Governed Agentic Automation for Chargebacks:

A Prompt-First Architecture for Policy-Driven Enterprise Workflows

Nataraj Agaram Sundar[1] and Tejas Morabia[1]

[1]eBay Inc.

April 25, 2026

Abstract

Generative AI adoption is outpacing reliable productionization, especially in enterprise workflows that combine structured inputs, policy constraints, and operational risk. This technical report describes a prompt-first architecture for chargeback automation that replaces brittle, retrieval-heavy integrations with a governed orchestration layer for prompt management, policy enforcement, and observability. Using chargeback dispute handling as a case study, the report examines the transition from a Retrieval-Augmented Generation (RAG) design to modular prompt chaining and agentic execution over structured inputs. In internal deployment measurements, the prompt-first approach reduced generation latency by 35%, improved internal quality scores by 20%, and enabled near-immediate policy updates without re-indexing. A centralized guardrail framework auto-resolved 85% of policy issues before human review, while inter-rater agreement between reviewers improved from Cohen's kappa 0.65 to 0.82 across three optimization cycles. The results suggest that for decision-grade automation over structured enterprise data, governed prompt orchestration can outperform more complex retrieval-heavy designs in speed, controllability, and operational fitness. Some system names and implementation details are generalized.

Keywords: Generative AI, Agentic AI, Prompt Engineering, AI Governance, Chargebacks, Enterprise AI, LLM Orchestration, Risk Operations

Author note. This report reflects production lessons from enterprise AI automation. Some system names and implementation details are generalized for confidentiality. Quantitative performance figures reported here are internal deployment metrics unless otherwise noted.

1 Introduction

The current enterprise AI market is experiencing what many teams describe as an implementation hangover: adoption is high, but durable production value remains uneven. Gartner has projected that at least 30% of generative AI projects will be abandoned after proof of concept by the end of 2025 because of poor data quality, inadequate risk controls, escalating costs, or unclear business value [1].

This report argues that the main bottleneck is often not model capability but system design. Many organizations still rely on point solutions: narrowly scoped, hard-coded wrappers around a model API that may work in isolated demos but degrade quickly under operational load, compliance review,

or policy change. In contrast, production-grade workflows require governed orchestration: reusable infrastructure for prompt management, policy enforcement, evaluation, and observability.

To ground the discussion in a high-value, operationally demanding workflow, this report uses chargeback automation as a case study. Chargebacks and consumer disputes continue to impose significant costs on digital commerce. Sift, citing Mastercard data, reports worldwide chargeback losses of $33.79 billion in 2025, projected to rise to $41.69 billion by 2028 [2]. LexisNexis Risk Solutions reports that merchants spend an average of $4.61 for every $1 lost to fraud in the U.S. ecommerce segment, underscoring the heavy operational burden surrounding fraud, disputes, and review processes [3].

Within this context, the objective was not simply to generate text faster. The goal was to make dispute handling operable at scale: measurable, auditable, policy-aware, and adaptable. The result was a prompt-first, agentic orchestration pattern that reduced resolution time from days to minutes in internal deployment while preserving control, traceability, and governance.

2 From Complexity to Clarity

Chargeback handling is a representative enterprise workflow because it combines structured evidence, evolving rules, and high review sensitivity. Engineering teams working in this class of problem face a foundational choice: increase system complexity in search of richer context, or simplify the architecture around the structure that already exists in the data and process.

Our design principle was to prioritize clarity, control, and adaptability over architectural novelty. That meant treating the workflow as a governed decision-support system rather than as a generalized question-answering problem. The practical implication was that the system had to support prompt versioning, policy updates, human review, and auditability as first-class requirements rather than afterthoughts.

A core lesson emerged early: architectural complexity can become its own source of latency, brittleness, and quality drift. In policy-heavy enterprise settings, the simplest architecture that preserves control often has the greatest chance of surviving production.

3 The RAG Roadblock: When Architecture Becomes Drag

The initial design followed a Retrieval-Augmented Generation (RAG) pattern [4]. In theory, retrieval should improve factual grounding by embedding case materials, retrieving similar examples, and supplying them to the model as additional context. In practice, the workflow introduced multiple operational burdens that outweighed its benefits for this use case.

First, the RAG stack added too many moving parts. Vector stores, embedding pipelines, retrievers, and ranking layers all required maintenance and tuning. Second, retrieval frequently introduced context drift: redundant or weakly relevant examples contaminated the prompt and made outputs less consistent. Third, every policy or schema adjustment created re-indexing work, slowing iteration precisely when fast operational adaptation was most important.

For structured, policy-driven inputs, these frictions turned architectural elegance on paper into operational drag in production. Figure 1 makes that contrast explicit: the current-state point-solution trap duplicates prompts, custom code, and model access across isolated applications, while

the target-state orchestration platform centralizes guardrails, prompt versioning, observability, and reusable model access behind a unified API.

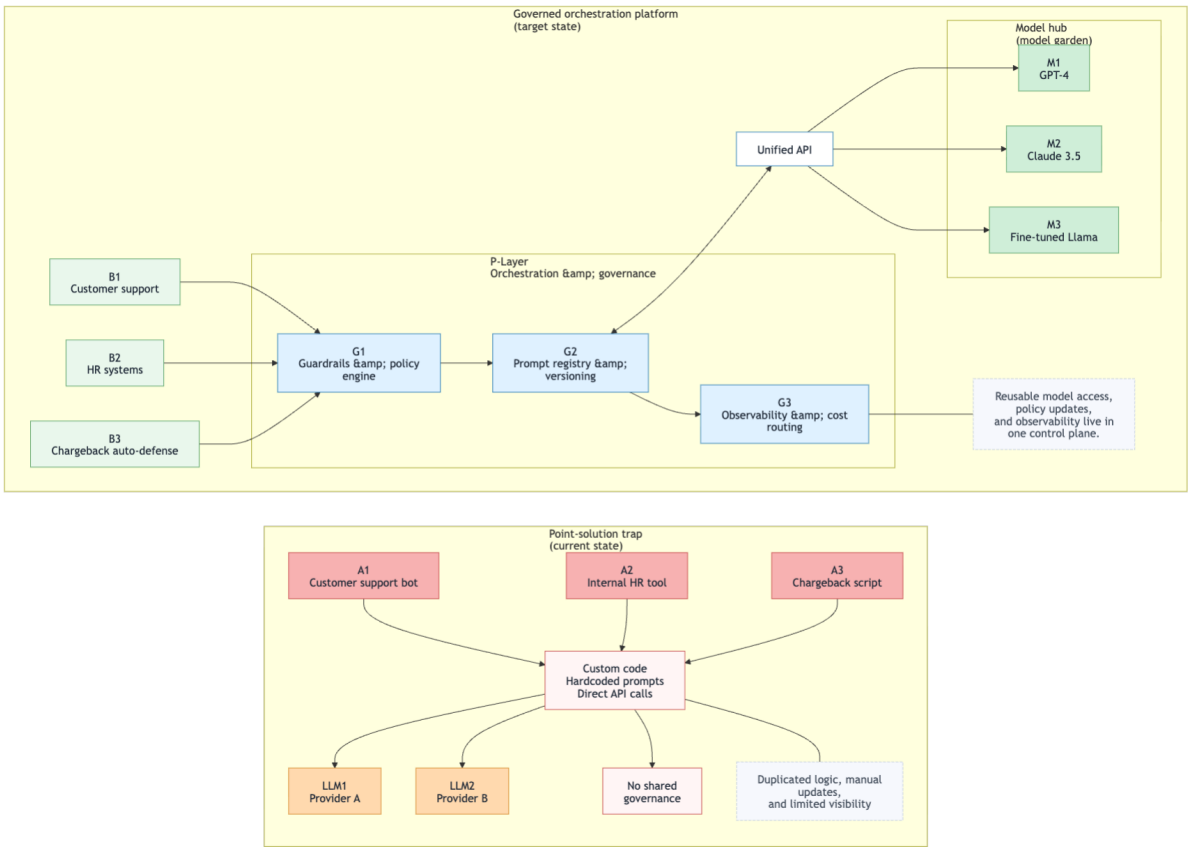


Figure 1: Architecture comparison. The lower panel shows the point-solution trap, where individual applications embed hard-coded prompts and direct model calls. The upper panel shows the governed target state, where shared guardrails, prompt registry/versioning, observability, and model routing sit in a reusable control plane.

4 The Data-Driven Pivot: Prompt-First Generation

A structured review of system behavior led to a prompt-first redesign. Instead of retrieving additional examples from a vector layer, the system used carefully organized case inputs and modular prompt design to guide generation directly. This reduced system dependencies and made the workflow easier to observe, evaluate, and update. The redesign also benefited from prompt-engineering practices that emphasize explicit structure, reusable prompt patterns, and decomposed reasoning steps [5].

Internal deployment measurements supported the shift. Generation latency decreased by 35%, internal quality scores increased by 20%, vector-store maintenance overhead was removed, and prompt changes could be applied immediately instead of waiting for re-indexing cycles. For chargeback summaries, where inputs are structured and policies change frequently, prompt-first generation outperformed the retrieval-heavy alternative in both efficiency and reliability.

Related practitioner implementations by the same author reinforce the same theme from adjacent

domains: additive relevance layers can improve retrieval without destabilizing the upstream content model, and long-term preference memory should remain separated from the immediate prompt context rather than being naively appended to every request [12, 15].

The practical implication is not that RAG is broadly obsolete. Rather, the result suggests that retrieval should be matched to the information topology of the problem. When the task is decision-grade automation over structured inputs, simpler prompt orchestration can be the better engineering choice.

5 Engineering Trust: Control, Observability, and Auditable AI

Simplifying the architecture did not mean relaxing control. Trust had to be engineered into the system through modularity, traceability, and constrained execution. Two implementation patterns were especially important: prompt chaining and agentic orchestration.

Prompt chaining replaced large monolithic prompts with a sequence of smaller, purpose-built steps. Tasks such as evidence counting, title generation, and summary drafting were decomposed so that each step could be inspected, tuned, and validated independently. This reduced failure propagation and improved determinism. More broadly, recent work on reasoning-and-acting loops supports the value of decomposing multi-step tasks into controllable intermediate stages [6].

Agentic orchestration then wrapped those steps in a controlled execution layer. Given a dispute identifier, the agent planned the workflow, invoked authorized data sources, coordinated prompt execution, and exposed step-level traces for review. Instead of behaving like a black box, the system produced auditable execution artifacts that could be examined during operations and review. Similar practitioner architecture patterns appear in production-ready voice-agent systems and plugin-mediated multi-agent bridges, where observability, guarded tool access, and clean extension boundaries are treated as first-class operating constraints [14, 13].

6 Governance as Data: Policy-Driven Guardrails

At enterprise scale, the central challenge is not simply generating fluent text. It is ensuring that every output is safe, policy-compliant, consistent, and measurable. To address this, the system incorporated a centralized evaluation layer, internally referred to as Cerberus, that ran multiple automated checks in parallel on every draft. The need for this kind of explicit governance aligns with current risk-management guidance for trustworthy AI and generative AI systems [7, 8, 9].

These checks covered several categories at once: sensitive-data detection, content policy moderation, schema and formatting validation, domain-specific rule enforcement, and readiness signals for downstream workflows such as translation or human review. Because the checks returned machine-readable evidence, the system could block, auto-correct, or reroute outputs before they reached an analyst. Figure 2 shows this logic explicitly: policy-as-data configures the checks, the draft is evaluated in parallel, and the resulting readiness signals determine whether the output passes, is repaired, is blocked, or is routed to human review.

The framework auto-resolved 85% of policy issues before human review in internal deployment. Equally important, the guardrails were implemented as data—specifically YAML-configured thresholds and rules—so policy teams could update behavior without software redeployment. This design preserved flexibility while keeping governance explicit and testable.

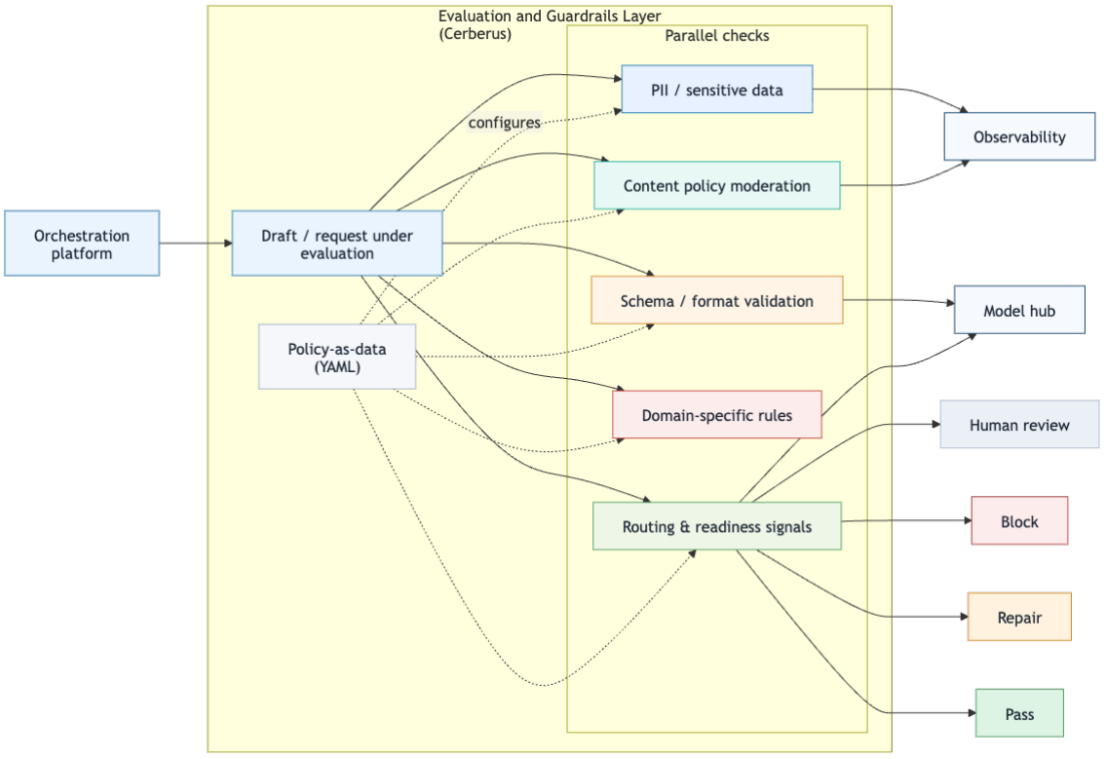
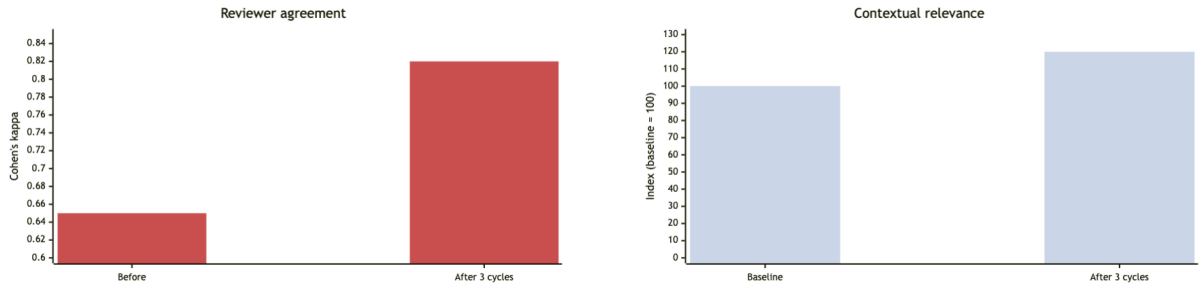


Figure 2: Cerberus evaluation and guardrails layer. A request under evaluation is checked in parallel for sensitive data, content policy, schema validity, domain-specific rules, and routing readiness, while YAML-based policy definitions configure how outcomes are logged, escalated, repaired, blocked, or approved.

7 Closing the Loop: Continuous Learning Through Human Review

No production AI system is flawless, so the architecture incorporated a Human Review and Feedback Loop (HRFL) to support continuous improvement. The purpose of the loop was not merely to catch errors, but to convert expert review into a measurable optimization signal.

Reviewer consistency was monitored with Cohen's kappa, a widely used measure of inter-rater agreement for categorical judgments [10, 11]. Across three optimization cycles, kappa increased from 0.65 to 0.82 in internal evaluation, while contextual relevance improved by 20%. The movement from substantial agreement toward almost perfect agreement supported the conclusion that reviewers found the system outputs increasingly reliable and aligned with operational expectations. Figures 3a and 3b show these two improvements directly.



(a) Reviewer agreement improved from Cohen's kappa 0.65 before optimization to 0.82 after three cycles.

(b) Contextual relevance improved from a baseline index of 100 to 120 after three optimization cycles.

Figure 3: Optimization outcomes from the Human Review and Feedback Loop (HRFL).

Figure 4 then consolidates the two findings into a single interpretive view: iterative prompt optimization increased both reviewer agreement and contextual relevance, suggesting that quality gains were not confined to one evaluation axis but reflected broader alignment with expert judgment.

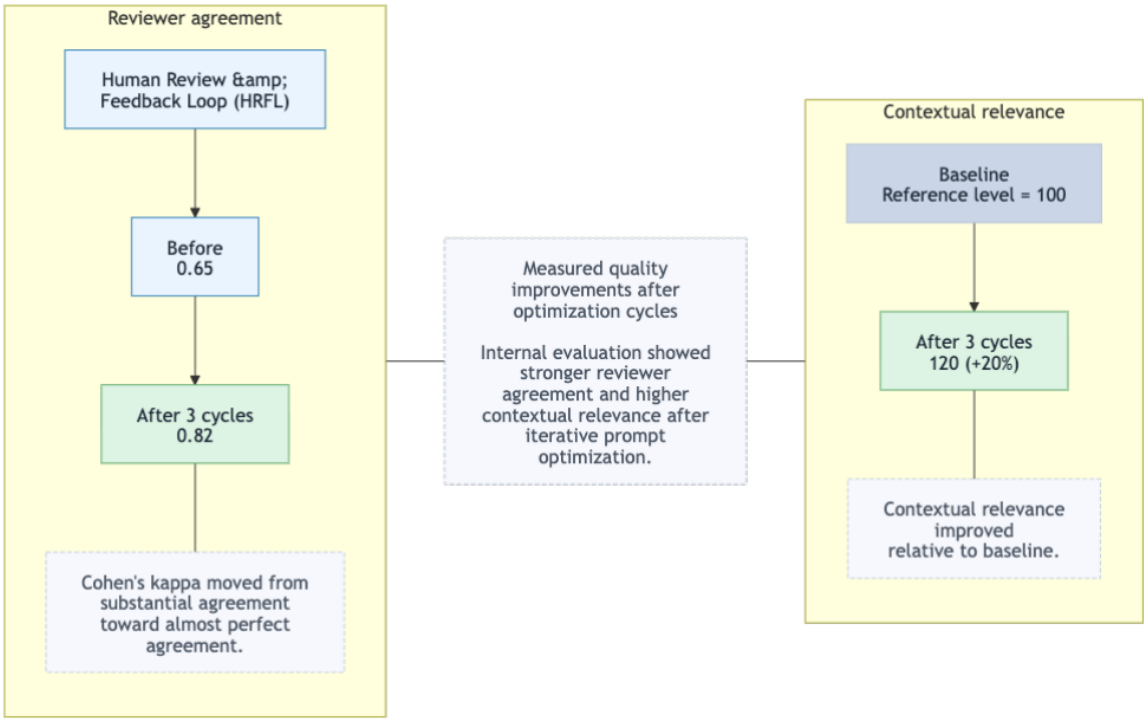


Figure 4: Consolidated quality improvements after optimization cycles. The summary view links higher reviewer agreement with stronger contextual relevance after iterative prompt tuning and human-in-the-loop evaluation.

The broader lesson is that production readiness is not achieved by a single model selection decision; it emerges from a repeatable loop of evaluation, feedback, and system refinement.

8 Final Takeaway: Match the Architecture to the Problem

The main conclusion of this case study is that AI architecture should be chosen as a response to workflow structure, not as an identity statement. Retrieval-centric systems remain useful for large, unstructured corpora and deep historical search. But for structured, policy-sensitive, latency-aware workflows, a prompt-first architecture can provide a better balance of speed, controllability, and operational fitness.

By shifting from brittle, retrieval-heavy integrations to a governed orchestration platform, the system described here made generative AI measurable, auditable, and adaptable in a decision-grade enterprise setting. In that sense, the most important design move was not adding more model complexity. It was building the surrounding architecture that allowed intelligence to operate safely at scale.

References

[1] Gartner. "Gartner Predicts 30% of Generative AI Projects Will Be Abandoned After Proof of Concept by End of 2025." Press release, July 29, 2024.

[2] Sift. "The Rising Impact of Chargebacks and Consumer Disputes." Q4 2025 Digital Trust Index, 2025.

[3] LexisNexis Risk Solutions. "Fraud Costs Surge as North America's Ecommerce and Retail Businesses Face Mounting Financial and Operational Challenges." True Cost of Fraud Study: Ecommerce and Retail Report – U.S. and Canada Edition, 2025.

[4] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K"uttler, Mike Lewis, Wen-tau Yih, Tim Rockt"aschel, Sebastian Riedel, and Douwe Kiela. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." In *Advances in Neural Information Processing Systems*, 33, 2020.

[5] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT." *arXiv preprint* arXiv:2302.11382, 2023.

[6] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. "ReAct: Synergizing Reasoning and Acting in Language Models." In *International Conference on Learning Representations*, 2023.

[7] Elham Tabassi. *Artificial Intelligence Risk Management Framework (AI RMF 1.0).* NIST AI 100-1, National Institute of Standards and Technology, 2023.

[8] Chloe Autio, Reva Schwartz, Jesse Dunietz, Shomik Jain, Martin Stanley, Elham Tabassi, Patrick Hall, and Kamie Roberts. *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile.* NIST AI 600-1, National Institute of Standards and Technology, 2024.

[9] OWASP Foundation. *OWASP Top 10 for Large Language Model Applications / GenAI Security Project.* 2025 update.

[10] Jacob Cohen. "A Coefficient of Agreement for Nominal Scales." *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[11] J. Richard Landis and Gary G. Koch. "The Measurement of Observer Agreement for Categorical Data." *Biometrics*, 33(1):159–174, 1977.

[12] Nataraj Sundar. "How to Use Context Hub (chub) to Build a Companion Relevance Engine." *freeCodeCamp*, April 17, 2026. Available at: https://www.freecodecamp.org/news/how-to-use-context-hub-chub-to-build-a-companion-relevance-engine/

[13] Nataraj Sundar. "How to Set Up OpenClaw and Design an A2A Plugin Bridge." *freeCodeCamp*, April 7, 2026. Available at: https://www.freecodecamp.org/news/openclaw-a2a-plugin-architecture-guide/

[14] Nataraj Sundar. "How to Build a Production-Ready Voice Agent Architecture with WebRTC." *freeCodeCamp*, March 6, 2026. Available at: https://www.freecodecamp.org/news/how-to-build-production-ready-voice-agents/

[15] Nataraj Sundar. "How to Build AI Agents That Remember User Preferences (Without Breaking Context)." *freeCodeCamp*, February 11, 2026. Available at: https://www.freecodecamp.org/news/how-to-build-ai-agents-that-remember-user-preferences-without-breaking-context/