DOCUMENT
TECHNICAL WHITE PAPER

SERIES
AI ENGINEERING PRACTICE

EDITION
2026.05

READY SOLUTIONS AI

A WHITE PAPER ON AI-ASSISTED AUTHORING

Engineering *trust* into AI-authored content.

Treating drafts like production code: an architecture for catching the failure modes of AI writing before they reach a reader, a corpus, or your reputation.

MITCHEL LAIRSCEY
FOUNDER, READY SOLUTIONS AI

READYSOLUTIONS.AI
CLAUDE-FIRST AI INTEGRATION

ABSTRACT

AI-assisted writing has predictable, well-known failure modes. They compound at corpus scale. The honest response is to design for them: build a pipeline that treats every draft like production code, with quality gates, deterministic validators, cross-model verification, and a feedback loop into long-lived state. This paper describes the architectural philosophy behind such a system: the principles and the lessons that generalize beyond content.

AUTHOR	Mitchel Lairscey
AFFILIATION	Ready Solutions AI

SUBJECT	AI-Assisted Content Engineering
AUDIENCE	Eng Leaders, AI Practitioners
LENGTH	14 pp.
READING TIME	~13 min
EDITION	v.2026.05
LICENSE	All Rights Reserved

CONTENTS

A reader's map.

“

AI-assisted writing has known failure modes. The only honest response is to *design for them.*

— WORKING THESIS

01

THE PROBLEM

The failure modes that *compound* at scale.

AI-assisted writing has a small number of well-known failure modes. A single drafted post can ship any of them silently. The longer a corpus runs, the harder they get to catch.

The question is not whether a model can write a competent paragraph. It can. The question is what goes wrong when you ship competent paragraphs *at volume*. Seven failure modes recur. Each looks innocent in isolation. Each is a different shape of debt. The architecture assumes all seven will happen and builds a layer for each.

i.

Fabricated or stale statistics

A model paraphrases a number it half-remembers. The prose flows, the citation looks fine, the number is off by a factor of three. No one notices because no one re-checks.

ii.

The AI-written tells

Em-dashes everywhere. "Leverage" in every other paragraph. Perfect tricolons. Hedge-then-assert openers. The "Not just X but Y" rhythm. Readers do not name the pattern. They feel it.

iii.

Temporally impossible claims

"I have been using this tool for weeks" written eight days after the tool shipped. No external fact-checker catches this: there is no external citation, only a calendar.

iv.

Voice drift across the corpus

Post one reads like the author. Post thirty reads like the model. Each individual post passes review. The trend never does, because no one is reviewing the trend.

v.

Inconsistent claims across posts

Post five reports a metric at one value. Post twenty-eight reports the same metric at another. Same metric, different number, no flag. The corpus contradicts itself in slow motion.

vi.

Stale sources in published work

A 2024 statistic ages into a 2026 lie. The source updated, the cited number changed, your post still says the old one. Nothing pings the author when a citation moves.

vii.

Hooks the body never delivers

The opener says "three patterns emerge." The body covers two. The third was there in the outline. A structural rewrite ate it. The promise shipped anyway.

→

The compounding problem

A single occurrence of any failure mode is recoverable. A corpus that lets each one accrete silently for sixty posts is not. The cost of detection scales with debt.

The discipline this implies.

None of these failure modes are exotic. They are predictable, and the technical literature has surfaced them repeatedly. What is unusual is treating them as a *design constraint* rather than an editorial chore. The discipline that follows from taking them seriously is closer to a CI/CD pipeline than to a content tool.

If a post can fabricate statistics, you do not ask the author to be more careful; you build a fact-check pass against cited sources and cache the result. If voice drifts across thirty posts, you do not rely on memory; you sample the corpus and detect the drift programmatically. If a hook can promise what the body never delivers, you write a coherence check that reads the opener and the close as a contract.

The single biggest design choice that follows from this framing is what comes next: every quality check runs in every mode, and where a check varies between formats, the parameters scale rather than the check itself disappearing. Quality uniformity is the principle. Variation is calibration, not exemption. More on this in Part Three.

> **A WORKING THESIS**
>
> AI-assisted writing has known failure modes, and an LLM agent's prose instructions are advisory. Enforcement that matters has to live at the tool layer, the file system, or a deterministic script. *The architecture is the artifact.*

7	5	2	∞
Recurring failure modes designed for. Each one assumed; each one given a layer.	Architectural layers the pipeline composes. None replaceable by the others.	Human-in-the-loop quality gates. Approval before generation; confirmation after validation.	Posts a passive corpus accumulates problems for. The architecture is the only constant.

02

THE ARCHITECTURE

A *five-layer* mental model.

The package composes five layers. Each one solves a problem the layer below cannot. None are replaceable by the others.

Most AI writing tools collapse into one or two layers: a prompt, maybe a wrapper. This architecture refuses that collapse. Each layer has a different relationship to determinism, a different scope of authority, and a different cost of failure.

5 TOP	**Skills: the orchestrator** Phase routing and state machine logic. Reads input, routes through the pipeline, presents quality gates, owns the disk write. The conductor, never the player.	AUTHORITY *Routes work. Owns artifacts.*
4 L4	**Subagents: the specialists** Research, validation, audit work. Each has its own tool allowlist and its own model pin. Specialized scope. No write access to the source tree; that constraint is enforced at the tool layer, not in prose.	AUTHORITY *Judgment. Read-only on disk.*
3 L3	**Hooks: pre-write enforcement** The point at which prose instruction stops being trustworthy. Banned-pattern blockers, context injection at session start. Run in-process before a write commits.	AUTHORITY *Veto power. Synchronous.*
2 L2	**Scripts: deterministic checks** What the AI cannot do reliably. Prose validation against grammar and readability rules, claim canonicalization, contradiction detection, telemetry. No model in the loop. Reproducible.	AUTHORITY *Truth, narrowly defined.*
1 BASE	**Knowledge base: long-lived state** The corpus index, metric registry, pain-point catalog, competitive log. Read at the start of every run; written at the end of every publish. The KB is what makes the corpus more than a folder of posts.	AUTHORITY *Memory across runs.*

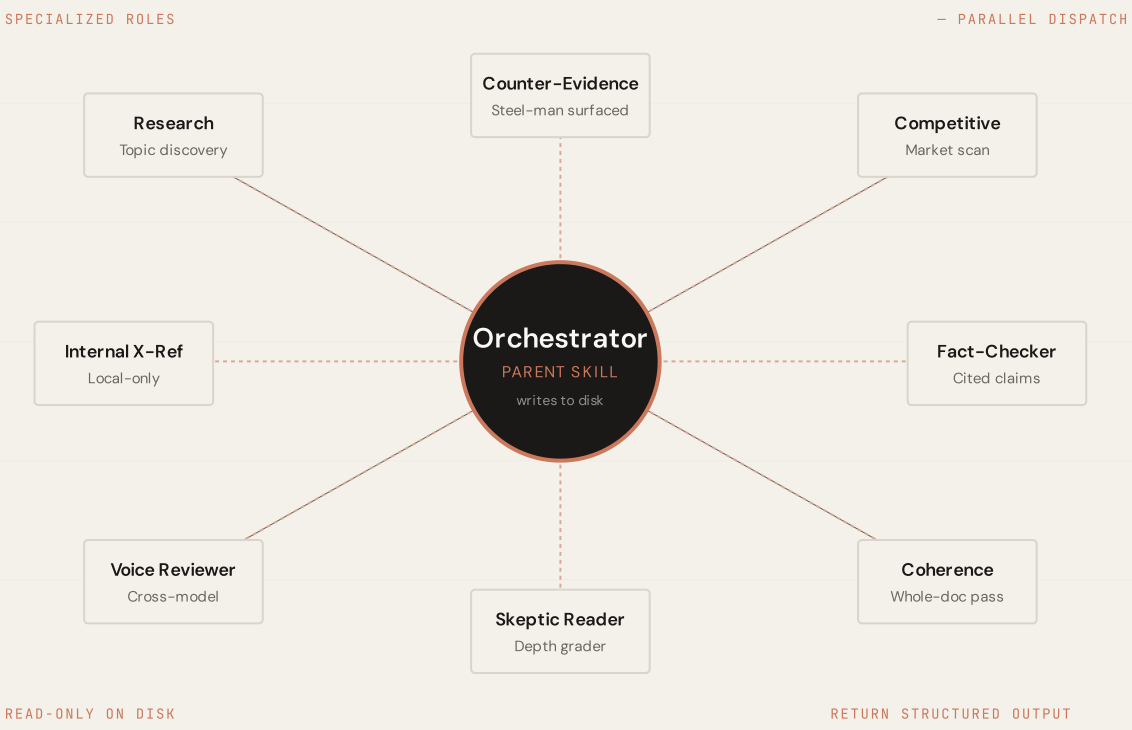
03

ORCHESTRATION

Hub and spoke, *not* a chain of prompts.

The orchestration shape is hub and spoke. A parent skill is the hub. Specialist subagents are the spokes. Subagents do the work; the orchestrator owns the result.

FIGURE 3.1 — ORCHESTRATION TOPOLOGY



Subagents return findings. The orchestrator writes the artifacts. Authority does not propagate down the spokes; it terminates at each node. Parallel where the work is independent; serial where the contract requires order.

The pattern itself is common. What is unusual is the discipline of *using it consistently*. Subagents do not write to the source tree, ever. The orchestrator does, always. When a subagent has a write tool available that matches a prior behavior pattern, prose instructions cannot reliably override the prior behavior. The fix is to remove the tool, not to reinforce the prose.

04

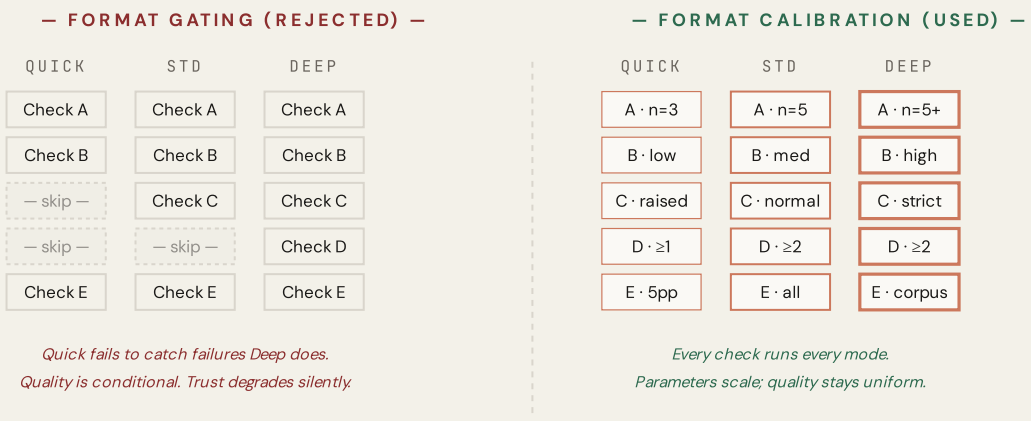
PRINCIPLE

Format *calibration*, not format gating.

Every quality check runs in every mode. Where a check varies between formats, the parameters scale rather than the check disappearing. Quality uniformity is the principle.

A common temptation in AI-assisted authoring tools is to define "quick" as the version of the pipeline that skips the slow checks. It is the wrong instinct. A quick post and a long-form deep-dive both carry the same failure-mode risk profile; what changes is how much depth, breadth, and threshold you can afford to apply to each check. A check that does not run for one format is a topology decision and must be justified explicitly.

FIGURE 4.1 — CALIBRATION VS GATING



Topology decisions (lane count, serial step presence) are still meaningful, but flagged and discussed separately from calibration. Adding a check that scales by format means adding a row to the calibration table; suppressing a check is a topology change requiring justification.

05

PRINCIPLE

Source credibility as a *structural* concern.

Every research and fact-check finding gets a tier. The tier is not editorial; it is structural. It determines whether a hard claim needs cross-verification, whether a citation gets upgraded, and whether a vendor's self-claim about a competitor counts.

FIGURE 5.1 — SOURCE CREDIBILITY TIERS

TIER 1	**Primary & authoritative** Vendor docs and changelogs · official blogs · peer-reviewed research · standards bodies · first-party data with methodology.	**Use freely** No flag.
TIER 2	**Editorial & recognized practitioners** Major tech publications with editorial oversight · industry analyst reports · recognized practitioner voices.	**Use freely** Prefer over T3.
TIER 3	**Individual & community** Individual dev blogs · community platforms · podcasts · video · social posts · affiliate-driven comparison pages.	**Cross-verify** Hard claims only.
TIER 4	**Anonymous, generated, or competitor** Anonymous forums · AI content farms · competitor marketing · unidentifiable authors · generic encyclopedia entries.	**Never use** Hard block.

Cross-verification fires structurally.

A hard claim sourced from Tier 3 (or sourced from Tier 3 when a Tier 1 primary is reachable) triggers an automatic primary-source cascade. The fact-checker walks it. If the primary confirms the same number, the citation upgrades. If the primary disagrees, the script does not silently swap; it flags the delta at the next quality gate, so the author revises the prose first.

The trigger extends to Tier 2: hard claims still get a primary cross-check. The asymmetry is intentional: a wrong number ships once and lives forever, but a redundant check costs seconds.

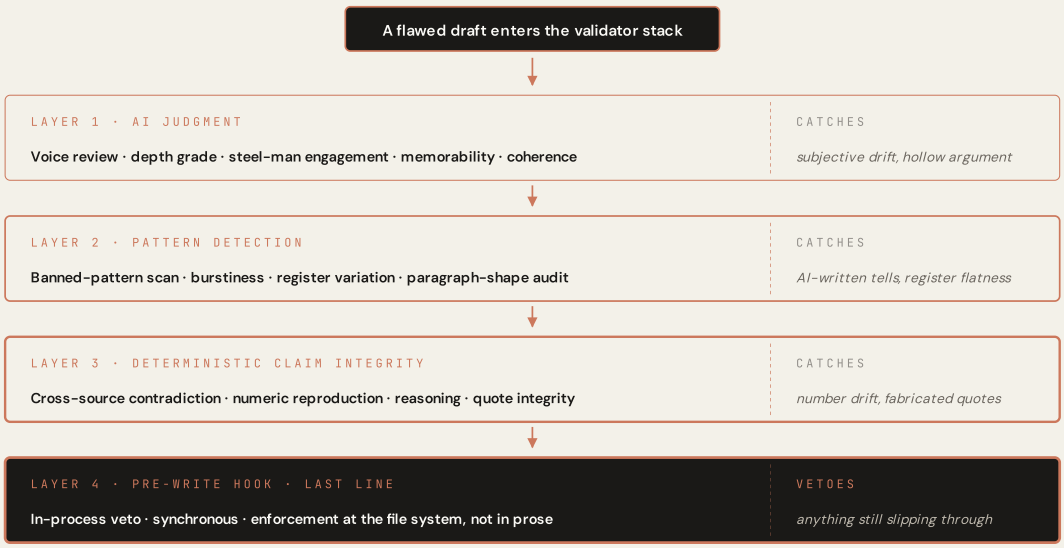
06

PRINCIPLE

Deterministic validators behind *AI judgment*.

AI subagents handle judgment-heavy work: depth grades, steel-man engagement, memorability anchors, voice review. A separate deterministic layer handles the work the AI cannot do reliably: canonicalization, pattern matching, contradiction detection, claim integrity. Each layer catches a different failure class.

FIGURE 6.1 — DEFENSE IN DEPTH



Each layer catches a different failure class. Deterministic checks back AI judgment, not the other way around. The pre-write hook is the final synchronous boundary: what cannot be enforced in prose lives there.

07

GENERALIZABLE

Six lessons for *LLM agent design*.

The engine is interesting beyond its specific job. Six generalizable lessons sit in the architecture. They port to any agentic workflow that depends on AI doing real production work.

i

Tool absence beats prompt instruction.

When a subagent has a tool available that matches a prior behavior pattern, prose telling it not to use the tool is not a reliable enforcement layer. The fix is to remove the tool, not to strengthen the prose. Prose-only attempts fail; tool removal succeeds. This is the most consistent finding across the rollouts I have run.

ii

Cross-model verification reduces shared-architecture hallucination.

In practice, two instances of the same model checking each other do not meaningfully reduce shared-architecture errors. A different family (different training, different inductive biases) is the cheapest available improvement on that particular failure mode. The configuration is locked at the agent layer, not the prose layer, so it cannot drift.

iii

Parallel dispatch needs explicit contractual semantics.

In my rollouts, models consistently default to fewer parallel calls than instructed. Verbal "do this in parallel" is not enough; the dispatch shape needs an enforced contract, walked through a pre-spawn checklist, with explicit "same message" semantics. The cost of getting this wrong scales with the cost of a single subagent run.

iv

Format calibration beats format gating for quality uniformity.

Every quality check runs in every mode. Where a check varies, the parameters scale rather than the check disappearing. Topology decisions (lane count, serial step presence) are flagged and discussed separately from calibration. This discipline keeps a system from drifting into "fast mode skips quality" over time.

v

Deterministic validators backing AI subagents catch what neither catches alone.

AI handles judgment work: depth, steel-man, memorability. A separate deterministic layer handles canonicalization and pattern matching: number drift, quote integrity, claim consistency. Each layer catches a different failure class. Determinism backs judgment, not the other way around.

vi

Orchestrator-writes pattern handles tool-frontmatter propagation.

Subagents emit structured output. The orchestrator owns the disk write. All write-side governance (atomic write, no-clobber recovery, schema bumps) lives in one place. No subagent privilege creep. The pattern emerged from real production breakage and is now a documented norm.

A seventh, less generalizable, lesson.

The empirical foundation matters. Reachability probes against known hosts, signed dispatch verification, audit harnesses for tool-call counts, "verified" and "inferred" confidence tags carefully distinguished: this is not vanity. It is how the engine knows what is real. The discipline is unusual for an LLM workflow tool, and it is what keeps the design from drifting into wishful thinking.

> **PORTABLE BEYOND CONTENT**
>
> None of these lessons are about blog posts. They are about *how to build agentic systems that stay honest under load.* The blog post output is incidental. The architecture is the artifact.

CLOSING

The architecture *is* the artifact.

The engine is interesting because it admits two things most AI writing tools do not. First, that AI-assisted writing has known failure modes and the only honest response is to design for them. Second, that an LLM agent's prose instructions are advisory, and that real enforcement has to live at the tool layer, the file system, or a deterministic script.

The discipline expresses itself at every layer: failure modes treated as design constraints, layered enforcement that does not depend on prose holding, credibility tracked as a structural parameter, and a knowledge base that gives the corpus a memory across runs.

The result is a system where every claim earns its place through sourced evidence, every post earns its publish through layered validation, and the corpus stays accountable for as long as it lives.

What ships is documentation discipline applied to AI authoring, expressed as a system. The lessons it encodes (tool absence over prose instruction, cross-model verification, parallel dispatch with explicit semantics, format calibration over format gating, deterministic validators backing AI judgment, orchestrator-controlled disk writes) are the part worth lifting into other agentic workflows. *The blog post output is incidental — what travels is the architecture itself.*

COLOPHON

Ready Solutions *AI*.

Claude-first AI integration for engineering organizations that need their adoption to actually land, and stay landed.

ABOUT THIS PAPER

Edition

First edition, May 2026.

Subject

Architectural philosophy and lessons from a multi-phase AI authoring pipeline.

Scope

Concept, principle, and generalizable lesson. Not an implementation guide.

ABOUT THE AUTHOR

Mitchel Lairscey

Founder, Ready Solutions AI. Engineering manager, Claude Code adoption practitioner, builder of the system this paper describes.

Practice areas

Claude Code rollouts. MCP server design. Custom skills, agents, hooks. AI strategy for technical leaders.

CONTACT

Web

readysolutions.ai

Engagement

Strategy sessions, integration projects, team training, and standing advisory work for SaaS engineering organizations.

Feedback

Disagreement is welcome. Sharper critique sharpens the next edition.

EDITION V.2026.05