The Sidecar Oracle:
A Provider-Agnostic Supervisor Harness for Real-Time Voice Agents

Sam Savage
SecondMind — github.com/samthedataman/secondmind
July 2, 2026 · v1.0

Abstract

Production voice agents operate under a hard latency budget: to sound human they must respond in well under a second, which structurally prevents the model that *talks* from also being the model that *deliberates*. We survey twenty-four published results across five research threads — dual-process agent architectures, the failure of intrinsic self-correction, external verifier and monitor models, inference-time multi-model oversight, and feedback-driven prompt optimization — and find they converge on a single architecture: a second, slower model running *beside* the conversation, off the latency path, injecting silent corrections through whatever control channel the serving platform natively exposes. The central obstacle to deploying this architecture is not the supervisor but the heterogeneity of the voice-AI substrate: speech-to-speech models, STT→LLM→TTS pipelines, agent frameworks, and raw speech components each expose different event vocabularies and different (sometimes no) injection channels. Our primary contribution is a runtime harness that makes supervision **provider-, LLM-, speech-to-speech-, and TTS-agnostic**: a canonical event algebra with per-provider adapters, a deterministic state reducer carrying ground-truth outcome signals, capability-aware compilation of one abstract correction decision into thirteen platforms' native controls, and degrade-safety semantics under which supervisor failure composes with the live call as a no-op. On top of the harness we describe an outcome-driven improvement loop in which calls are scored against verified tool results and a versioned *standing directive* — a prompt layer the harness owns on every platform — is optimized OPRO-style from accumulated reports, without ever touching the operator's own system prompt and without placing optimization pressure on the supervised agent. The harness is open source (MIT) with a hosted control plane.

1 Introduction

A voice agent is one mind on a stopwatch. Perceived conversational quality collapses when response latency exceeds roughly a second, so every serving stack — whether an audio-native speech-to-speech model or an STT→LLM→TTS pipeline — optimizes the talking model for speed. The consequence is architectural, not incidental: everything that determines whether a call *succeeds* is deliberation the latency budget forbids. Reading distress in a caller's phrasing, noticing a mid-call language switch, catching the agent about to confirm a booking whose API call silently failed, remembering that this operator never quotes fees — each requires the slow, contextual reasoning that the fast path cannot afford.

The standard industry response is prompt enlargement. But prompts are frozen at call start, and the moments that matter are precisely the moments the prompt did not anticipate. Human call centers solved the identical problem decades ago without retraining agents mid-shift: a supervisor listens silently and slides a note across the desk. The agent keeps talking; the note changes the call.

This paper makes three claims. First (§2), the published evidence, read together, already validates each component of that supervisor pattern for LLM systems — and, importantly, rules out the cheaper alternative of asking the talking model to check itself. Second (§3), the binding constraint in practice is substrate heterogeneity, and it is solvable with a compact formal contract: canonical events in, one abstract decision out, compiled per capability. Third (§4), the same harness position enables a safe self-improvement loop over a prompt layer the harness owns, sidestepping both the fragmented ownership of operator prompts and the known failure mode of optimizing against a monitor.

SecondMind implements all three; we use it as the running example and note where its design decisions are forced by the literature.

2 A Meta-Analysis of Machine Supervision

2.1 Thread 1: Dual-process architectures

Booch et al. proposed Kahneman's System 1/System 2 as an AI design blueprint [2]; Google DeepMind's *Talker-Reasoner* architecture [1] is its most direct agentic realization: a fast conversational Talker reads a belief state that a slower Reasoner asynchronously maintains, with the Talker explicitly permitted to "operate with a delayed view of the world." Two observations matter for our purposes. The Talker-Reasoner evaluation is qualitative — a single sleep-coaching case study with no benchmark results — so it supplies design precedent rather than efficacy evidence. And its authors flag the open problem of *when* the fast agent should defer to the slow one. The sidecar pattern inherits the architecture but resolves the deferral problem differently: the supervisor never blocks the talker; it intervenes only through asynchronous, confidence-gated injections (§3.4).

2.2 Thread 2: Self-correction fails without an external signal

The pivotal negative result is Huang et al. [3]: under *intrinsic* self-correction — the model reviewing its own answer with no external feedback — performance reliably *degrades*. GPT-3.5 falls from 75.9% to 74.7% on GSM8K over two self-correction rounds and from 75.8% to as low as 38.1% on CommonSenseQA; GPT-4 falls from 95.5% to 89.0% on GSM8K. Yet the same experiments show large headroom when correction is triggered by an *oracle* signal (GPT-3.5 GSM8K 75.9% $\rightarrow$ 84.3%; GPT-4 HotpotQA 49% $\rightarrow$ 59%). Self-Refine [4] and Reflexion [5] report substantial gains ($\sim$20% average; HumanEval pass@1 80% $\rightarrow$ 91%) precisely in settings where feedback is externally grounded (unit tests, environment signals). The synthesis is unambiguous: *the refinement mechanism works; self-generated judgment is the weak link.* A supervisor must therefore be a separate process with separate context and access to ground truth — in a voice call, the tool results and event stream that the runtime, not the model, holds.

2.3 Thread 3: External verifiers work, and small ones suffice

Cobbe et al. [6] established the generator/verifier split: a 6B verifier ranking a generator's samples outperforms a fine-tuned 175B model — "a boost approximately equivalent to a 30$\times$ model size increase" — while also documenting that verifiers under heavy selection pressure get gamed (performance degrades beyond $\sim$400 ranked samples). Lightman et al. [7] sharpened the granularity result: process supervision (per-step labels) reaches 78.2% on MATH versus 72.4% for outcome-only supervision, with better out-of-distribution generalization. Baker et al. [8] moved supervision to *runtime*: a prompted GPT-4o monitoring a frontier agent's chain of thought catches systemic

reward hacks with 95% recall versus 60% for an action-only monitor — and, critically, when the monitor's judgment was added to the training reward, agents learned *obfuscated* hacking and monitor recall collapsed toward zero. Shepherd [9] showed a purpose-trained 7B critic matches ChatGPT-scale judges at the critique task specifically.

Three design rules follow. (i) The supervisor can and should be small and cheap relative to the supervised agent. (ii) It should judge *turns* (process), not just call outcomes. (iii) Its judgments must never become direct optimization pressure on the supervised model — they should be *injected as context*, a rule SecondMind treats as inviolable (§4).

2.4 Thread 4: Multi-model oversight at inference time

Sakana AI's AB-MCTS [10] frames inference-time collaboration as a principled search: at each node the system chooses (via Thompson sampling) between going *wider* (a fresh attempt) and *deeper* (refining an existing one, using external feedback), with multiple frontier models participating; the multi-LLM variant outperformed every constituent model on ARC-AGI-2. Their *RL Teachers* result [11] is more surprising: a 7B teacher trained not to solve problems but to maximize *student uptake* (dense rewards from student log-probabilities) produced better students than distillation data derived from a 671B reasoner — AIME'24 23.3% vs. 20.0% (7B student) and 66.7% vs. 63.3% (32B student). The correct objective for a helper model is what the downstream model does with its output, not the helper's own task skill — which is exactly the objective a whispering supervisor should hold. The Darwin Gödel Machine [12] extends the theme to open-ended self-improvement: agents rewriting their own code, retaining variants by empirical benchmark fitness, moved SWE-bench from 20.0% to 50.0%. Multiagent debate [15, 16] contributes the adversarial framing, with one caution from Huang et al.'s matched-compute measurement (debate 83.0% vs. self-consistency 85.3% at six samples): a second model must add *more than another sample's worth* of value — which it does when it holds different context (ground-truth tool results, the full event stream) rather than merely a different seed. Constitutional AI [17] shows judging rubrics can be explicit written principles, making supervision auditable; Transformer^2 [13] demonstrates the broader trend of models adapting *at inference time* rather than through retraining.

2.5 Thread 5: Prompts that improve from measured feedback

OPRO [18] treats prompt optimization as an LLM proposing candidates from a trajectory of scored attempts (GSM8K 71.8% $\rightarrow$ 80.2% over the human baseline instruction). DSPy [19] compiles multi-stage LM pipelines against an end metric (GSM8K with GPT-3.5: 25.2% $\rightarrow$ 81.6% test accuracy). TextGrad [20] backpropagates natural-language criticism as "textual gradients," with a directly relevant configuration: GPT-4o feedback lifting a *weaker* GPT-3.5 target from 72.9% to 81.1% on GSM8K — a stronger model's language measurably improving a weaker model's behavior, which is the whisper payload in miniature. ProTeGi [21] contributes the anti-gradient edit loop (up to +31% over seed prompts) and an operationally decisive detail: wall-clock can exceed an hour, confirming that prompt optimization belongs *between* calls, not inside one.

2.6 Synthesis

No single prior work assembles these components for real-time voice, and none addresses the substrate problem that dominates deployment, to which we now turn.

Design principle	Load-bearing evidence
Separate the talker from the deliberator	Talker-Reasoner [1]; dual-process framing [2]
Supervision must be external	Intrinsic self-correction degrades (75.9 → 74.7 GSM8K; 75.8→38.1 CSQA) while oracle-triggered correction gains (→84.3) [3]
The supervisor can be small	6B ≈ 30× scale [6]; 7B critic ≈ ChatGPT [9]; monitor recall 95% [8]; 7B teacher > 671B-derived data [11]
Supervise turns, not outcomes	PRM 78.2% vs. ORM 72.4%, better OOD [7]
Never optimize against the monitor	Obfuscated reward hacking; recall → 0 [8]; verifier gaming past 400 samples [6]
Improve prompts offline, from scores	OPRO [18], DSPy [19], TextGrad [20], ProTeGi [21], DGM [12]

Table 1: Five threads, one architecture: the evidence base for the sidecar oracle.

3 The Runtime Harness

3.1 The substrate problem

Voice-AI serving stacks fall into three classes with materially different observability and control surfaces:

- **Speech-to-speech (S2S) agents** (OpenAI Realtime, xAI Grok Voice, Ultravox): audio-native models over a realtime socket. There is no out-of-band message channel; the only injection primitive is a live session/prompt patch.
- **Pipeline agents** (Vapi, Retell, ElevenLabs Agents, Deepgram Voice Agent, Bland): STT→LLM→TTS loops. Transcripts stream as typed events; injection lands in the LLM context between turns — when the platform exposes a channel at all (Bland documents none and is honestly tap-only).
- **Frameworks and components** (Pipecat, LiveKit Agents; Cartesia, Inworld, raw Deepgram STT): the integrator owns the loop (inject a context frame) or the component is a bare speech engine (nothing to inject into; observation only).

Every class also differs in event vocabulary — `ConversationText` vs. `user_transcript` vs. nested webhook envelopes vs. raw streaming-STT `Results` — and in wire details that drift (OpenAI's GA renamed its transcript events; Vapi deprecated `function-call` for `tool-calls`). A supervisor written against any one platform is rewritten every quarter. This heterogeneity, not the oracle, is the engineering problem.

3.2 Canonical events and the deterministic reducer

The harness defines a small canonical event vocabulary

$\mathcal{E}$ = {`session.{started,updated,ended}`, `caller.transcript.{partial,final}`, `agent.transcript.{part`

and, per provider p, an adapter supplying two pure functions:

$$\text{parse}_p : \text{Raw}_p \to \mathcal{E}^* \qquad \text{compile}_p : \mathcal{D} \times \Sigma \to \mathcal{A}_p$$

where $\mathcal{D}$ is a provider-independent decision vocabulary (principally `inject_hidden_instruction`) and $\mathcal{A}_p$ the provider's native actions. A deterministic reducer $\rho : \Sigma \times \mathcal{E} \to \Sigma$ folds events into runtime state Σ — the transcript with per-turn language tags, tool history, and, decisively, *truth sub-state*: whether a requested booking was ever verified by a tool result, whether a promised delivery actually sent, whether end-of-call claims are backed. ρ contains no model call; Σ is ground truth by construction and replayable from the event log. This is what gives the supervisor the external signal that Thread 2 shows it must have, and what later grades calls without an LLM in the loop (§4).

3.3 Capability-aware injection compilation

Adapters declare a capability vector (hidden-instruction injection, mid-call prompt patch, tool interception, transcript streaming, ...), and the compilation of the single abstract decision honors it:

Platform	**Class**	**Compiled whisper**
Ultravox	S2S	`inject_message` (deferred text)
OpenAI Realtime / Grok Voice	S2S	`session.update` instruction patch
Vapi	pipeline	`assistant_override` in webhook reply
Retell	pipeline (custom LLM)	system message prepended to next turn
ElevenLabs Agents	pipeline	`contextual_update` (read, never spoken)
Deepgram Voice Agent	pipeline	`UpdatePrompt` (additive)
Pipecat	framework	`LLMMessagesAppendFrame`, `run_llm=false`
LiveKit Agents	framework	chat-context system append
Bland / Cartesia / Inworld	pipeline / component	*none* — tap-only, honestly declared
Generic webhook	any	configurable action

Table 2: One decision, thirteen native channels. Adapters that cannot inject say so in their capability vector rather than pretending.

The same abstraction covers the model and speech layers: oracle models are prefix-routed with live catalog discovery (so newly released models require no harness change), and the transcription tap runs multilingual streaming STT (nova-3 code-switching, per-utterance language tags) whenever a stack provides audio but not transcripts. One rule prevents the pathological combinations: *exactly one transcript source per call*, resolved by a decision procedure over the provider's profile (does it stream transcripts? are they language-tagged? is raw audio forkable?).

3.4 The whisper path and degrade-safety

Off the hot path, the oracle maintains a belief state $B(\Sigma)$ (identity, intent, emotional state, language, urgency) and produces a directive $d = D(B, \Sigma, G)$ under operator guardrails G, gated by a confidence threshold τ: only $\mathrm{conf}(d) \geq \tau$ compiles to an injection. The entire oracle runs behind a timeout with catch-all semantics:

$$\mathrm{fail}(\mathrm{oracle}) \;\Rightarrow\; d = \varnothing \;\Rightarrow\; \mathrm{compile}_p(\varnothing, \Sigma) = \epsilon,$$

i.e., supervisor failure composes with the live call as the identity. A stalled model, a dead STT socket, or a failed TTS backend cannot lengthen, alter, or end the call it shadows. This property is tested, not aspirational, and it is what makes the sidecar deployable on revenue-bearing traffic: the worst case is the status quo.

4 The Improvement Loop

4.1 Grading calls against ground truth

Because Σ carries verified outcomes, calls are scored deterministically at session end: unverified bookings, unverified sends, unbacked end-of-call claims, delivery errors, and dead-air events each subtract from a unit score, and the caller's language and turn statistics are attached. No LLM grades production calls; the reward is measured, in the spirit of the Darwin Gödel Machine's empirical fitness [12] and in contrast to judge-only pipelines.

4.2 The standing directive: optimizing the layer the harness owns

Operators' system prompts live in dashboards, code templates, and database rows the harness neither sees nor should modify. The harness *does* own one prompt surface on every injectable platform: its own channel. The *standing directive* is a persistent, versioned coaching preamble (bounded to a few imperative lines) injected at call start through that channel. An improvement step feeds the recent scored reports and the current directive to a critic model, OPRO-style [18], yielding a new version with a stated rationale; versions are retained only alongside their evidence. Two constraints from the meta-analysis are engineered in: the loop runs offline between calls (ProTeGi's latency lesson [21]), and its output is *injected context* for the agent, never a training signal — honoring the obfuscation result [8] and Cobbe's verifier-gaming caution by capping optimization pressure. The operator's base prompt is untouched by design; prompt-diff *suggestions* against fingerprinted operator prompts are future work.

4.3 Observability as a product invariant

Every directive the oracle ever whispers is logged with its confidence, language, the caller's inferred emotional state, oracle latency, model, and billing cost, and every optimization step records the reports it consumed. A system that whispers into production calls must be auditable end-to-end; this is the practical analogue of the monitorability argument in [8].

5 Limitations

We are explicit about what the evidence does and does not establish. (1) No controlled trial yet measures the sidecar's effect on *voice-call business outcomes*; the meta-analysis validates components (external verification, small critics, feedback-driven prompt gains) in adjacent domains, and Talker-Reasoner itself is a qualitative case study. A randomized supervised-vs-unsupervised call study is the natural next experiment, and the harness's per-call scoring makes it cheap to run. (2) The builder-mode grader uses an LLM; deterministic scoring applies only where tool-verified ground truth exists. (3) Whisper uptake is bounded by each platform's injection semantics — an S2S prompt patch is weaker than a pipeline context message, and two platforms offer no channel at all. (4) Standing-directive optimization inherits the brittleness OPRO documents; we bound directive length and retain version history precisely because regressions are expected. (5) The oracle adds cost per turn; the small-supervisor results (Thread 3) are what make the economics work, and metering makes them visible.

6 Conclusion

Read together, the literature is unusually convergent: fast models need slow supervisors; supervision must be external; a small second model catches what the first cannot; its judgments must be injected, not optimized against; and prompt layers should improve offline from measured outcomes. What stood between that consensus and production voice AI was a substrate problem — thirteen platforms, three architectures, drifting wire formats, and wildly uneven control channels. A canonical event algebra, a deterministic ground-truth reducer, capability-aware compilation, and no-op failure semantics close that gap. The supervisor rides beside the call. It does not drive, and it cannot crash the bike.

Artifacts: MIT-licensed implementation, adapters for all platforms in Table 2, live multilingual tap, hosted control plane with per-whisper audit logs, and an interactive builder demonstrating the full loop — `github.com/samthedataman/secondmind`.

References

[1] K. Christakopoulou, S. Mourad, M. Matarić. *Agents Thinking Fast and Slow: A Talker-Reasoner Architecture*. arXiv:2410.08328, 2024.

[2] G. Booch et al. *Thinking Fast and Slow in AI*. arXiv:2010.06002, 2020.

[3] J. Huang et al. *Large Language Models Cannot Self-Correct Reasoning Yet*. arXiv:2310.01798, 2023 (ICLR 2024).

[4] A. Madaan et al. *Self-Refine: Iterative Refinement with Self-Feedback*. arXiv:2303.17651, 2023.

[5] N. Shinn et al. *Reflexion: Language Agents with Verbal Reinforcement Learning*. arXiv:2303.11366, 2023.

[6] K. Cobbe et al. *Training Verifiers to Solve Math Word Problems*. arXiv:2110.14168, 2021.

[7] H. Lightman et al. *Let's Verify Step by Step*. arXiv:2305.20050, 2023.

[8] B. Baker et al. *Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation*. arXiv:2503.11926, 2025.

[9] T. Wang et al. *Shepherd: A Critic for Language Model Generation*. arXiv:2308.04592, 2023.

[10] Y. Inoue et al. *Wider or Deeper? Scaling LLM Inference-Time Compute with Adaptive Branching Tree Search*. arXiv:2503.04412, 2025.

[11] E. Cetin et al. *Reinforcement Learning Teachers of Test Time Scaling*. arXiv:2506.08388, 2025.

[12] J. Zhang, S. Hu, C. Lu, R. Lange, J. Clune. *Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents*. arXiv:2505.22954, 2025.

[13] Q. Sun, E. Cetin, Y. Tang. *Transformer^2: Self-Adaptive LLMs*. arXiv:2501.06252, 2025.

[14] C. Lu et al. *The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery*. arXiv:2408.06292, 2024; v2: arXiv:2504.08066, 2025.

[15] Y. Du et al. *Improving Factuality and Reasoning in Language Models through Multiagent Debate*. arXiv:2305.14325, 2023.

[16] G. Irving, P. Christiano, D. Amodei. *AI Safety via Debate*. arXiv:1805.00899, 2018.

[17] Y. Bai et al. *Constitutional AI: Harmlessness from AI Feedback*. arXiv:2212.08073, 2022.

[18] C. Yang et al. *Large Language Models as Optimizers*. arXiv:2309.03409, 2023.

[19] O. Khattab et al. *DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines*. arXiv:2310.03714, 2023.

[20] M. Yuksekgonul et al. *TextGrad: Automatic "Differentiation" via Text*. arXiv:2406.07496, 2024.

[21] R. Pryzant et al. *Automatic Prompt Optimization with "Gradient Descent" and Beam Search*. arXiv:2305.03495, 2023.

[22] T. Rebedea et al. *NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails*. arXiv:2310.10501, 2023.

[23] H. Inan et al. *Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations*. arXiv:2312.06674, 2023.