Why Our Autonomous AI Agent Quality Dropped 31 Percent

Vasyl Golubenko
TOV ZELTREX, Kyiv, Ukraine
ceo@zeltrex.com

March 2026

Abstract

Night Shift is an autonomous AI development system that runs 24/7, dispatching tasks to Claude, Gemini, and other LLMs during off-hours. Over its first 31 operational days, we observed average quality scores decline from 7.37/10 to 5.04/10 — a 31% drop — followed by a 4-day complete outage caused by a safety mechanism feedback loop. This article presents the root cause analysis, driven entirely by production data from 287+ task executions across 5 models, the 18 systemic fixes deployed across three phases, and the lessons learned from operating an autonomous AI development agent in production.

The findings challenge several intuitions: the biggest quality killer was not model capability — it was **output truncation** (costing 2.7 quality points per task). The apparent "decline" was partly an **assessor calibration shift** as our LLM-as-Judge activated. Our mentoring feedback loop had **0% measurable effectiveness** despite 88 tracked interventions. And our emergency brake safety mechanism, designed to prevent low-quality output, caused a worse outage than the problem it was designed to solve.

Keywords:

1 Introduction

Autonomous AI development agents — systems that generate code, reports, and research without continuous human oversight — are an emerging paradigm in software engineering [**?**, **?**]. These systems promise 24/7 productivity but introduce novel failure modes that differ fundamentally from supervised AI usage. When a human developer uses an LLM, they can immediately evaluate output quality and retry. When an autonomous agent operates overnight, quality degradation can compound undetected across dozens of tasks.

This paper presents a longitudinal case study of Night Shift, an autonomous development agent deployed in production for 31 days (February 24 to March 24, 2026). We report six findings from analyzing 287+ task executions, document 18 systemic fixes informed by recent literature, and discuss the implications for building reliable autonomous AI systems.

2 System Architecture

Night Shift operates as an autonomous development agent:

- **Dispatch cycle:** Every 2 hours, selects the highest-priority task from a 170-item backlog
- **Budget governor:** 3.5M tokens/week across Anthropic, Google, and open-source providers
- **Quality assessor:** Hybrid heuristic + LLM-as-Judge scoring (1-10 scale)
- **Mentoring loop:** Human reviews injected into system prompts as context
- **Self-improvement:** Follow-up tasks auto-generated from completed work

3 Data

Table 1: Task execution summary by period.

287 assessed task executions over 10 days (Feb 24 — Mar 5, 2026):

Period	Tasks	Avg Quality	Good (≥$7)	Bad (≤$3)	Truncated
Week 1 (Feb 24-28)	169	7.1	106 (63%)	15 (9%)	49 (29%)
Week 2 (Mar 1-5)	118	5.7	56 (47%)	22 (19%)	33 (28%)

Table 1: Task execution summary by period

4 Findings

4.1 Truncation is the Primary Quality Determinant

The single strongest predictor of quality is whether the output was truncated (Table 2).

Output	Tasks	Avg Quality	Delta
Full	198	**7.3**	—
Truncated	89	**4.6**	**-2.7**

Table 2: Quality impact of output truncation.

Categories with highest truncation had lowest quality:

- LIVINGCORP: 64% truncated → q=5.5
- RESEARCH: 47% truncated → q=5.9
- BRIDGE: 11% truncated → q=7.3

Root cause: Fixed output token limits did not account for task complexity. Research tasks naturally produce longer outputs but were given the same token budget as simple code fixes. The continuation mechanism (requesting the model to continue) did not help — tasks with 1 continuation averaged q=5.1, worse than tasks with 0 continuations (q=7.4).

Insight: It's better to scope a task smaller than to truncate a larger one. Truncated output is almost always worse than complete-but-shorter output.

4.2 What the Literature Says About Truncation

Two recent papers validate our finding. **SelfBudgeter** (arXiv 2505.11274) shows that letting models self-estimate their token budget achieves 61% response compression with maintained accuracy. **TALE** (arXiv 2412.18547) demonstrates 68.64% token reduction with <5% accuracy loss through task-complexity-aware budgets. The core insight from both: *"the optimal token budget is not fixed but varies depending on the complexity of the problem."* Our fixed `max_tokens` was the root cause — not model capability.

Devin (Cognition) solves this differently: time-bounded rather than token-bounded execution, letting the agent decide when to terminate. CrewAI explicitly acknowledges truncated outputs as an unsolved problem. LangGraph accumulates growing state history, reaching 15K+ tokens in complex tasks — the problem is universal.

Our fix (deployed): Disabled continuations entirely (`MAX_CONTINUATIONS=0`), added task-aware output budget estimation that adjusts tokens based on output type, category, and prompt complexity (4K-20K range), and added category-specific token overrides in the dispatcher. When a task would need continuation, the system now scopes it smaller upfront.

4.3 Assessor Calibration Shift

Week 1 had an extreme score distribution: 48% of all tasks scored exactly 8/10. Week 2 had a more normal distribution across 4-8.

The cause: our LLM-as-Judge (a secondary scorer using Haiku) was introduced mid-Week 1 with a 0.6/0.4 blend (60% heuristic, 40% LLM). The heuristic scorer had a "happy path" that defaulted to 8 for any non-truncated output with basic structure. The LLM judge corrected these inflated scores downward.

Insight: When measuring quality trends in autonomous systems, the measurement instrument itself can shift. Without calibration anchors (known-good and known-bad reference outputs), you cannot distinguish "quality declined" from "scoring became more accurate."

4.4 What the Literature Says About LLM-as-Judge

MT-Bench (Zheng et al., NeurIPS 2023) is the foundational work. Key finding: few-shot calibration anchors improve scoring consistency from 65.0% to 77.5% for GPT-4. All LLM judges exhibit position bias, and prompt sensitivity varies by model — the same scoring prompt can increase consistency for one model but decrease it for another.

Our heuristic scorer exhibited classic **Goodhart's Law** — "when a measure becomes a target, it ceases to be a good measure." The heuristic rewarded structure (headings, bullets, line count) as a proxy for quality. Research on RLHF reward model overoptimization (Gao et al., ICML 2023) confirms this is fundamental: increasing optimization against a given reward model eventually *decreases* actual quality.

AlpacaEval addresses calibration drift through length-controlled evaluation — penalizing models that score higher simply by producing longer outputs. This is exactly our problem: longer, well-structured outputs scored higher regardless of substance.

Our fix (deployed): Inverted to 40% heuristic / 60% LLM-as-Judge, removed the score

5-8 gate so the judge always runs, added calibration anchors (`anchors.yaml` with 6 reference outputs across 3 output types), and enabled dual score logging to track heuristic vs LLM scores independently.

4.5 Context Bloat Degrades Output

Opus tasks on March 4 averaged 138,000 tokens of context — and produced quality of 4.7/10. The same model on Feb 25 with 91,000 tokens scored 8.7/10.

Context Size	Opus Quality
75K tokens	7.0
91K tokens	8.7
95K tokens	5.2
139K tokens	**4.7**

Table 3: Context size vs. Opus quality

The context engine loaded everything available: full codebase maps, two complete mentoring reviews (2000 chars each), dependency outputs from parent tasks, and the task prompt itself. Without a budget, context accumulated over time as more mentoring reviews and more features were added.

Insight: More context $\neq$ better output. There's an optimal context window for each model, beyond which the model struggles to find the relevant signal in the noise.

4.6 What the Literature Says About Context Bloat

The **"Lost in the Middle"** paper (Liu et al., TACL 2024) showed that LLM performance drops by more than 30% when relevant information shifts to the middle of the input context. In 20- and 30-document settings, performance can be *lower than having no input documents at all* — meaning context actively hurts.

Chroma Research (2025) tested all 18 frontier models and found every single one exhibits "context rot" — even with 100% perfect retrieval, performance degrades 13.9% to 85% as input length increases. The causes: (1) RoPE-based attention bias toward beginning/end tokens, (2) quadratic attention scaling (n^2 pairwise relationships), (3) semantically similar distractors interfering with relevance identification.

Anthropic's engineering blog (2025) states it directly: *"Good context engineering means finding the smallest possible set of high-signal tokens that maximize the likelihood of some desired outcome."*

Synthesizing the research, optimal working ranges are: Claude Opus/Sonnet 40K-80K, Claude Haiku 20K-40K, GPT-4 <64K — far below the raw context window sizes. Night Shift's Opus tasks at 138K tokens were operating at nearly 2x the optimal range.

Our fix (deployed): Hard per-model context budgets (Opus 80K, Sonnet 60K, Haiku 30K), U-shaped position-aware placement (critical instructions at top and bottom, supplementary material in the middle where attention is weakest), and smart dependency compaction that

extracts code blocks, headings, and conclusions — reducing `MAX_DEPENDENCY_CHARS` from 30K to 8K (73% reduction).

4.7 Auto-Generated Follow-Ups Are Lower Quality

Night Shift auto-generates follow-up tasks: research → implementation, code → tests. These follow-ups:

Task Origin	Week 1 Quality	Week 2 Quality	Truncation
Original (human-written)	7.5	6.0	31%
Auto-generated	6.8	**5.1**	40%
— write-tests	—	—	**39%**
— implement-findings	—	—	31%

Table 4: Follow-up task quality comparison

`write-tests` tasks were particularly problematic: they tested the LLM's own generated code with no access to the real codebase, producing tests that validated mock implementations.

Insight: Autonomous follow-up generation needs quality gates. If the parent task was poor (truncated, low score), the follow-up will be worse. Gate on parent quality ≥ 7 before spawning children.

4.8 The Mathematics of Cascade Failure

Research on multi-agent reliability reveals a fundamental problem: **the 0.95^n compound error effect.** If each step in an agent workflow has 95% reliability, over 20 steps this yields only 36% success rate. A Towards Data Science analysis found that unvalidated "bag of agents" approaches create up to **17.2x error amplification**.

OWASP ASI08 (2026) classifies cascading failures as a top security risk in agentic AI: "dependent agents exponentially amplify load on downstream systems." Microsoft Azure's agent orchestration patterns recommend output validation at every handoff — exactly what Night Shift was missing.

Our fix (deployed): Quality gate (parent score ≥ 7 before spawning children), truncation gate (never generate follow-ups from truncated parents), `MAX_FOLLOWUP_DEPTH` reduced from 2 to 1, `write-tests` killed for non-code categories, and a new parent output validation step that checks for minimum length, code block presence (for code tasks), and heading structure (for reports).

4.9 The Mentoring Loop Was Decorative

We tracked 88 mentoring interventions across 8 review sessions. The context engine injected the 2 most recent reviews into every task's system prompt. Measured effectiveness: **0%**.

Why it failed:

1. **Generic injection:** Same reviews given to ALL tasks regardless of category

2. **Too long:** 2000 chars per review, mostly grade tables and infrastructure notes — models couldn't extract actionable guidance

3. **Wrong audience:** Interventions like "deploy depth limit" are system-level recommendations the model cannot implement

4. **No targeting:** A BRIDGE task got RESEARCH feedback; a code task got report feedback

Insight: Mentoring feedback for autonomous agents must be (a) category-specific, (b) concise (<500 chars), (c) model-actionable ("always include a methodology section"), and (d) measured at the category level, not globally.

4.10 What the Literature Says About Agent Feedback Loops

Reflexion (Shinn et al., NeurIPS 2023) achieved 91% pass@1 on HumanEval (up from GPT-4's 80%) using verbal reinforcement learning — the model converts feedback into natural language descriptions of what went wrong. The key: feedback is per-task, verbal, stored in episodic memory, and generated from specific failure analysis. Night Shift's generic 2000-char reviews violated every one of these principles.

TextGrad (Yuksekgonul et al., Nature) treats AI systems as computation graphs where textual feedback serves as gradients for optimization. Each variable receives feedback *specific to itself*, not generic system-level observations.

OpenAI's Self-Evolving Agents Cookbook (2025) emphasizes that "agentic systems often reach a plateau after proof-of-concept because they depend on humans to diagnose edge cases and correct failures" — exactly Night Shift's situation.

Our fix (deployed): Per-category guidance files (max 500 chars), context engine loads only the matching category's guidance, Reflexion-style per-task reflections stored in `mentoring/reflections/{category}.jsonl` (last 20 per category), and feedback written as model-actionable instructions ("always include a methodology section") not system recommendations ("deploy depth limit"). Additionally, category budget caps (30% max per category per week) prevent any single category from dominating dispatch.

5 Interventions

All 15 fixes were deployed to production over two days (2026-03-08 to 2026-03-09). We organized them into two phases: Phase 1 addressed the 5 root causes directly, Phase 2 implemented deeper infrastructure changes informed by the literature.

5.1 Phase 1: Root Cause Fixes (6 interventions, deployed 2026-03-08)

5.2 Phase 2: Literature-Informed Infrastructure (10 interventions, deployed 2026-03-09)

5.3 Key Implementation: Calibration Anchors (Fix 7)

The literature showed that few-shot calibration anchors improve LLM judge consistency from 65% to 77.5% (MT-Bench). We created `mentoring/calibration/anchors.yaml` with reference

#	Fix	File	Impact
1	Assessor weight inversion (40/60)	quality_assessor.py	Fixes score inflation
2	Category-specific token overrides	dispatcher.py	Eliminates truncation
3	Task generator quality gates	task_generator.py	Prevents cascades
4	Per-category guidance (500 chars)	context_engine.py	Replaces 0%-effective reviews
5	Task-aware budget allocation	quota_governor.py	Prevents saturation
6	MAX_FOLLOWUP_DEPTH 2→1	task_generator.py	Limits error chains

Table 5: Phase 1: Root Cause Fixes (deployed 2026-03-08)

#	Fix	File	Research Basis
7	Calibration anchors for LLM judge	assessor + anchors.yaml	MT-Bench: +12.5%
8	Dual score logging	learning.py	Evidently AI
9	Category budget caps (30%)	quota_governor.py	Prevents saturation
10	Continuations disabled	dispatcher.py	q=5.1→7.4
11	Reflexion-style reflections	dispatcher + context_engine	Reflexion: 91% HumanEval
12	U-shaped context placement	context_engine.py	Lost in the Middle
13	Monthly recalibration template	human_grades.yaml	AlpacaEval
14	Compact dependency output	context_engine.py	Context rot: 13.9-85%
15	Parent output validation	task_generator.py	OWASP ASI08
16	Task-aware output budget	dispatcher.py	SelfBudgeter/TALE

Table 6: Phase 2: Literature-Informed Infrastructure (deployed 2026-03-09)

outputs for 3 output types (code, report, research), each with a score=9 (exemplary) and score=3 (poor) example:

[Code implementation available in the source repository.]

The anchors provide the judge with concrete reference points: "this is what a 9/10 code output looks like" and "this is what a 3/10 looks like." Without anchors, the judge drifts toward whatever scoring baseline it internalized during training.

5.4 Key Implementation: Reflexion-Style Feedback (Fix 11)

Inspired by Reflexion's per-task verbal reinforcement, we store category-specific reflections after each task:

[Code implementation available in the source repository.]

The context engine then loads the 3 most recent reflections for the matching category into the prompt — targeted, concise, and model-actionable, unlike the previous 2000-char generic reviews.

5.5 Key Implementation: U-Shaped Context Placement (Fix 12)

The "Lost in the Middle" paper showed 30%+ performance drop for information placed in the middle of context. We reorganized `build_prompt()`:

[Code implementation available in the source repository.]

Critical instructions that the model must follow go at the beginning and end. Supplementary reference material goes in the middle where attention is weakest anyway.

5.6 Key Implementation: Compact Dependencies (Fix 14)

Instead of passing raw parent output (up to 30K chars), we extract high-signal content:

[Code implementation available in the source repository.]

This reduced `MAX_DEPENDENCY_CHARS` from 30,000 to 8,000 — a 73% reduction in dependency context while preserving the highest-signal content. The research shows this should improve quality because less noise means better signal extraction.

6 Discussion: Precision Over Volume

Across all five findings and fifteen fixes, the same pattern emerges: **more is not better.**

- More output tokens without budget management → truncation → **Fix: task-aware budgets (SelfBudgeter/TALE)**
- More scoring criteria without calibration → inflated scores → **Fix: calibration anchors (MT-Bench)**
- More mentoring feedback without targeting → 0% effectiveness → **Fix: per-task reflections (Reflexion)**
- More context without budgeting → quality degradation → **Fix: U-shaped placement + compaction (Lost in the Middle)**
- More follow-up tasks without quality gates → error amplification → **Fix: parent validation + depth limits (OWASP ASI08)**

The research literature converges on the same conclusion. Anthropic's context engineering guide recommends "the smallest possible set of high-signal tokens." MT-Bench shows calibration anchors matter more than scoring sophistication. Reflexion demonstrates that per-task verbal feedback beats generic reviews. The 0.95^n compound error effect means that quality gates between stages aren't optional — they're the only thing preventing cascade failures.

Every fix we deployed follows this principle: replace volume with precision. Fewer tokens in context, but better-placed. Fewer follow-up tasks, but higher-quality parents. Fewer scoring dimensions, but better-calibrated. The 15 interventions collectively target an estimated +3.5 quality points (from the current 5.04 baseline).

7 Implications for Autonomous AI Systems

For anyone building autonomous AI agents that operate continuously:

1. **Measure the measurer.** Quality assessment tools need their own calibration. Without known-good/bad anchors, you're measuring noise. Goodhart's Law applies: when your heuristic becomes the optimization target, it stops measuring quality (Gao et al., ICML 2023).

2. **Truncation > capability.** The limiting factor was not model intelligence — it was output buffer management. SelfBudgeter and TALE show that dynamic, task-aware token budgets solve this without losing accuracy. This is an infrastructure problem, not an AI problem.

3. **Feedback loops require precision.** Generic mentoring is worse than no mentoring (wastes context tokens). Reflexion (NeurIPS 2023) proved that per-task, verbal, episodic feedback achieves 91% HumanEval pass@1. Category-specific, concise, model-actionable feedback is the only kind that works.

4. **Auto-generated work amplifies problems.** The 0.95^n compound effect means 95% per-step reliability yields only 36% over 20 steps. If your system generates follow-up tasks, quality gates at every handoff are mandatory. Poor parents produce poor children — up to 17.2x error amplification without validation.

5. **Context is a resource, not a free lunch.** Every frontier model exhibits "context rot" (Chroma Research, 2025). Even with perfect retrieval, performance degrades 13.9-85% as input length increases. The optimal working range is 40-80K tokens for Claude models — far below the 200K context window.

8 Post-Fix Results: Day 14-31

With all 15 fixes deployed (March 9), we monitored the system for 22 days. The results were mixed — the fixes worked when the system was running, but exposed a new class of failure.

8.1 Day 14-20: Partial Recovery

Period	Tasks	Avg Quality	Good (>=7)	Truncated	Cost
Pre-fix (Day 1-13)	287	5.04	47%	48%	$52.32
Post-fix (Day 14-20)	19	5.0	26%	71%	$13.66

Table 7: Post-fix results: Day 14–20

The Day 20 dispatch cycle (19 tasks in one burst) showed:

- **Sonnet:** 2 tasks, avg 7.65, 0% truncated — A to B- range
- **Haiku:** 3 tasks, avg 6.07, 0% truncated — B to C range
- **Gemini Flash:** 14 tasks, avg 4.56, **71% truncated** — B- to F range

The free model output limit fix (65536 tokens) was deployed but had not yet been pulled on the server at this point. The 74% routing to Flash (cheapest model) meant most tasks still hit the old limits.

8.2 Day 20-31: The Second Outage (4-Day Brake Lock)

Quality brake triggered at avg 3.97 $<$ 4.5. Auto-resume was configured (2h) but a **stale-score bug** caused re-triggering:

1. Brake paused dispatcher
2. After 2h, auto-resume activated
3. Post-cycle monitor checked quality scores — same stale scores from before the pause
4. Brake re-triggered immediately
5. Loop continued for 4 days with 0 tasks dispatched

Root cause: The brake counted quality scores without checking whether they were from tasks completed AFTER the resume. With no new tasks between resume and check, the same low scores kept triggering the brake.

Fix deployed (Day 31): Added a task-count guard — the brake only re-evaluates after at least `lookback` (10) new tasks have completed since the last resume.

8.3 Finding 6: Safety Mechanisms Need Escape Hatches

This is a new finding from the Day 20-31 data. The emergency brake was a correct response to quality degradation, but its implementation created a worse outcome than the problem it was designed to solve:

Scenario	Duration	Tasks Lost	Recovery
Quality degradation (no brake)	Ongoing	None (low-quality output)	Self-correcting via evolution
Quality brake (with bug)	**4 days**	**72 tasks** (18/day $\times$ 4)	Required manual intervention

Table 8: Quality brake impact comparison

The brake's cure was worse than the disease. This mirrors the "overzealous safety mechanism" pattern documented in OWASP ASI08: a well-intentioned guardrail that creates cascading failures worse than the original risk.

8.4 Additional Fixes (Day 31)

Beyond the brake guard, we deployed:

1. **Evolution plateau escape:** When mutation rate hits ceiling (0.25) and plateau persists $>=$ 5 generations, reset to default (0.15) and force population restart with fresh seeds. Previously, the system stayed stuck at maximum mutation rate indefinitely.
2. **Quality-aware model routing:** P2 priority research/spec tasks now route to Haiku (avg 6.07) instead of Gemini Flash (avg 4.56). Only P3+ low-priority tasks use free cloud. This reduces Flash's share from 74% to ~40%.

Metric	Day 1	Day 13	Day 20	Day 31
Avg quality	7.37	5.04	5.0	— (idle)
Truncation rate	29%	28%	71%	Fix deployed
Auto-merge rate	0%	0%	0%	0%
Hall of Fame entries	0	0	0	0
Evolution generation	1	13	20	27 (stagnated)
Mutation success rate	—	—	2.3%	Fix deployed
Total cost	$0	$52.32	$65.98	$65.98

Table 9: Revised key metrics (Day 1–31)

8.5 Revised Key Metrics (Day 31)

9 Conclusion

We presented a 31-day longitudinal case study of Night Shift, an autonomous AI development agent in production. Our analysis of 287+ task executions revealed six findings: (1) output truncation is the primary quality determinant, costing 2.7 points per task; (2) LLM-as-Judge calibration shift can masquerade as quality decline; (3) context bloat degrades output beyond model-specific thresholds; (4) auto-generated follow-up tasks amplify errors through the 0.95^n compound effect; (5) generic mentoring feedback has 0% measurable effectiveness; and (6) safety mechanisms without escape hatches cause worse outages than the failures they prevent.

The 18 interventions deployed across three phases follow a unifying principle: precision over volume. Fewer tokens with better placement, fewer tasks with higher-quality parents, and fewer scoring dimensions with better calibration consistently outperformed their high-volume counterparts.

The core lesson from 31 days of production operation: **autonomous AI systems require escape hatches at every safety layer**, and evaluation data staleness must be explicitly managed to prevent feedback loops between safety mechanisms and the metrics they monitor.

10 Data and Code Availability

The Night Shift codebase is proprietary to TOV ZELTREX (EDRPOU 46125819). Aggregated metrics, anonymized task data, and configuration files referenced in this paper are available upon request to the corresponding author. The quality assessment framework, calibration anchors, and model routing logic are described in sufficient detail for independent reimplementation.

References

[1] Liu, N.F. et al. (2024). "Lost in the Middle: How Language Models Use Long Contexts." *Transactions of the Association for Computational Linguistics*, 12, 157–173. arXiv:2307.03172

[2] Zheng, L. et al. (2023). "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena." *NeurIPS 2023*. arXiv:2306.05685

[3] Shinn, N. et al. (2023). "Reflexion: Language Agents with Verbal Reinforcement Learning." *NeurIPS 2023*. arXiv:2303.11366

[4] Yuksekgonul, M. et al. (2024). "TextGrad: Automatic 'Differentiation' via Text." *Nature*. arXiv:2406.07496

[5] Gao, L. et al. (2023). "Scaling Laws for Reward Model Overoptimization." *ICML 2023*. arXiv:2210.10760

[6] Wei, J. et al. (2025). "SelfBudgeter: Adaptive Token Allocation for Efficient LLM Reasoning." arXiv:2505.11274

[7] Lv, K. et al. (2024). "TALE: Token-Budget-Aware LLM Reasoning." arXiv:2412.18547

[8] Chroma Research (2025). "Context Rot: How Increasing Input Tokens Impacts LLM Performance." Technical report.

[9] Anthropic (2025). "Effective Context Engineering for AI Agents." Engineering blog.

[10] OpenAI (2025). "Self-Evolving Agents Cookbook."

[11] OWASP (2026). "ASI08: Cascading Failures in Agentic AI." *OWASP Agentic Security Initiative*.

[12] Microsoft Azure (2025). "AI Agent Orchestration Patterns." Azure Architecture Center.

[13] Evidently AI (2025). "LLM-as-a-Judge: A Complete Guide to Using LLMs for Evaluation."

[14] Towards Data Science (2025). "Why Your Multi-Agent System is Failing: The 17x Error Trap."

[15] Li, X. et al. (2024). "AlpacaEval: An Automatic Evaluator of Instruction-Following Models."

[16] Golubenko, V. (2026). "Night Shift Day 31 Review: Stale-Score Brake Guard and Evolution Plateau Escape." Internal report, TOV ZELTREX.