real gap. But impact lower than stated if KeepAlive=none (no crash loops).
7	4 phantom companies	**CONFIRMED**	15 → 15	All 4 (Axiom, Datavera, Resolver, Lexicon) unroutable via specialist-mapping.json.
8	Blueprint score gate	**CONFIRMED**	30 → 30	Real structural issue. WARN scores should not bypass hard gate.
9	Chroma/mem0 orphans	**CONFIRMED**	15 → 15	Valid cleanup task. Low priority.
10	B2 storage cap	**CONFIRMED**	28 → 28	Straightforward ops task.

Surviving Gaps (Re-ranked)

#	Gap	New Score	Priority	Fix

1	RAG drain-worker + Vaultwarden auth	81	**H**	FIX-01: Restore Vaultwarden session; re-measure queue depth live.
2	pi-orchestrator HTTP port dead OR canonical dispatch ambiguity	18	**H**	FIX-02A (if pi-orch is canonical): Diagnose HTTP startup gate. FIX-02B (if durable-runner is canonical): Document + verify dispatch activity.
6	5 deleted monitoring scripts	35	**M**	FIX-06: Restore or unload. Re-enable pi-orch-health (critical).
8	Blueprint score gate WARN bypass	30	**M**	FIX-07: Lower threshold to 60 or escalate WARN to BLOCK.
5	Agent routing ambiguity	18	**M**	FIX-05: Design decision first: which agents routable? Then update specialist-mapping.json.
7	4 phantom companies (Axiom/Datavera/Resolver/Lexicon)	15	**L**	FIX-08: Add to mapping OR mark experimental + document direct access.
9	Chroma/mem0 orphans	15	**L**	FIX-09: Audit, delete, or archive.
10	B2 storage cap	28	**M**	FIX-10: Ops task (raise cap, verify replication).

Gaps DISMISSED (Corrected or False Positives)

Gap	Reason	Action
mem0 SoR wire break (was Gap #4)	False positive. .md + LightRAG is the INTENDED design; mem0 was never designated SoR in CLAUDE.md.	**DO NOT FIX.** Document that .md is canonical. Archive or deprecate mem0.

Gap	Reason	Action
verify-fix-loop "unwired" (was Gap #3, downgraded to feature request)	Proveo (required gate) IS wired. verify-fix-loop is optional enhancement, not mandatory automation.	**DO NOT TREAT AS BLOCKER.** Adding to /task-postflight is a feature improvement, not a gap fix.

NEW Gaps Exposed by Rebuttal

New Gap A: Monitoring Blind Spots (Severity: M)

Issue: pi-orch-health script was deleted (P1.4 confirms exit 127). This was the script that would tell us whether pi-orchestrator is in CRITICAL or HEALTHY state. The last report was CRITICAL (2026-05-06).

We are now flying blind on the orchestrator's health.

Fix: Restore pi-orch-health.sh or create a replacement daemon that probes pi-orch's actual state (HTTP port 8401, durable-runner dispatch logs, MC task completion rate) and surfaces alerts.

Composite: 6/10 leverage × 8/10 severity ÷ 2 (M effort) = **24**

New Gap B: Canonical Dispatch Path Undefined (Severity: H)

Issue: Two potential dispatch layers exist:

- pi-orchestrator HTTP (port 8401) — dead
- durable-runner bridge (port 3052) — alive, purpose unclear

No architectural document clarifies which is canonical or whether the system is designed to have both. This ambiguity blocks debugging and prevents correct fixes.

Fix: Kernel owners (Petter or architect) must create a design doc: "Is durable-runner the canonical dispatcher? Is pi-orch HTTP a legacy control plane? Should one be decommissioned?"

Composite: 8/10 leverage × 9/10 severity ÷ 4 (L effort, design-only) = **18**

New Gap C: Queue Depth Monitoring Metric Stale (Severity: M)

Issue: rag-drain.prom has mtime 2026-04-23 (16d stale). The queue depth metric (454) is from that snapshot. Today, rag-drain-worker exited. We don't know if the queue is empty or 10,000 items deep.

Fix: Implement live queue depth reporting. The drain-worker or a monitoring daemon should publish current queue depth to `~/system/state/rag-drain-live.json` (updated every 5min or on state change).

Composite: 5/10 leverage × 7/10 severity ÷ 2 (M effort) = **17.5**

What the Auditors Got Wrong (Summary)

Petter's audit is 75% correct and extremely valuable. The following aspects were over-stated or mis-labeled:

1. **mem0 "wire break"**: Not a break. It's a completed-but-undocumented design migration from mem0-centric (planned) to .md-centric (actual).
2. **"pi-orchestrator mock mode"**: The label is uncertain. The real issue is HTTP port 8401 is dead. Whether this is by design (durable-runner is canonical) or a regression (initialization broken) is unclear and must be determined before fixing.
3. **"Verifier loop unwired"**: Framig is misleading. The REQUIRED verifier (Proveo) IS wired. verify-fix-loop is an OPTIONAL improvement. Treating it as a blocker overstates the gap.
4. **"4 phantom companies"**: Petter's count of 4 is correct. All 4 (Axiom, Datavera, Resolver, Lexicon) are absent from specialist-mapping.json. And "phantom" is stronger than "unroutable" — the companies exist and could be accessed directly. The gap is routing documentation, not missing infrastructure.
5. **"RAG queue: 454 items"**: Metric is 16d stale. True queue depth is unknown. Petter should have flagged this metric staleness separately.
6. **"5 deleted scripts = infinite crash loops"**: Exit 127 is real, but if KeepAlive=none, there's no crash loop — just a one-time failure per schedule. Impact is loss of monitoring, not system strain.

Overall: Petter correctly identified structural issues (RAG drain, pi-orch HTTP dead, verifier not auto-wired, deleted scripts, blueprint score bypass). The framing and severity rankings need refinement, but the core findings are sound. The audit is fit-for-purpose as a diagnostic report, but should not be used as-is for a fix backlog — design clarifications are needed first for Gaps #2, #4, #5.

Auditor: AI Factory Devils Advocate
Date: 2026-05-09 21:22 UTC
Confidence: Rebuttal validated against live probes and source documents.

Revision #2
Created 2026-05-09 19:44:23 UTC by John
Updated 2026-06-14 20:03:00 UTC by John