SYSTEMS ARCHITECTURE & PRODUCTION AI ENGINEERING

# Production LLM Evaluation & Multi-Agent Operations

## The Glass-Box Playbook for Cost Routing, Rubric Gates & AI Reliability

| 88,000 | 52,167 | $17.12 | 72-Trap |
|---|---|---|---|
| HTTP 200 Wire Calls Audited | Multi-Agent Grading Sessions | 6-Rung Router (23,740 Calls) | 256GB Apple Silicon Benchmark |

**INSIDE THIS 112-PAGE PRACTITIONER PLAYBOOK**

Stage 1-3 Backward Design (UbD) Contracts • Out-of-Band Binary & Wire Truncation Gates • 4-Vendor Unanimous Blind Quorums (0.0% False-Pass Leak) • Counterfactual Negative-Control Rubrics • 40-Layer Apple Silicon MLX Logit-Lens Probes • Asymmetric 6-Rung FinOps Cost Routing • 5 Zero-Dependency Python 3.11 Harnesses • 6 Printable Audit Worksheets • 40-Question Scenario Exam

**Joshua Durey**

Principal AI Evaluation & Multi-Agent Operations Architect

First Print Edition • 8.5 x 11.0 in Technical Reference

EDITION METADATA & PROVENANCE

# Production LLM Evaluation & Multi-Agent Operations

> The Glass-Box Playbook for Cost Routing, Rubric Gates & AI Reliability

**Copyright © 2026 Joshua Durey. All rights reserved.**

Published independently by Joshua Durey. No part of this book may be reproduced in bulk commercial redistribution without written permission of the author, except for brief quotations in technical reviews and internal engineering team adoption of the standard-library verification patterns and audit worksheets herein.

**100% CLEAN-ROOM ENGINEERING & EMPIRICAL PROVENANCE**

Every Python 3.11 reference harness, JSON schema, equation, diagnostic exam scenario, and architecture worksheet in this volume was authored from scratch as independent clean-room intellectual property using Python standard-library primitives. Operational case-study metrics reference aggregate architectural lessons (including 88,000 HTTP 200 wire completions, 52,167 multi-agent grading sessions with 5,150 failure-to-rule fixes on a prior US learning contract, 23,740 six-rung cost-router calls, and 72 SHA-256 hashed local-vs-cloud benchmark runs on a 256GB Apple M5 Ultra workstation) without reproducing any third-party proprietary code, customer records, or confidential schemas.

**GLASS-BOX AI CONTENT & VERIFICATION DISCLOSURE**

In accordance with transparent Glass-Box engineering standards: reference code tables, structured formatting, and candidate diagnostic items were compiled from Joshua Durey's Stage-0 Understanding by Design (UbD) architecture blueprint with AI assistance (OpenAI GPT-4.1-mini and Google Gemini 2.5 Flash), verified via cold blind-stem solver models for 100% answer-key agreement and offline `stdin=DEVNULL` Python 3.11 execution (exit code 0), and edited by Joshua Durey.

**Trademarks & Liability Notice:** Python, OpenAI, Anthropic, Google Gemini, DeepSeek, Apple Silicon, and Mac Studio are trademarks of their respective owners, none of whom endorse or sponsor this independent handbook. All patterns and scripts are provided "as is" without warranty; validate thresholds in shadow mode before live cutover.

TABLE OF CONTENTS

# Architectural Contents (Pages 1-34)

## Front Matter, Part I (Backward Design Contracts) & Part II (Wire & Truncation Gates)


**FRONT MATTER**

| | |
|---|---|
| Title Page & Production Telemetry Summary | p. 1 |
| Copyright, Clean-Room Provenance & Glass-Box Disclosure | p. 2 |
| Detailed Table of Contents | p. 3 |
| Preface: When Green Dashboards Ship Broken Agents | p. 5 |
| System Architecture Reference Map & Reader Tracks | p. 6 |

**PART I: STAGE 1-3 BACKWARD DESIGN (UbD) FOR PRODUCTION AI SYSTEMS**

| | |
|---|---|
| Part I Charter: Backward Contracts Over Prompt Tinkering | p. 7 |
| 1.1 Why Forward-Prompted Multi-Agent Workflows Drift in Production | p. 8 |
| 1.2 Translating Wiggins & McTighe Backward Design into Systems Contracts | p. 9 |
| 2.1 Stage 1 Transfer Goals (TG1-TG6) & System Invariants (EU1-EU6) | p. 10 |
| 2.2 Essential Engineering Questions (EQ1-EQ6) & Execution Matrices | p. 11 |
| 2.3 Stage 2 GRASPS Evidence: Analytic Rubrics & Clause Hashing | p. 12 |
| 2.4 Annotated 00_UBD_SCOPE_AND_SEQUENCE.json Schema Contract | p. 13 |
| 3.1 Stage 3 WHERETO Execution Graphs & Fail-Closed Topology | p. 14 |
| 3.2 The 6-Stage Glass-Box Compiler Pipeline | p. 15 |
| 3.3 Out-of-Band Binary Anti-Gaming Verification (IHDR, CSV, MediaBox) | p. 16 |
| 3.4 CI/CD Permission Separation: Locking Graders Read-Only | p. 17 |
| 4.1 Case Study: Multi-Modal Artifact Compilation Under $0.60 Compute | p. 18 |
| 4.2 Part I Structural Failure-Mode & Remediation Catalog | p. 19 |
| 4.3 Part I Synthesis & Stage-0 Readiness Gate | p. 20 |

**PART II: TELEMETRY GATES & SILENT TRUNCATION DEFENSE (88,000-CALL AUDIT)**

| | |
|---|---|
| Part II Charter: Wire-Protocol Integrity Before Semantic Grading | p. 21 |
| 5.1 Inside the 88,000-Call HTTP 200 Wire Telemetry Audit | p. 22 |
| 5.2 Taxonomy of Silent HTTP 200 Failures & Truncation Modes | p. 23 |
| 5.3 Downstream Cascade Damage in Multi-Agent State Dictionaries | p. 24 |
| 6.1 Wire Gate 1: Cross-Provider finish_reason Whitelists | p. 25 |
| 6.2 Wire Gate 2: Token-Budget Headroom & Reasoning-Token Starvation | p. 26 |
| 6.3 Wire Gate 3: AST, JSON Schema & Structural Byte Pre-Screens | p. 27 |
| 6.4 Streaming SSE Integrity: Sequence Counters & Usage Reconciliation | p. 28 |
| 7.1 Multi-Turn Context Erosion & Tool-Loop State Corruption | p. 29 |
| 7.2 Deterministic JSON-RPC 2.0 MCP Tool Envelopes (-32700 to -32603) | p. 30 |
| 7.3 Request Idempotency Caching Against Socket Timeout Retries | p. 31 |
| 7.4 Compiling Recorded HAR Traffic into Zero-Socket AgentEnv Sandboxes | p. 32 |
| 8.1 Incident Runbook: Diagnosing Truncation Spikes Post-Upgrade | p. 33 |
| 8.2 Part II Wire-Defense Matrix & Pre-Production Gate Sign-Off | p. 34 |

PREFACE // FROM BLACK-BOX HOPE TO GLASS-BOX GATES

# When Green Dashboards Ship Broken Agents

> **Why standard HTTP 200 monitoring, single-model LLM judges, and unguarded retry loops fail in live production.**

Walk into almost any engineering organization shipping multi-agent software today and ask to see their AI reliability dashboard. Nine times out of ten, you will see green HTTP 200 status bars, a single frontier model grading its own outputs at 94% "helpfulness," and a cloud billing chart climbing three times faster than customer revenue.

Underneath those green indicators sit four silent failure modes that surface only when enterprise users complain or finance freezes cloud credentials:

| Failure Class | What Green Dashboards Report | What Actually Happened on the Wire |
|---|---|---|
| **1. Silent Truncation** | HTTP 200 OK \| Latency 1.8s | Token cap cut JSON in half (`length` / `rho=1.0`) |
| **2. Self-Preference** | 98.8% Overall Judge Accuracy | Solo judge leaked ~5% of broken answers from its own family |
| **3. Goodhart Gaming** | Rubric Pass Rate Jumped to 99% | Diluted rubric let bare 1-shot prompts tie 5-step agent DAGs |
| **4. Quadratic Retries** | "Self-Healing Agent Loop Active" | 1,800 dirty records triggered O(N^2) replay -> $28,593 bill |

This playbook replaces black-box prompt begging with deterministic Python 3.11 gates. You will wire Stage-0 contracts first, block truncated wire payloads at $0.00, calibrate 4-vendor blind quorums to 0.0% false-pass leakage, probe 40-layer local Apple Silicon activations for covert hint leaks, and route 23,740 calls across six FinOps rungs for $17.12.

> **HOW TO USE THIS HANDBOOK:** Read Parts I-IV with a highlighter, run the five stdlib harnesses in Part V offline, print Worksheets 1-6 for deployment reviews, and test your pod against the 40-question exam in Part VI.

CHAPTER 2.4 // ANNOTATED STAGE-0 JSON CONTRACT

# Complete Annotated 00_UBD_SCOPE_AND_SEQUENCE.json Contract

> **Storing backward-design contracts as strict JSON lets Python pre-flight scripts verify structural completeness, alignment coverage, and clean-room hygiene before build execution.**

```
{
  "framework": "Understanding by Design (3-Stage Backward Systems Contract)",
  "stage_1_desired_results": {
    "transfer_goals": [
      {"id": "TG1", "module": 1, "statement": "Eliminate false passes via 4-vendor blind
consensus."},
      {"id": "TG4", "module": 4, "statement": "Route multi-agent traffic fail-closed under $18/24k
calls."}
    ],
    "enduring_understandings": [
      {"id": "EU1", "module": 1, "statement": "Single-family judges share blind spots (~5% leak)."},
      {"id": "EU4", "module": 4, "statement": "Early rungs may auto-pass provably clean invariants
only."}
    ]
  },
  "stage_2_acceptable_evidence": {
    "grasps_performance_tasks": [{
      "module": 1,
      "goal": "Achieve 0.0% false-pass rate across 10,000 benchmark items with 5% planted defects.",
      "product_performance": "Deterministic JSON receipt with exit_code == 0 and quorum_leak ==
0.0.",
      "standards_analytic_rubric": {
        "exemplary_4": "0.0% false-pass + 150-repeat flip log + cross-vendor provenance stripped.",
        "proficient_3": "Reduces false-pass below 0.5% with partial provenance stripping.",
        "developing_2": "Single-vendor majority voting; correlated blind spots remain.",
        "novice_1": "Unverified single-shot self-grading."
      },
      "stage1_alignment": ["TG1", "EU1", "EQ1"]
    }]
  }
}
```

Notice how exemplary_4 states concrete testable artifacts rather than subjective praise. When a Python build runner loads this file at Step 1, it checks that every TG code in Stage 1 is covered by at least one GRASPS task and one exam domain in Stage 2, halting immediately if coverage gaps exist.

> **SCHEMA VERSIONING:** Commit 00_UBD_SCOPE_AND_SEQUENCE.json to git alongside a SHA-256 manifest digest so every production release links back to the exact contract version that governed it.

CHAPTER 6.1 // WIRE GATE 1: FINISH_REASON WHITELISTS

# Wire Gate 1: Cross-Provider finish_reason Whitelists

> **Multi-vendor routers cannot hardcode OpenAI's exact string values; normalize finish reasons across OpenAI, Anthropic, Google, and OpenRouter into one canonical enum.**

Every model provider names its termination states differently. OpenAI and DeepSeek emit 'stop', 'length', 'tool_calls', and 'content_filter'. Anthropic Messages API emits stop_reason values 'end_turn', 'max_tokens', 'tool_use', and 'stop_sequence'. Google Gemini emits 'STOP', 'MAX_TOKENS', 'SAFETY', and 'RECITATION'.

When an OpenRouter or custom gateway sits in front of all four families, subtle mapping bugs appear if application code checks if finish_reason == 'length' using a blacklist. Any unfamiliar provider string (such as 'MAX_TOKENS' or None on a dropped stream) slips right past a blacklist!

Always enforce a strict positive whitelist instead. Map raw provider strings into a canonical CanonicalFinish enum (COMPLETE, TOOL_CALL, TRUNCATED, FILTERED, UNKNOWN) and reject anything outside {COMPLETE, TOOL_CALL}.

**Table 6.1: Cross-Vendor Termination State Normalization Matrix**

| Provider Family | Clean Text Complete | Clean Tool Dispatch | Truncated (BLOCK) | Filtered / Unsafe (BLOCK) |
|---|---|---|---|---|
| **OpenAI / DeepSeek** | stop | tool_calls | length | content_filter |
| **Anthropic Claude** | end_turn / stop_sequence | tool_use | max_tokens | refusal |
| **Google Gemini** | STOP | STOP (with functionCall) | MAX_TOKENS | SAFETY / RECITATION |
| **Null / Dropped Stream** | N/A (Never valid) | N/A (Never valid) | None / '' / 'error' | other / unknown |

**WHITELIST OVER BLACKLIST:** A blacklist misses None, empty strings, and new vendor error codes; a positive whitelist fails closed on every unexpected termination state.

CHAPTER 10.3 // UNANIMOUS VETO MATH

# Unanimous Veto Math (all(votes)) vs. Leaky 3-of-4 Majority Rule

> **Under independent 5% false-pass rates, unanimous 4-of-4 voting cuts defect leakage to 0.000625%, whereas 3-of-4 majority voting leaks 80x more bad items.**

Engineers accustomed to distributed consensus protocols (Raft or Paxos) instinctively reach for majority voting (3 of 4). In fault-tolerant storage, majority voting masks crashed nodes. In AI safety and quality gating, majority voting re-opens defect leakage.

Walk through the binomial probability math on a defective item where each independent model family has false-pass probability p_fp = 0.05 (95% catch rate). Under unanimous voting (all(votes) in Python), a bad item slips through only if all four families miss it simultaneously: P(leak_4of4) = (0.05)^4 = 0.00000625, or roughly 6 times per million defective items.

Relax the gate to 3-of-4 majority (sum(votes) >= 3), and P(leak_3of4) = 4 * (0.05)^3 * (0.95) + (0.05)^4 = 0.00048125, a 77-fold jump even before accounting for partial cross-model correlation! On hard edge cases where p_fp rises to 0.20, 3-of-4 leaks 2.72% of defects while 4-of-4 holds leakage down to 0.16%.

**Table 10.2: Defect Leakage & Clean-Item Recall Under 4-Vendor Voting Rules**

| Per-Model Defect Leak (p_fp) | Solo Judge Leak | 3-of-4 Majority Leak | 4-of-4 Unanimous Leak | Clean Pass Recall (p_tp=0.99) |
|---|---|---|---|---|
| **p_fp = 0.05 (Routine)** | 5.0000% | 0.0481% | 0.0006% (~0 in 10k) | 96.06% auto-approved |
| **p_fp = 0.10 (Tricky)** | 10.0000% | 0.3700% | 0.0100% | 96.06% auto-approved |
| **p_fp = 0.20 (Adversarial)** | 20.0000% | 2.7200% (Unacceptable) | 0.1600% | 96.06% auto-approved |

**ASYMMETRIC LOSS FUNCTION:** False alarms only cost a human review or a cheap regeneration retry; false passes ship broken answers to customers. Optimize for zero false passes.

CHAPTER 15.3 // EMPIRICAL MODEL-BY-MODEL TELEMETRY

# Model-by-Model Telemetry: Qwen3 MLX vs. Claude Sonnet 4.6 & GPT-4o

> **Solo local 4-bit models run at $0.00 token cost and 83.3% accuracy on Suites A and B, but leak 20% of subtle Dirty Ops traps when deployed without a disagreement router.**

Inspect the actual measured results from tribunal_receipts.json. Local Qwen3-32B-4bit on M5 Ultra passed 13 of 18 cases (72.2% overall, 83.3% on Blind Solver and Injection Grading) at 1,492ms median latency and $0.00 cost. However, on Suite C's tricky multi-step Dirty Ops JSON cases, solo Qwen3-32B-4bit passed only 3 of 6 and allowed 2 false passes across the 10 defect traps (20.0% trap leak).

Look at what this proves about honest engineering measurement: deploying solo 4-bit Qwen3-32B as an unguarded replacement for frontier models would violate our zero-false-pass mandate!

Yet when we pair Qwen3-32B-4bit with Qwen3-8B-4bit in a Conservative Dual-Local Disagreement Screen (totaling 24.61 GB VRAM), their combined conservative false-pass count drops straight from 2 down to 0 (0.0% false-pass rate).

**Table 15.3: Measured Performance on the 72-Run M5 Ultra vs. Cloud Benchmark**

| Tier / Configuration | VRAM / Cost per 100k | Suites A & B Acc | False Passes | Role in Production Router |
|---|---|---|---|---|
| **Solo Qwen3-32B-4bit (Local)** | 19.29 GB \| $0.00 | 83.3% (10/12) | 2 / 10 (20%) | Unsafe solo; pair in Rung 3 |
| **Dual-Local Conservative** | 24.61 GB \| $0.00 | High Conservative | 0 / 10 (0.0%!) | Safe Rung-3 local filter |
| **Claude Sonnet 4.6 (Cloud)** | API \| ~$344 / 100k | 100.0% (12/12) | 0 / 10 (0.0%) | Rung-5 escalation anchor |
| **Hybrid Disagreement Router** | 24.61 GB \| $210.38 | 100.0% (18/18!) | 0 / 10 (0.0%!) | 38.9% free offload + 100% acc |

**HONEST BENCHMARKING:** Local 4-bit models excel as conservative Rung-3 pre-screens, provided any local disagreement or FAIL flag escalates to cloud consensus.