The Intelligence Council

Provider-Agnostic, Domain-Specialized Multi-Model Review
for Autonomous AI Systems

A Practitioner's Architecture and Case Study

Adnan Tanveer (Addy)
Speedrun AI Labs
Sydney, Australia
speedrunlab.ai

July 12, 2026 | Paper Version 1.0, submitted to SSRN

NOTE ON TIME-SENSITIVITY: This paper documents a multi-model review architecture and a set of model rosters current as of July 2026. The specific models named in the domain rosters (Section 6) are the strongest available at the time of writing, ranked by third-party benchmarks that are re-measured continuously; model versions, slugs, and rankings change on the order of weeks. The *architecture*, meaning a provider-agnostic substrate, independent multi-vendor panels, domain-specialised roster selection, and structured verdict synthesis, is intended to be stable across model generations. Readers should treat the named rosters as worked examples of the selection *method*, re-run the selection against current benchmarks before deploying, and verify version compatibility of any configuration shown.

Abstract

A widely reported, and widely debated, figure holds that roughly 95% of enterprise AI pilots deliver no measurable return. A recurring explanation, advanced most prominently by the founder of the workflow-automation platform n8n, is architectural rather than technological: systems that bind themselves to a single AI model inherit that model's blind spots, outages, regressions, and biases with no recourse. The proposed remedy is to stay model-agnostic and to locate value in the orchestration layer *between* the models rather than inside any one of them.

This paper operationalises that principle into a concrete, deployable pattern we call the **Intelligence Council**: a standing capability that, on a provider-agnostic substrate, convenes a panel of independent frontier models to review a plan, a code change, or a decision before it is acted upon, and then synthesises their verdicts into consensus, dissent, and blockers. Its value is *independence*: different vendors, different training corpora, and genuinely separate reasoning, which a single model, however capable, cannot supply when asked to check its own work. The pattern is model-agnostic by construction and harness-agnostic in deployment: it reduces to a set of parallel HTTP calls to a model aggregator, plus optional native subagents where the host offers them, so the same council embeds in any coding harness, agent framework, or pipeline without being tied to a particular vendor or runtime.

We make four contributions. (1) We formalise the council pattern and the independence argument that motivates it. (2) We document a reference implementation: a live seven-model general-purpose council operated at Speedrun AI Labs, including its provider-agnostic access recipe and a reproducible configuration failure mode, in which mandatory-reasoning models return empty verdicts when their token budget is set too low. (3) We present a four-domain taxonomy of specialised councils (Coding, SEO / Search, Cybersecurity, and Intelligence), giving for each a domain-appropriate model-selection criterion and a recommended roster grounded in July-2026 benchmarks; Speedrun operates domain-specialised councils in production, and we describe the generalisable method by which such panels are composed. (4)

We define a synthesis protocol that converts per-model verdicts into an actionable, auditable recommendation. This is a practitioner experience report.

Keywords: multi-model architecture, model-agnostic orchestration, LLM-as-judge, model ensembles, AI decision review, OpenRouter, Amazon Bedrock, agentic AI, autonomous agents, domain specialisation, Speedrun AI Labs

1. Introduction

At the RAISE Summit in mid-2026, Jan Oberhauser, founder of the workflow-automation platform n8n, argued that the reason a large majority of AI projects fail is a single architectural decision: *dependence on one AI model* [1]. He paired the claim with the widely cited, and widely contested, MIT-affiliated finding that approximately 95% of enterprise AI pilots deliver no measurable return [2]. That figure has been criticised for a narrow definition of "return" and a small sample, so we use it as a directional signal of a real problem, not as a precise measurement. His platform's answer is to remain deliberately model-agnostic, plugging into OpenAI, Anthropic, and others interchangeably, on the thesis that "the value sits between the models, not inside them." The market has, at least in part, validated the position: n8n raised at a reported $5.2B valuation with roughly 1,400 enterprise customers, and organisations including Meta, Vodafone, and Mercedes-Benz build on the platform [1].

The observation is easy to state and surprisingly hard to act on. Most teams reach for a single "best" model, wire it into their pipeline, and treat its output as authoritative. When that model is confidently wrong, silently regresses after a provider update, refuses on a false safety trigger, or is simply unavailable, the entire system inherits the failure. There is no second opinion because there is no second brain, only the same brain asked twice.

This paper argues that model-agnosticism is necessary but not sufficient. Being *able* to swap models is a substrate property; it does not, by itself, catch a bad decision. The missing discipline is *review*: before an autonomous system takes a consequential, hard-to-reverse action, a panel of independent models should be convened to pressure-test the decision, and their verdicts should be synthesised into a clear recommendation. We call this capability the Intelligence Council. It is model-agnosticism turned from a procurement fact into an operational habit.

The remainder of the paper is organised as follows. Section 2 catalogues the concrete ways single-model systems fail. Section 3 defines the council pattern and the independence property that gives it its value. Section 4 examines the substrate layer, the provider-agnostic access tier, and compares the two practical model sources, OpenRouter and Amazon Bedrock. Section 5 documents a live seven-model reference council, including its exact invocation recipe and a reproducible configuration footgun. Section 6 presents the four domain-specialised councils with selection criteria and benchmark-grounded rosters. Section 7 specifies the verdict-synthesis protocol. Sections 8 and 9 cover operational considerations and threats to validity. Section 10 situates the work in related literature.

2. Why Single-Model Systems Fail

The failure of single-model architectures is not a story about weak models. Frontier models in 2026 are extraordinarily capable. It is a story about *correlation*: when one model is your only reviewer, every error it is prone to becomes an error your whole system is prone to, with nothing positioned to catch it. We observe five recurring modes.

Correlated blind spots. A model asked to generate a solution and then to review that same solution shares the exact priors, training artefacts, and reasoning habits that produced the original. It tends to ratify its own work.

Self-review raises confidence far more than it raises correctness.

Silent regression. Providers ship near-continuous updates. A prompt that produced correct output last week can degrade after a model revision with no error, no warning, and no version bump the caller notices. A single-model pipeline has no baseline to detect the drift.

Availability and rate coupling. When the one model is down, throttled, or regionally unavailable, the dependent system is down. There is no graceful degradation to a second opinion.

Sycophancy and false refusal. A single model may agree with a flawed premise to be helpful, or refuse a legitimate request on a mis-fired safety heuristic. In both directions the system has no counterweight.

Domain mismatch. The best general reasoning model is frequently not the best model for a live-web SEO question, and a strong coding model may be mediocre at adversarial security analysis. A single fixed model is, by construction, a compromise across every task it is asked to perform.

Each mode is individually survivable and collectively corrosive. The common cure is the same: introduce genuinely independent judgement, and route the right independent judges to the right question.

3. The Council Pattern

An Intelligence Council is a standing capability that convenes a panel of independent models to review a defined target, whether a plan under consideration, a specific code diff, or a decision about to be executed, and returns a synthesised verdict. It is convened deliberately, not on every call: it is a gate for consequential, hard-to-reverse actions, not a tax on trivial ones.

Three properties define the pattern.

Independence is the product. The panel's worth comes from its members being different vendors trained on different data with different methods, giving genuinely separate reasoning rather than the session's own model re-rolled at a higher temperature. Seven samples from one model give you that model's opinion with error bars; one sample each from seven vendors gives you seven opinions. The latter catches classes of error the former cannot, because the errors are no longer perfectly correlated.

Independence has more than one axis. Vendor is the most visible axis, but not the only one. Members can also differ by training corpus and architecture, by hosting and access path, and by *jurisdiction*: labs in different regulatory and cultural regimes tend to have less-correlated training data, alignment choices, and refusal behaviour. These axes are not the same, and a panel can score well on one while being concentrated on another. Our selection rule is capability first: rank on the domain benchmark, and where two candidates sit within benchmark noise, break the tie toward vendor and jurisdictional spread to reduce correlated failure. Section 6.4 returns to this point, where a pure capability ranking happens to concentrate in one jurisdiction, and Section 9 treats the limits of the independence assumption directly.

The reviewer is not the author. The model that drafted the work, often the model running the session, may sit on the panel, but it must not also be the sole synthesiser of the panel's verdict. Collect all members' independent judgements first, then synthesise neutrally. A model grading its own homework is the single-model failure mode wearing a council's clothes.

Structured verdicts, not vibes. Each member returns a machine-comparable verdict per item (for example APPROVE / APPROVE-WITH-CAUTION / REJECT), the specific failure vector it foresees, a safer alternative, and one overall call. Free-form prose from seven models is unactionable; a verdict matrix is a decision instrument (Section 7).

Model-agnostic and harness-agnostic. The council depends on neither a specific model nor a specific runtime. Members are reached through a substrate that abstracts the vendor (Section 4), so any model can join or leave by configuration rather than code. And the mechanism itself is only a set of parallel HTTP calls to that substrate, plus optional native subagents where the host provides them, so the same council embeds in any environment that can make an HTTP request: a coding harness such as an agentic CLI or IDE assistant, an autonomous agent framework, a CI or deployment pipeline, a serverless function, or a bespoke orchestrator. Nothing in the pattern is bound to the tool the operator happens to run, which is what lets a team adopt it without re-platforming.

The pattern is intentionally modular along two axes. The *substrate* axis (Section 4) is how you reach many vendors' models through one interface. The *roster* axis (Section 6) is which models you convene for a given class of question. Holding the substrate fixed and varying the roster by domain is what turns a single review panel into a portfolio of specialised councils.

4. The Substrate: Sourcing the Models

A council is only as practical as the ease of reaching many vendors' models through one account, one request shape, and one bill. Rather than integrating each vendor separately, a council sources its models from an *aggregator*: a single gateway that resells access to models from many vendors behind one unified API. This aggregator is the "layer between the models" where the thesis locates value. In 2026 two aggregators are the practical choices: **OpenRouter** and **Amazon Bedrock**. Either can supply a full multi-vendor roster; the choice between them is an engineering and governance decision, not an ideological one.

Dimension	OpenRouter	Amazon Bedrock
Model breadth	Very broad; hundreds of models across most vendors behind one API	Broad within partnered providers; a curated catalogue
Onboarding a new model	Change one slug string	Enable in the console; occasionally region-gated
Billing	Single consolidated account, pay-per-token	Consolidated on the AWS bill; committed-use / provisioned options
Data governance	Aggregator-mediated; check per-endpoint retention / zero-data-retention flags	In-account, VPC / PrivateLink, regional residency, IAM-scoped access
Latency	One extra network hop; typically small	In-region and low; provisioned throughput available
Best fit	Fast iteration and maximum roster breadth	Regulated enterprises needing in-account governance

Table 1: The two practical model-sourcing aggregators for a multi-vendor council. The council pattern is source-agnostic; the reference implementation in Section 5 sources its non-native members from OpenRouter, and much of the same roster can be sourced through Amazon Bedrock where governance requirements demand it, subject to that catalogue's coverage.

The decisive property, whichever source you pick, is **uniformity**: every model is callable with the same request shape, so adding or swapping a member is a configuration change, not a code change. That is the property that keeps a council easy to evolve as models come and go. OpenRouter maximises roster breadth and speed of iteration, and is the source used for the non-native members of our reference council. Amazon Bedrock is the appropriate source when in-account data residency, VPC isolation, and IAM-scoped access are hard requirements; the council architecture transfers unchanged, and each member is remapped to the gateway's catalogue where available. The two catalogues do not overlap completely: some vendors' newest models, and some non-US labs, may appear on one source and not the other, so the reachable roster is the intersection of the

desired members and the chosen source's coverage. Verify availability per member at deployment time rather than assuming parity between sources.

5. Reference Implementation: A General Seven-Model Council

Speedrun AI Labs operates a standing general-purpose review council, convened before shipping any sensitive, platform-wide, or hard-to-reverse change. We document it in full because its internals are the concrete anchor for the pattern; the domain-specialised councils of Section 6 are variations on this same machinery.

5.1 Roster

The general council has seven members drawn from two pools: three native Anthropic models reached directly, and four models from other vendors reached through the OpenRouter aggregator. The mix is deliberate. No single vendor supplies a majority, so no single vendor's blind spot can carry a verdict.

#	Member	Vendor	Pool
1	Opus 4.8	Anthropic	Native
2	Sonnet 5	Anthropic	Native
3	Fable 5	Anthropic	Native
4	GPT-5.5	OpenAI	Aggregator
5	Gemini 3.1 Pro	Google	Aggregator
6	Kimi K2.7 Code	Moonshot AI	Aggregator
7	DeepSeek V4 Pro	DeepSeek	Aggregator

Table 2: The general-purpose reference council. Four vendors, two access pools. The session's own model may be a member but is never also the sole synthesiser of the verdict.

5.2 Invocation Recipe

The four aggregator members are called in parallel, in the background, over plain HTTPS. The three native members are spawned as parallel review tasks in the same round. All seven receive the identical review brief and return only their structured verdict. A representative call to the aggregator pool, with credentials and endpoints shown as placeholders, is:

```
KEY=$(cat ~/.config/openrouter/key)          # never inline a secret
call() {                                      # $1 = model slug, $2 = output file
  jq -n --arg m "$1" --arg s "$SYS" --arg u "$USR" \
    '{model:$m, max_tokens:8000,
      messages:[{role:"system",content:$s},{role:"user",content:$u}]}' \
  | curl -s --max-time 280 https://openrouter.ai/api/v1/chat/completions \
      -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" \
      -H "HTTP-Referer: https://speedrunlab.ai" -H "X-Title: Intelligence Council" \
      --data-binary @- > "$2"
}
call "openai/gpt-5.5"                gpt.json      &
call "google/gemini-3.1-pro-preview" gemini.json   &
call "moonshotai/kimi-k2.7-code"     kimi.json     &
call "deepseek/deepseek-v4-pro"      deepseek.json &
wait
```

Two construction details matter for correctness. Payloads are assembled with a JSON tool (`jq -n --arg`) rather than string interpolation, so a review prompt containing quotes or braces cannot corrupt the request. And

no secret, credential, or item of personal data is ever placed in the review brief: the council receives the technical change and generic code patterns only, never keys, tokens, or private records.

5.3 A Reproducible Failure Mode: Empty Verdicts from Mandatory-Reasoning Models

Several current frontier models enforce *mandatory reasoning*: the endpoint will not disable the model's internal chain-of-thought, and that hidden reasoning consumes output tokens before any user-visible answer is produced. In our roster this applies to Gemini 3.1 Pro, Kimi K2.7 Code, and Fable 5. The recurring practitioner failure is subtle and expensive to diagnose:

- The caller sets a modest `max_tokens` (say 1,024) to control cost.
- The model spends that entire budget on hidden reasoning.
- The response returns with `finish_reason: "length"` and **empty visible content**.
- The council appears to have a silent, non-voting member, and the reviewer sees a blank where a verdict should be.

The fix is counter-intuitive but simple: do *not* lower the reasoning effort to save tokens, and do *not* attempt to disable reasoning (the endpoint refuses). Instead, raise `max_tokens` to a generous ceiling, 8,000 in our configuration, so the hidden reasoning and a full verdict both fit. If a member still returns empty with `finish_reason: "length"`, re-run that member alone at a higher ceiling (12,000). This single configuration point is, in our experience, the most common reason a multi-model panel silently degrades to fewer voices than the operator believes are present.

Raising the ceiling reduces this failure but does not detect it, and a member can still exhaust even a generous budget or change its hidden-reasoning behaviour after a provider update. The caller should therefore add an active liveness guard: assert that each member returned non-empty content, and inspect `finish_reason`, before counting the member as having voted. A member that returns empty or truncated is re-run or recorded as absent, never silently treated as a verdict. The guard turns a silent degradation into a visible, handled event.

5.4 Cost and Latency

The three native members incur no marginal API cost. The four aggregator members together cost a few cents per convening at current pricing. Because all members are called in parallel in the background, wall-clock latency for a full seven-model round is bounded by the slowest single member (heavy reasoning models dominate), not the sum. One caution applies: because the round waits on the maximum of seven concurrent draws, its latency tends toward the tail of the latency distribution rather than the median, and a single stalled member sets the pace. The practical mitigation is to proceed on the first N of seven responses once a quorum has returned, treating a member that misses the deadline as absent rather than blocking the round. With that guard, a council review is cheap enough to make a standing habit and fast enough not to stall a deployment.

6. The Four Domain Councils

A single general council is a strong default, but the domain-mismatch failure of Section 2 argues for specialisation: the best panel for an SEO decision is not the best panel for a security review. We propose a taxonomy of four domain councils, each defined by a *selection criterion*, the property that makes a model a good judge for that domain, and populated with a *recommended roster* drawn from the strongest models against that criterion as of July 2026.

Two honesty notes frame this section. First, Speedrun operates domain-specialised councils in production, including a search / SEO council and a combined coding-and-security council, alongside the general council of Section 5. The *specific* production rosters, prompts, and routing logic are proprietary and are not disclosed here;

what follows is the generalisable method for composing such panels, with rosters chosen purely from public benchmarks so that any reader can reproduce the selection. Second, the selection *criterion* is the durable contribution: model names churn weekly, but "for search, choose models that retrieve the live web and cite it" does not.

Third, each roster below follows one reproducible selection rule: take the top models on the domain's benchmark as of the stated date, require at least four vendors (and, where a non-US model clears the bar, more than one jurisdiction) for independence, and admit only members reachable on the operator's chosen source (Section 4). The named tables are illustrative snapshots of applying that rule in July 2026, not Speedrun's production configuration.

6.1 The Coding Council

Selection criterion. Strength on software-engineering benchmarks (SWE-bench-class coding indices), agentic tool-use reliability, and the ability to read and reason about a diff in the context of a surrounding codebase. A coding reviewer's job is to find the bug the author missed and the simplification the author did not see, so raw coding capability dominates the roster choice.

Recommended roster (July 2026). Ranked by the Artificial Analysis coding index, the strongest coding models cluster tightly at the top; a good council spreads them across vendors to preserve independence.

Model	Vendor	Coding index
GPT-5.6 (Sol)	OpenAI	77.4
Fable 5	Anthropic	76.5
GPT-5.5	OpenAI	74.9
Opus 4.8	Anthropic	74.3
Grok 4.5	xAI	72.4
Gemini 3.1 Pro	Google	68.8
Kimi K2.7 Code	Moonshot AI (China)	60.8
DeepSeek V4 Pro	DeepSeek (China)	59.4

Table 3: Recommended coding-council members and Artificial Analysis coding-index scores, retrieved via the OpenRouter model catalogue on 12 July 2026 and illustrative of that date. A five-member panel spanning at least four vendors and two jurisdictions (for example GPT-5.6, Fable 5, Grok 4.5, Gemini 3.1 Pro, DeepSeek V4 Pro) balances top-of-benchmark strength against vendor independence.

6.2 The SEO / Search Council

Selection criterion. Live web retrieval, citation grounding, and recency. SEO and search decisions turn on the *current* state of the web and of ranking behaviour, information that is, by definition, after any model's training cutoff. Here raw reasoning is secondary; what matters is whether the model actually *sees* the live web and grounds its answer in retrievable sources rather than reciting stale training data. A brilliant model with a January knowledge cutoff is the wrong judge for a July SERP question.

Recommended roster (July 2026).

Model / service	Vendor	Why it belongs
Sonar Pro	Perplexity	Search-native; live retrieval with inline citations on every answer; strong on published factuality evaluations (a vendor-reported F-score near 0.858; treat vendor self-benchmarks as directional)

Model / service	Vendor	Why it belongs
Sonar Reasoning	Perplexity	Chain-of-thought over freshly retrieved sources for analytical search questions
Grok 4.5	xAI	Real-time access to current web and social signal; strong on "what is true right now"
Gemini 3.1 Pro (grounded)	Google	Native search grounding; multimodal; deep index integration
GPT-5.x (web search)	OpenAI	Tool-based live browsing with source attribution as a cross-vendor check

Table 4: Recommended SEO / search-council members. Selection is by web-grounding and recency, not by general-reasoning rank, the opposite emphasis to the coding and intelligence councils. Figures cited are vendor-reported where noted; this roster is an illustrative July-2026 snapshot.

The contrast with Section 6.1 is the point of the whole taxonomy: a model that tops the coding index can be a poor SEO reviewer if it cannot see today's web, and a search-native service that is unremarkable at code can be the most trustworthy voice on a live-ranking question. Route the question to judges selected for *that* question.

6.3 The Cybersecurity Council

Selection criterion. Demonstrated capability on security-specific evaluations, including capture-the-flag success rate (for example CyBench) and real-world vulnerability detection, exploitation, and patching (for example BountyBench and SEC-bench), together with secure-code generation and a low false-positive rate on vulnerability claims. A security reviewer is asked to think adversarially, to find the exploit path the author did not consider. General reasoning strength is necessary but not sufficient; security-benchmark evidence is the discriminator.

Recommended roster (July 2026). The current CyBench capture-the-flag leaderboard is topped by the Claude family, with GPT and Grok models close behind on the harder CTF and vulnerability suites. A strong security council draws these leaders and spreads them across vendors, then prompts every member adversarially.

Model	Vendor	Security evidence
Opus 4.8	Anthropic	Claude family leads public CyBench CTF leaderboards; among the strongest on detect-and-patch (reported leaderboard figures vary by subset and date, so confirm the exact split before relying on a number)
GPT-5.5	OpenAI	Near-saturation on narrow cyber-range suites; strong exploit reproduction and secure-code generation
Grok 4.5	xAI	Highest non-Claude CyBench performer in its line; adversarial, and current on newly disclosed CVEs
Gemini 3.1 Pro	Google	Long-context vulnerability analysis across large codebases; independent second vendor
DeepSeek V4 Pro	DeepSeek	Strong code and reasoning at an independent vendor, widening cross-vendor coverage

Table 5: Recommended cybersecurity-council members and their security-benchmark evidence as of July 2026. Scores on CTF and vulnerability benchmarks (CyBench, BountyBench, SEC-bench, NYU CTF Bench) move quickly; confirm current standings at deployment time. Every member is prompted to attempt to refute the change's safety rather than confirm it.

Two operating rules distinguish a security council from the others. First, every member is prompted adversarially and told to default to "unsafe / concern" under uncertainty, because a missed vulnerability is far more costly than a false alarm that is later cleared. Second, because security evaluation is dual-use, the council is scoped to *defensive* review of the operator's own systems, assessing whether a change introduces a weakness,

not to producing offensive capability. Named security-benchmark rankings move quickly, so select from the current leaderboards at deployment time rather than from any fixed list.

6.4 The Intelligence Council

Selection criterion. Raw reasoning quality across domains: general-intelligence indices (GPQA-class reasoning, mathematics, multi-step planning) and cross-domain judgement. This is the roster for decisions that are not narrowly coding, search, or security questions: strategy calls, architectural trade-offs, ambiguous plans, and any high-stakes judgement where breadth of reasoning matters more than any single specialism. It is the direct generalisation of the reference council in Section 5.

Recommended roster (July 2026). Ranked by the Artificial Analysis intelligence index:

Model	Vendor	Intelligence index
Fable 5	Anthropic	59.9
GPT-5.6 (Sol)	OpenAI	58.9
Opus 4.8	Anthropic	55.7
GPT-5.5	OpenAI	54.8
Grok 4.5	xAI	53.8
Sonnet 5	Anthropic	53.4
GLM 5.2	Z.ai (China)	51.1
Gemini 3.1 Pro	Google	46.5

Table 6: Recommended intelligence-council members and Artificial Analysis intelligence-index scores, retrieved via the OpenRouter model catalogue on 12 July 2026. The roster spans five vendors and two jurisdictions while keeping every member near the top of the reasoning frontier.

On jurisdictional concentration. Ranked purely by the intelligence index, this roster leans heavily toward United States vendors, because the current top of that index does. We include Z.ai's GLM 5.2 on merit: it outranks Gemini 3.1 Pro on the same index, so a table claiming to rank by the index while omitting it would be inconsistent. Its inclusion also illustrates the point made in Section 3. A panel drawn entirely from one country's labs is a narrower form of independence than vendor diversity alone suggests, because those labs share overlapping training corpora, alignment norms, and a single regulatory regime, all of which can correlate their blind spots. We do not impose a nationality quota; selection is capability first. But where a non-US model clears the same bar, seating it strengthens the very property the council exists to provide. The countervailing cost is governance: some operators cannot route review briefs to endpoints domiciled in certain jurisdictions, which is a constraint on the reachable roster (Section 4), not a judgement on the model. Section 9 discusses why independence is a spectrum rather than a guarantee.

6.5 The Taxonomy at a Glance

Council	Selection criterion	What a member must be good at	Roster emphasis
Coding	Coding-benchmark strength; agentic tool use; diff comprehension	Finding bugs and simplifications in real code	Top coding index, cross-vendor
SEO / Search	Live web retrieval; citation grounding; recency	Seeing and citing the current web	Search-native and real-time models
Cybersecurity	CTF / vulnerability-benchmark performance; adversarial reasoning; low	Finding the exploit the author missed	Security-benchmark leaders and breadth

Council	Selection criterion	What a member must be good at	Roster emphasis
	false-positive		reasoners
Intelligence	General reasoning index; cross-domain judgement	Sound judgement on ambiguous, high-stakes calls	Top intelligence index, cross-vendor

Table 7: The four domain councils. The council architecture is identical across rows; only the roster and the review prompt change. Selection criteria are durable; the specific models that satisfy them are re-selected against current benchmarks at deployment time.

7. The Synthesis Protocol

Collecting seven verdicts is not a decision; synthesising them is. The protocol converts independent member verdicts into a single auditable recommendation.

Step 1: Shared brief. Write one system prompt (the reviewer's role and the bar to clear) and one user prompt (the change described precisely, with real excerpts, plus the exact verdict format required). Every member receives the identical brief. Ask each for: a per-item verdict (APPROVE / APPROVE-WITH-CAUTION / REJECT, or RESOLVED / STILL-CONCERNED / NEW-CONCERN on a re-review), the specific breakage vector, a safer alternative, an overall call, and the single most important thing missing. Instruct members to be decisive and brief.

Step 2: Parallel collection. Fire all members concurrently (Section 5.2). Extract each verdict; any member that returned empty with `finish_reason: "length"` is re-run at a higher token ceiling before synthesis, not silently dropped.

Step 3: Verdict matrix. Build a member-by-item table. This is the artefact that makes disagreement legible: a column where six members approve and one rejects is a different situation from a column where verdicts are evenly split, and the matrix shows it at a glance.

Step 4: Synthesise into three sections.

- **Consensus:** what all or most members approve; the safe-to-proceed core.
- **Dissent:** who pushed back and on what, naming the model. A lone dissent is not noise to be out-voted; a single vendor seeing a failure the others missed is exactly the independence the council exists to capture. Investigate it before discounting it.
- **Blockers:** anything a member flags as must-fix-before-shipping. Every blocker is verified against ground truth before it is believed *or* dismissed, because models raise false positives; an unverified blocker is a claim, not a fact.

Step 5: Recommendation. Close with a clear call and the one open decision for the human owner. The council advises; it does not autonomously ship.

A quorum policy makes the output deterministic: for example, proceed on unanimous or strong-majority APPROVE; hold on any REJECT until its blocker is verified and resolved; escalate a split panel to the human. The policy is a governance choice, but having *a* policy, rather than re-adjudicating each convening from scratch, is what makes the council a system rather than a ritual.

The synthesiser must not become a new single point of failure. The verdict matrix of Step 3 is built deterministically in code: it records each member's stated verdict verbatim, and no model rewrites another's vote. Consensus, dissent, and blocker lists are derived from that matrix by rule, not distilled by a model that is free to agree with itself. Where a model is used to draft the prose synthesis, it operates on the fixed matrix, is explicitly instructed never to drop or soften a dissent, and its draft is checked back against the matrix so that a

lone dissent survives intact. Collapsing seven verdicts through one model that may ignore the votes it disagrees with would reintroduce, at the final step, precisely the single-model dependence the council exists to remove.

7.1 A Worked Convening

This paper was put through the process it describes. A seven-member council (the reference roster of Section 5) was convened to review the draft; each member received the identical brief and returned an independent verdict in the format above. The outcome illustrates both the mechanics and the value.

The panel returned five "accept with revisions" verdicts and two "major revision" verdicts. No member judged the draft ready to ship unchanged, and none rejected it. Consensus formed around two points that a single reviewer had not surfaced with the same force: that the draft asserted the council's benefit without a worked example, and that several benchmark figures needed dated primary citations. The verdict matrix also carried a lone dissent that proved correct on inspection. One member, from a different vendor than the draft's author, observed that the synthesis step of Section 7 was an unguarded single point of failure, because a model left free to summarise the panel could quietly bury a dissent it disagreed with. No other member raised it. That objection is the reason Section 7 now specifies a deterministic verdict matrix and the requirement that a lone dissent survive synthesis intact. A single-model review would have been structurally unable to produce the objection, since the single model would have been the very component the objection concerns.

The revisions in this version, including the roster and independence changes of Sections 3 and 6.4, the liveness and tail-latency guards of Section 5, and the citation tightening throughout, follow directly from that convening. The episode is a single case, not a controlled study (Section 9), but it is a concrete instance of an independent panel surfacing a defect its own author's model had not. It is also self-referential by nature: the council was operated by the paper's own author and reviewed the author's own draft, so it demonstrates the mechanism rather than independently validating it. We report it as an illustration, and treat controlled evaluation as future work.

8. Operational Considerations

When to convene. A council is a gate for consequential, sensitive, or hard-to-reverse changes, and for a second-round check that a revised plan resolved the first round's concerns. It is deliberately *not* convened for trivial or surgical changes, where a single reviewer or none is appropriate. Over-convening trains operators to ignore verdicts, so reserve the council for decisions where an independent multi-vendor read is worth its seconds and cents.

Secrets hygiene. Because a convening fans the review brief out to multiple external vendors, the brief must never contain secrets, credentials, tokens, or personal data. Send the technical change and generic patterns only. This is a hard rule, not a preference: a council is a deliberate act of disclosure to several third parties at once.

Slug and version churn. Model identifiers change as vendors ship new versions (a code model moves from K2.6 to K2.7; a Gemini generation from 3 to 3.1). A council must re-verify its members' identifiers periodically and when a call returns not-found, and re-select its roster against current benchmarks rather than freezing a list. The rosters in Section 6 are a July-2026 snapshot by construction.

Transport robustness. Heavy-reasoning members are slow, and naive synchronous calls time out. Call over a transport with a generous timeout, in the background, in parallel, so that one slow member does not block the panel and a single failure resolves to a null vote rather than a crashed convening.

Self-synthesis bias. The model orchestrating the session must not be the sole grader of a verdict it contributed to. Where the orchestrator is itself a member, its own verdict is collected like any other and the synthesis is performed against the full panel, not authored by the interested party.

Correlated dependencies. Independence can be quietly undermined by a shared component even when the models themselves differ. If every non-native member is reached through a single aggregator, that aggregator is a common dependency and a common point of failure; if every member sits in one jurisdiction, the panel shares one regulatory and cultural prior (Section 6.4). Neither is fatal, but both should be named and, where the decision warrants it, mitigated by sourcing across more than one path and seating more than one jurisdiction.

Harness portability. Because a convening is only parallel HTTP calls to the model source plus, optionally, whatever native subagent mechanism the host offers, the council is portable across development environments. The aggregator members are identical whether the operator drives them from an agentic coding CLI, an IDE assistant, a scheduled job, or a custom service; only the native-model access path is host-specific, and it is optional (a council can run entirely on aggregator members if the host has no native subagents). This portability is what makes the council adoptable as a shared review gate across a team that uses different tools, rather than a feature of one of them.

9. Threats to Validity

This is a practitioner report, and its claims should be read with the following limits. Benchmarks are proxies: the Artificial Analysis coding and intelligence indices, and the security and factuality benchmarks cited, are useful rankings but imperfect predictors of review quality on any specific task; a model's index is a selection heuristic, not a guarantee. The rosters are a July-2026 snapshot and will date quickly. Independence is a spectrum, not a binary: as vendors converge on similar architectures and training data, council members' errors may become more correlated over time, eroding the property the council depends on, so operators should periodically test whether their members still disagree in informative ways. The council adds latency and cost, and is inappropriate for high-frequency or trivial decisions. Finally, we report qualitative operating experience, in which the pattern has repeatedly surfaced concerns a single model missed (the convening of Section 7.1 is one documented instance), rather than a controlled experiment isolating the council's marginal effect on outcome quality; a rigorous A/B evaluation of council-gated versus single-model decisions across a task distribution is future work.

10. Related Work

The council pattern sits at the intersection of several established lines. *Mixture-of-Agents* methods [3] show that aggregating the outputs of multiple LLMs and having later layers refine them improves quality over any single model, establishing the general value of multi-model composition. *LLM-as-a-judge* work [4] validates using strong models to evaluate other models' outputs and characterises the biases (including self-preference) that motivate our "reviewer is not the author" rule. *Self-consistency* [5] improves single-model reliability by sampling multiple reasoning paths and taking a majority, a same-model analogue whose limitation, correlated errors, is exactly what cross-vendor independence addresses. *Multi-agent debate* [6] has models argue toward a more accurate joint answer, a complementary interaction mode to independent verdict synthesis. At the platform level, model-agnostic orchestration frameworks such as n8n [1] provide the substrate on which such patterns run, and the MIT enterprise-pilot findings [2] supply the motivating problem. Our contribution is not a new aggregation algorithm but a deployable operational discipline: an independent, provider-agnostic, domain-specialised review gate for autonomous systems, documented from production practice with a reproducible configuration.

11. Conclusion

The reason a majority of AI projects disappoint is less often the model than the architecture around it: a single model, trusted alone, with no independent check on its confident mistakes. Staying model-agnostic is the necessary substrate, but agnosticism only pays off when it is exercised, when, before a consequential action, the system convenes independent judges and listens to their disagreement. The Intelligence Council turns "the value is between the models" from a slogan into a habit: a provider-agnostic panel, specialised by domain, that reviews decisions and surfaces the dissent a single model cannot. It is cheap enough to run routinely, simple enough to build on today's substrates, and portable enough to embed in whatever coding harness or agent framework a team already uses, since it asks only for an HTTP call and, optionally, a subagent. The models will keep changing, and so will the tools that drive them; the discipline of asking several genuinely different minds before you act is what endures.

References and Data Sources

References

[1] The Next Web (TNW). Coverage of Jan Oberhauser (founder, n8n) at the RAISE Summit on single-model dependence as a cause of AI-project failure, and n8n's model-agnostic positioning (reported ~$5.2B valuation; ~1,400 enterprise customers; adopters including Meta, Vodafone, and Mercedes-Benz). 2026. Figures are trade-press reported and should be confirmed against a primary source before load-bearing use.

[2] MIT NANDA / MIT Project on AI. "The State of AI in Business 2025." Source of the widely cited claim that approximately 95% of enterprise generative-AI pilots produced no measurable P&L return. The figure has been publicly contested for a narrow definition of "return" and a small sample; it is treated here as directional, not precise.

[3] Wang, J., Wang, J., Athiwaratkun, B., et al. "Mixture-of-Agents Enhances Large Language Model Capabilities." 2024. Layered aggregation of multiple LLMs' outputs outperforms any single constituent model; motivates the value of multi-model composition.

[4] Zheng, L., Chiang, W.-L., Sheng, Y., et al. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena." NeurIPS Datasets and Benchmarks Track, 2023. Establishes strong LLMs as evaluators of other models and documents judge biases including self-preference; grounds the "reviewer is not the author" rule (Section 3).

[5] Wang, X., Wei, J., Schuurmans, D., et al. "Self-Consistency Improves Chain-of-Thought Reasoning in Language Models." ICLR, 2023. Sampling multiple reasoning paths from one model and taking a majority; the same-model analogue whose correlated-error limit cross-vendor independence is designed to overcome.

[6] Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., Mordatch, I. "Improving Factuality and Reasoning in Language Models through Multiagent Debate." 2023. Multiple models argue toward a joint answer; a complementary interaction mode to independent verdict synthesis.

[7] Zhang, A. K., et al. "Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models." arXiv:2408.08926, 2024. 40 professional capture-the-flag tasks drawn from four competitions, scored by unguided end-to-end success; the primary selection benchmark for the Cybersecurity council (Section 6.3).

[8] "SEC-bench: Automated Benchmarking of LLM Agents on Real-World Software Security Tasks." arXiv:2506.11791, 2025. Real-world software-security tasks (detection and repair) for LLM agents; a secondary Cybersecurity-council selection signal.

[9] BountyBench. Real-world bug-bounty evaluation of offensive and defensive cyber agents (order of 40 tasks across ~25 web systems), scored on detect / exploit / patch with dollar-impact weighting. Used alongside CyBench [7], SEC-bench [8], and NYU CTF Bench for Section 6.3.

[10] Perplexity AI. Sonar and Sonar Pro API documentation and factuality reporting ("Introducing the Sonar Pro API"), 2025 to 2026. Sonar models perform live web retrieval with inline citations on every answer; the vendor-reported factuality F-score (~0.858 for Sonar Pro) is cited as directional in Table 4. OpenRouter slugs: perplexity/sonar-pro, perplexity/sonar-reasoning-pro, perplexity/sonar-deep-research.

[11] Artificial Analysis. Independent third-party LLM benchmarking: the Intelligence Index, Coding Index, and Agentic Index. All index figures in Tables 3 and 6 were retrieved from Artificial Analysis data as surfaced in the OpenRouter model catalogue on 12 July 2026 (exact figures in Data Sources below).

[12] OpenRouter. Unified multi-model API and model catalogue (openrouter.ai), 2026. Aggregator used to source all non-native council members; a single endpoint (/api/v1/chat/completions) with one credential reaches every third-party model named here.

[13] Amazon Web Services. "Amazon Bedrock" managed foundation-model gateway documentation, 2026. Alternative in-account model source for governance-constrained deployments (Section 4); catalogue coverage differs from OpenRouter and must be checked per member.

[14] Design Arena. Head-to-head model rankings (ELO, win rate) by category, referenced as a secondary capability signal for coding-style tasks. Accessed via the OpenRouter catalogue, 12 July 2026.

Data Sources (live figures retrieved 12 July 2026 via the OpenRouter model catalogue [11][12])

Coding council (Table 3), Artificial Analysis coding index: GPT-5.6 Sol (openai/gpt-5.6-sol) 77.4; Fable 5 (anthropic/claude-fable-5) 76.5; GPT-5.5 (openai/gpt-5.5) 74.9; Opus 4.8 (anthropic/claude-opus-4.8) 74.3; Grok 4.5 (x-ai/grok-4.5) 72.4; Gemini 3.1 Pro (google/gemini-3.1-pro-preview) 68.8; Kimi K2.7 Code (moonshotai/kimi-k2.7-code) 60.8; DeepSeek V4 Pro (deepseek/deepseek-v4-pro) 59.4.

Intelligence council (Table 6), Artificial Analysis intelligence index: Fable 5 59.9; GPT-5.6 Sol 58.9; Opus 4.8 55.7; GPT-5.5 54.8; Grok 4.5 53.8; Sonnet 5 (anthropic/claude-sonnet-5) 53.4; GLM 5.2 (z-ai/glm-5.2) 51.1; Gemini 3.1 Pro 46.5. For reference, DeepSeek V4 Pro (44.3) and Kimi K2.7 Code (41.9) fall below the table's cutoff on this index, which is why they appear on the coding but not the intelligence roster.

Cybersecurity council (Section 6.3): member selection guided by CyBench [7], BountyBench [9], SEC-bench [8], and NYU CTF Bench standings current at the time of writing; the Claude family led the public CyBench CTF leaderboard, with GPT and Grok models among the next strongest. Exact percentages are omitted deliberately, since leaderboard figures vary by subset and date.

SEO / Search council (Table 4): selection by live-retrieval capability and vendor-reported factuality rather than a single numeric index; Perplexity Sonar Pro's reported factuality F-score (~0.858) [10] is the one cited figure and is vendor-reported.

All model slugs above were verified resolvable on OpenRouter on 12 July 2026. Benchmark rankings are re-measured continuously by their maintainers; the figures here are a dated snapshot and will drift. Re-run the selection against current data before deploying any roster.