Case Study:

Newsroom Release Console: Ship/Block Governance with Feedback Loop

Prepared by **Rex Chang**

GenAI & Cloud Solution Architect / Program Lead



Check AI drafts before they go live.

Review workflow

Catch missing references, review decisions, and test fixes before the next release.

Measured result

- Release checks identified all 5 deliberately flawed drafts in a 12-case controlled test.

Checks

- Check memo structure, reference IDs, and required support markers.

Review trail

- Run receipts help editors review issues and retest fixes.



Why This System Exists

Test AI changes against evidence, then turn feedback into regression checks.

Business Context:

Editorial teams increasingly use AI to draft internal decision memos for sensitive content, including health claims, sponsored articles, and policy-sensitive coverage.

A single incorrect approval can create:

- Compliance exposure
- Reputational damage
- Editorial trust erosion

Operational challenge

- Investigating flagged outputs can require manual reconstruction.
- Teams frequently cannot determine which sources, instructions, or configuration produced the AI's conclusion.

Why naive solutions fail:

Even when retrieval is used:

- Models may cite policies that were never actually provided.
- Missing evidence can remain hidden behind fluent summaries.
- Without replayability, teams cannot reconstruct how the decision was produced.

Economic Exposure:

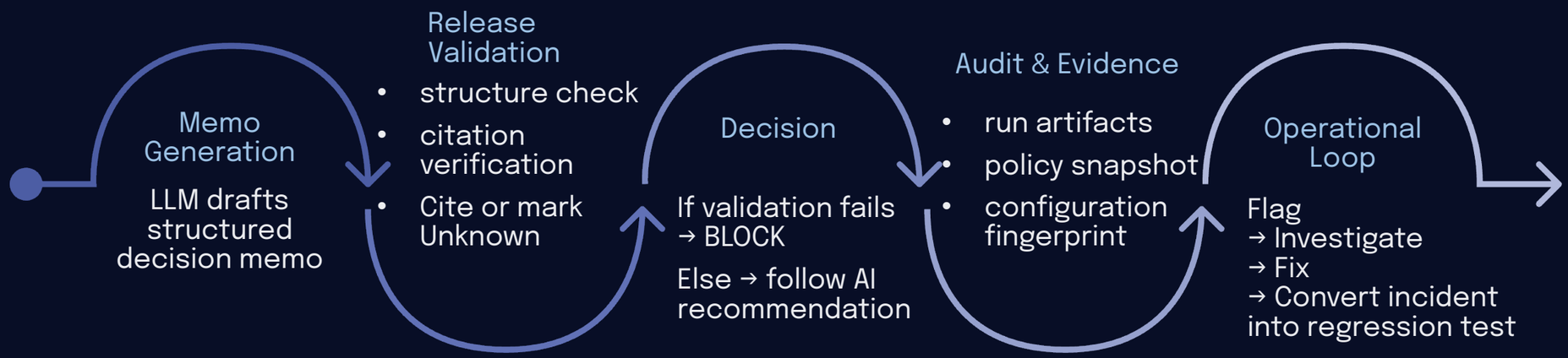
Organizations face a trade-off:

- Pay the ongoing cost of manual verification

OR

- Accept the risk of publishing claims that cannot later be justified.

End-to-End Decision Flow



Governance
— validation layer

Mechanical checks validate structure, reference IDs, and required support markers.

Measurability
— stored run artifacts

Every run stores memo, source snapshot, and configuration fingerprint, enabling deterministic replay and audit.

Cost control
— lightweight policy index, no retrieval stack

Uses a fixed source index per run, avoiding retrieval overhead while keeping citation targets auditable.

⚙ Scalability
— deterministic replay + regression suite

Flagged incidents become regression tests.

Deliberate Choices at Decision Points

Each row maps to the flow: *Memo Generation → Release Verification → Decision Logic → Operation Loop*

Flow Node	Deliberate Choice	Alternative	Business Implication	Trade-off
Memo Generation	Single LLM + stub testing modes	Multi-agent pipelines	Simpler infra vs reasoning sophistication	Memo Generation
Release Validation	Mechanical rule gates	Human review or LLM judge	Deterministic enforcement vs semantic flexibility	Release Validation
Decision Logic	Separate compliance vs editorial decision	Single decision output	Clear accountability vs additional reporting fields	Decision Logic
Operational Loop	Regression suite promotion	Manual fixes only	Maintenance cost vs regression prevention	Operational Loop

Release Checks in a Controlled Test

Controlled release checks with recorded decisions and regression tests.

Controlled test setup

Test set:
- 12 constructed newsroom cases, including 5 deliberately flawed drafts.

Checks:
- Memo structure, reference IDs and selected-claim support markers.

Check	Cases	Flagged	Meaning
Policy references	9	4	Invalid document/section IDs
Citation or Unknown	12	2	Missing required support marker

Reference check: policy document and section IDs exist.

For selected claims, a citation or explicit Unknown is required.

Controlled-Test Result

Controlled test outcome:

- **5 flawed drafts flagged**
- **12 constructed test cases**
- **Fixed-rule checks**

5 distinct failures; 1 case failed both checks.

Recorded run context

- Prompt and policy versions
- Replayable run receipts

12 constructed cases; 1 fails both checks.

Reference checks: 9 cases with policy context. Support-marker checks: 12 cases. Citation meaning still requires editorial review.



Release Gates and Incident Review

Validation gates separate a model recommendation from the release decision, with receipts available for review.

Release Control

- A memo that fails the configured validation checks is blocked by the release gate.

Auditability

Each decision stores a reproducible evidence bundle for incident reconstruction:

- exact prompt used
- exact sources available
- configuration fingerprint

Operational Control Loop

Reader flag

→ Retrieve receipts

→ Diagnose root cause

→ Fix input or policy

→ Convert incident into regression guard

Data Governance

- The design includes feedback scrubbing and retention rules for run and feedback data.

Appendix

Technical Deep Dive



Architecture: Data Flow + Policy Flow + Observability

Data Flow
Editor request → memo generation → parsing → structured memo artifact

Policy Flow
Policy pack snapshot → citation rules → release

Corrections flow
Flag → receipts → triage → promote to regression suite

1. Inputs & Context

- Editor request defines the business question to answer
- Prompt registry fixes the instruction version
- Policy pack snapshot fixes which document IDs and sections are allowed to be cited

6. Ops & Governance

- Release gate decides PASS/FAIL before any memo can be relied on
- Audit trail keeps build fingerprint, prompt version, and sources
- Retention rules control how long run / feedback data is kept

5. Measurement & Evidence

- Eval harness measures citation errors, Unknown-rule failures, format compliance, and editorial block behavior
- Run traces make each result inspectable by trace_id
- Cost / latency fields support future go / no-go decisions



2. Control Plane

- Validation gates check memo structure, citation-reference validity, and whether support-sensitive claims have a citation or an Unknown marker
- Governance status is separated from model recommendation
- Final release decision blocks on gate failure, otherwise follows the memo recommendation

3. Generation & Tools

- LLM / mock modes generate the draft decision memo
- Parser converts raw model text into a fixed structure
- Raw output is preserved so parsing or model behavior can be inspected later

4. State & Learning

- Feedback store logs reader / editor flags after PII scrubbing
- Triage converts incidents into labels + cheapest-fix actions
- GoldenSet promotion turns known failures into regression guards

Deep Mechanism:

Mechanical Checks, Run Receipts, and Regression

Core invariant:

Publish decisions must be reproducible, policy-grounded, and diagnosable using the exact run context available at decision time.

Gates = Mechanical Compliance, Not Model Vibes

- **MemoSchemaGate** blocks malformed outputs: all required memo sections must exist, and the decision field must parse cleanly.
- **CitationIntegrityGate** checks that every POLICY:<doc_id>#<section_id> citation exists in the selected policy pack index.
- **UnknownRuleGate:** Require support-sensitive claims to include a valid policy reference or an explicit Unknown marker.

Receipts Preserve the Decision Context

- Every run stores **raw_generation.txt**, **memo.json**, and **metadata.json** with trace_id, prompt version, reference pack, and build fingerprint.
- **Show Receipts** reconstructs the exact decision context: policy pack snapshot, prompt alias, model stack, and governance outcome.

Regression Promotion = System Memory, Not Manual Fixes

- A flagged incident can be promoted into **new dataset** with prompt, reference pack, expected decision, and assertion.
- This turns post-release corrections into pre-release protection.

EvalSet = Adapter Decision Gate, Not More Complexity by Default

- **EvalSet** is separate from GoldenSet: it measures baseline behavior across curated newsroom scenarios.
- Metrics include format compliance, phantom citation rate, unknown-rule violations, and editorial block behavior.



Thank You!

Rex Chang

GenAI & Cloud Solution Architect / Program Lead

Portfolio: rexchang.dev | LinkedIn: linkedin.com/in/rex-chang-tw | Email: rex@rexchang.dev