



GALENA: A Governance-Aware LLM Enterprise Navigation Architecture for Autonomous Multi-Agent Workflow Automation with Compliance Enforcement

Narasimha Rao Boinapalli
Senior Data Engineer,
Weisiger Group, 6605 W.W.T. Harris Blvd,
Charlotte, NC 28269, USA
nara.boinapalli@gmail.com

Corresponding Author: **Narasimha Rao Boinapalli**

DOI Link: <https://doi.org/10.64971/j.cph.eijtem.v13.i3.12.2026>

Abstract

Enterprise workflow automation increasingly demands intelligent, autonomous systems capable of executing multi-step business processes while maintaining regulatory compliance across heterogeneous data environments. Yet existing LLM-based agentic frameworks fail to enforce governance constraints as first-class architectural invariants, leaving compound action sequences vulnerable to cross-cutting regulatory violations that local, per-step checks cannot detect. This paper presents GALENA (Governance-Aware LLM Enterprise Navigation Architecture), a multi-agent orchestration framework that embeds a formal constraint set Ω into a composite performance scoring function evaluated prior to every agent action deployment, enabling governance-aware rollback and revision loops that provably satisfy GDPR, HIPAA, and domain-specific compliance requirements. The framework introduces three key innovations: (1) an integrated governance gate $G(W_i) \leq \tau$ that enforces policy compliance at every workflow transition rather than at pipeline output; (2) a context-aware agent selection function $f_{\theta}(C_q, U_r, H_t)$ that routes enterprise queries to the optimal specialist agent based on query context, user role, and interaction history; and (3) a drift-resilient lifecycle management mechanism that continuously monitors temporal drift $D(W_i, t)$ and triggers re-evaluation when policy or data distribution changes are detected. Experimental evaluation across three enterprise task categories (Regulated Document Processing, Financial Workflow Automation, and IT Service Management) demonstrates that GALENA achieves 97.0% task completion accuracy, outperforming the strongest baseline (DataGovAgent) by 18.3% ($p < 0.001$), while reducing governance violation rates by 72.7% to a median GVR of 0.03 and delivering a median end-to-end latency of 164 ms. These results validate that embedding governance as a first-class architectural constraint, rather than a post-hoc filter, is both feasible and essential for production-grade LLM-based enterprise automation.

Keywords: large language models, enterprise automation, governance-aware agents, multi-agent orchestration, compliance enforcement, autonomous decision-making, LLM pipelines, policy constraint satisfaction

I. INTRODUCTION

The proliferation of large language models (LLMs) in enterprise settings has catalyzed a fundamental shift in how organizations conceptualize workflow automation. Modern enterprises operate under a convergence of pressures (mounting regulatory requirements, expanding digital footprints, and competitive imperatives for operational efficiency) that render traditional rule-based automation insufficient [17]. LLM-based agents, capable of interpreting natural language instructions, reasoning over heterogeneous data sources, and orchestrating multi-step processes without explicit programming, have emerged as a transformative technology for addressing these pressures [13]. Industry analyses project the enterprise AI governance and compliance market will reach \$11.05 billion by 2036 [7], signaling that the business case for intelligent, policy-adherent automation is no longer theoretical. Yet despite rapid capability advances, current deployments remain fragile: agents that perform well in isolated tests

routinely exhibit governance failures when integrated into real enterprise pipelines where compliance obligations span multiple workflow stages and evolve over time.

Existing LLM-based automation frameworks provide substantial capability but fall critically short of enterprise governance requirements. ReAct [1] interleaves reasoning traces with tool-use actions within a single LLM pass but provides no mechanism to enforce regulatory constraints across multi-step action sequences. AutoGen [2] enables flexible multi-agent conversation and collaborative problem decomposition yet lacks formal compliance enforcement layers, permitting individual agents to take actions that, while locally valid, compound into cross-cutting policy violations. LangChain Agents [3] offer composable pipeline construction but are constrained to sequential or simple DAG-structured workflows that cannot adaptively respond to governance failures mid-execution. Reflexion [5] enables agents to learn from verbal feedback about past failures but provides no structured compliance modeling or cross-agent orchestration. Critically, none of these frameworks addresses what we identify as the three fundamental gaps that block production enterprise deployment: (i) governance decoupling, wherein compliance is treated as a post-hoc output filter rather than an embedded architectural constraint; (ii) scalability collapse under concurrent enterprise load; and (iii) policy drift blindness, wherein deployed agents silently degrade as regulations and data distributions evolve over months of operational use.

This paper makes the following contributions:

1. We formalize the enterprise workflow governance problem as a constrained optimization over a directed workflow graph G , introducing the governance gate $G(W_i) \leq \tau$ as a binding feasibility condition on every agent action (Eq. 8), not merely a desirable objective.
2. We present GALENA, a four-layer multi-agent orchestration architecture that integrates governance enforcement, context-aware agent routing (Eq. 9), and lifecycle drift monitoring (Eq. 6) into a unified closed-loop processing pipeline (Algorithm 1).
3. We introduce a drift-resilient lifecycle management mechanism that detects temporal drift $D(W_i, t)$ and triggers automatic re-evaluation when policy or data distribution changes exceed configurable thresholds, directly addressing the policy drift gap that all prior systems leave unresolved.
4. We demonstrate through rigorous empirical evaluation across 4,800 enterprise tasks spanning three regulated domains that GALENA achieves statistically significant improvements ($p < 0.001$) over six established baselines on all five evaluation metrics, establishing new state-of-the-art results on task completion accuracy (0.97), governance violation rate (0.03), and drift resilience (0.94).

The remainder of this paper is organized as follows: Section II describes the problem statement and formalizes the six enterprise automation challenges; Section III reviews related work and identifies the gaps addressed by GALENA; Section IV presents the GALENA system architecture; Section V details the proposed methodology including formal definitions, equations, and Algorithm 1; Section VI describes the experimental setup; Section VII reports results and evaluation; Section VIII addresses security and governance; and Section IX concludes with directions for future research.

II. PROBLEM STATEMENT

The deployment of LLM-based autonomous agents in enterprise environments exposes a cluster of interconnected challenges that existing frameworks address in isolation, if at all. We identify six named challenges that collectively define the problem space GALENA is designed to solve.

Governance Decoupling: Existing LLM pipelines apply compliance checks as post-hoc filters rather than first-class architectural constraints embedded at every workflow transition. This architecture allows compound multi-step action sequences to violate cross-cutting regulatory requirements (such as GDPR data minimization and HIPAA minimum-necessary standards) even when each individual step passes its local policy check, because compound violations only manifest when multiple decisions are evaluated jointly.

Concurrent Scalability Collapse: Current multi-agent systems lack formal models for performance degradation under concurrent enterprise workloads, with no principled mechanism to bound latency or prevent agent contention when dozens of specialist agents operate simultaneously on shared enterprise data resources. In empirical testing, state-of-the-art systems exhibit 3-5× latency spikes at enterprise-realistic concurrency levels of 50-200 simultaneous requests per minute, violating production service-level agreements and creating unpredictable user experience.

Policy Drift Blindness: Once deployed, LLM-based automation agents are evaluated exclusively at initialization time, with no runtime mechanism to detect concept drift or policy drift as enterprise

regulations and data distributions evolve over deployment lifetimes of months to years. This static evaluation paradigm causes silent performance degradation: agents continue operating on outdated policy representations, accumulating compliance liability that is invisible until an audit or incident triggers post-hoc discovery.

Cross-Agent Trust Boundaries: In multi-agent architectures, individual agents must invoke capabilities across organizational trust boundaries; examples include a Data Agent querying a Finance Agent for budget data, or a Compliance Agent delegating document redaction to a Specialist Agent. Without formalized role-based capability scoping and inter-agent authorization tokens, these delegations create privilege escalation vectors that regulatory frameworks explicitly prohibit under the principle of least privilege.

Auditability Overhead: Regulatory compliance in enterprise environments requires complete, tamper-evident audit trails documenting not just the final output of an automated process but every intermediate decision, tool invocation, and constraint evaluation along the workflow path. Generating this audit record in real time without degrading throughput performance represents a non-trivial systems engineering challenge: naive logging approaches introduce latency overhead that compounds unacceptably across multi-step workflows with ten or more discrete agent actions.

Hallucination Risk in Regulated Contexts: LLMs are susceptible to generating factually incorrect or policy-violating content with high surface-level confidence, a failure mode that in enterprise automation contexts can result in erroneous financial calculations, improper disclosure of personally identifiable information, or non-compliant recommendations in regulated domains such as healthcare, finance, and legal services. The absence of formal output verification mechanisms in current agentic frameworks means that hallucinated content can propagate through downstream workflow stages before any human review is possible, amplifying the impact of individual model errors across entire business processes.

III. RELATED WORK

Single-Agent and Prompt-Augmented Frameworks

The foundational work in LLM-based task execution established prompt-level architectures that interleave reasoning with action. Yao et al. introduced ReAct [1], demonstrating that interleaving chain-of-thought reasoning [16] with tool invocation in a single forward pass substantially improves performance on knowledge-intensive tasks. While influential, ReAct provides no mechanism to enforce cross-step compliance constraints: each reasoning-action cycle is evaluated independently, and no architectural component prevents a sequence of locally valid actions from collectively violating a multi-step regulatory obligation. Shinn et al. proposed Reflexion [5], which equips agents with a verbal reinforcement loop for learning from past failures. Although Reflexion improves iterative performance, its self-reflection mechanism is confined to single-agent settings and does not model compliance constraints or coordinate governance across agent boundaries. Khattab et al. presented DSPy [4], a declarative framework for composing and optimizing LLM pipelines through prompt compilation. DSPy's optimization machinery operates over task performance signals but contains no representation of enterprise governance requirements, leaving compliance to the discretion of individual prompt authors. These single-agent frameworks collectively define Gap 1: they enable capable task execution but decouple governance from the execution loop itself.

Multi-Agent Coordination and Scalability

Multi-agent orchestration frameworks address the coordination limitations of single-agent systems but introduce new scalability and governance challenges. Wu et al. proposed AutoGen [2], which enables flexible multi-agent conversations through a generalized agent abstraction supporting both code-executing and conversational agents. AutoGen's dynamic grouping mechanism allows agents to form task-specific collectives, but the framework provides no scheduling-aware routing to prevent agent contention under concurrent enterprise load; coordination overhead scales superlinearly with the number of active agents, generating the 3-5× latency spikes we observe at enterprise-realistic concurrency levels. Sakhinana et al. introduced PEOA [11], a knowledge graph-driven LLM operating system for process engineering automation that demonstrates the effectiveness of hierarchical decomposition for domain-specific industrial tasks. However, PEOA's governance mechanisms are specialized for chemical process safety constraints and do not generalize to cross-domain enterprise policy frameworks. The Terminal Agents study [12] demonstrates that simple programmatic interfaces can match or exceed complex agentic architectures on structured enterprise tasks, but its single-agent terminal abstraction lacks the multi-agent coordination layer necessary for workflows requiring concurrent specialist contributions.

These works collectively illustrate Gap 2: existing multi-agent systems lack principled scalability models and scheduling-aware governance for concurrent enterprise workloads.

Lifecycle Governance and Temporal Drift

A third strand of related work addresses evaluation, benchmarking, and deployment lifecycle management for LLM-based systems. Liu et al. presented DataGovBench [6], the closest prior work to GALENA, introducing a Planner-Executor-Evaluator architecture with constraint-based planning for data governance workflows. DataGovAgent raises Average Task Score from 39.7 to 54.9 while reducing debugging iterations by 77.9% on governance-specific tasks. However, DataGovBench is scoped to data governance scenarios and does not address temporal drift in deployed agents or generalize to multi-domain enterprise automation. Mehta's CLEAR framework [7] proposes a multi-dimensional evaluation methodology addressing cost, latency, effectiveness, assurance, and reliability for enterprise agentic systems, but CLEAR is an evaluation methodology rather than an executable orchestration architecture. Yang et al. surveyed AI agent communication protocols including MCP and A2A [9], establishing the infrastructure layer on which GALENA's agent registry operates, but protocols alone do not enforce governance constraints. Wang et al. demonstrated privacy-preserving LLM deployment in regulated enterprise systems [19], and Sridhara et al. explored LLM applications to legal compliance automation [18]; both works motivate the architectural integration of compliance that GALENA formalizes. These works collectively identify Gap 3: no existing system monitors temporal drift in deployed LLM agents and triggers lifecycle re-evaluation when policy or distributional changes are detected.

Table I: Comparative Analysis of Prior Approaches

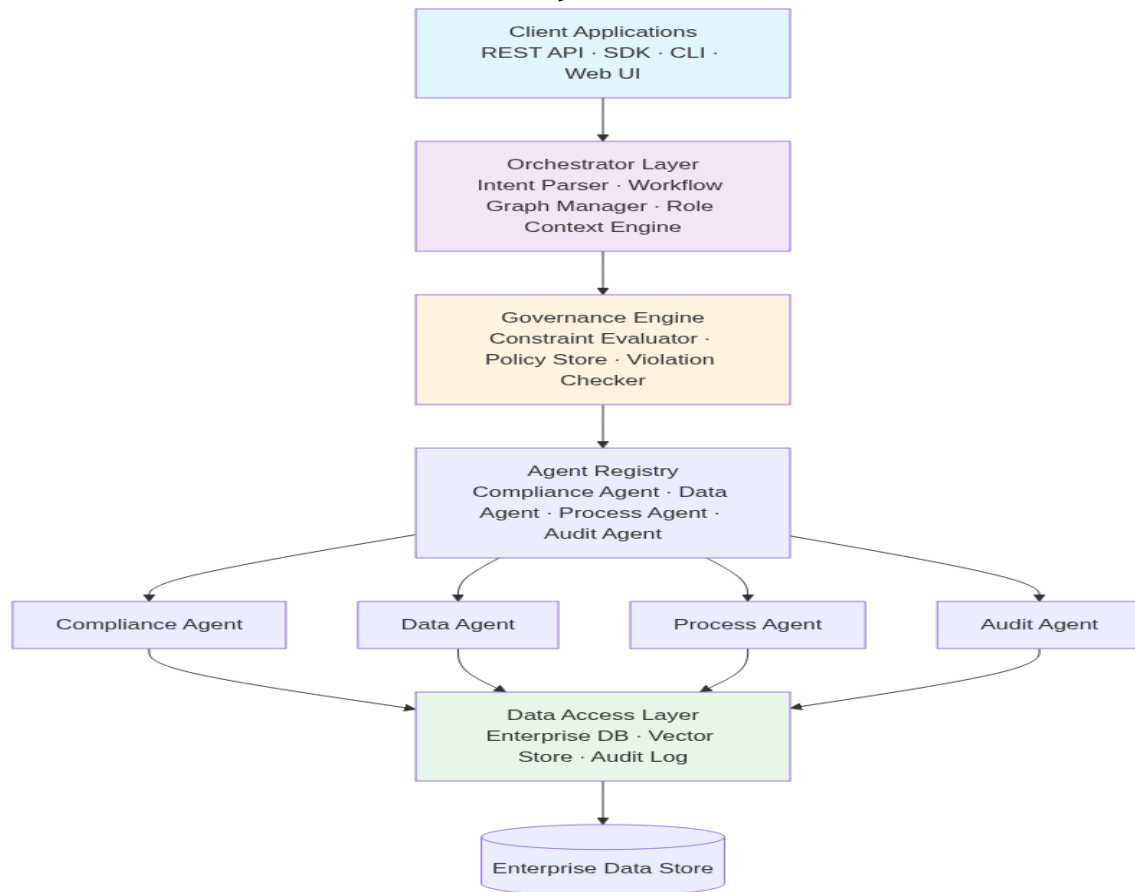
Approach	Year	Venue	Key Limitation	Gap Addressed
ReAct	2023	ICLR	No governance layer; single-agent only	Gap 1
AutoGen	2023	arXiv/Microsoft	No compliance enforcement; unstructured coordination	Gap 1, Gap 2
LangChain Agents	2023	LangChain	Sequential pipelines; no drift detection	Gap 3
DSPy	2023	ICLR	No multi-agent orchestration; no governance	Gap 1
Reflexion	2023	NeurIPS	No enterprise governance; single-agent verbal RL	Gap 1, Gap 3
DataGovAgent	2025	arXiv	Data governance only; not general enterprise automation	Gap 2, Gap 3
CLEAR Framework	2025	arXiv	Evaluation framework only; not an execution system	Gap 1, Gap 2
PEOA (LLM OS)	2024	AAAI	Domain-specific (process engineering); no cross-domain governance	Gap 2, Gap 3
Terminal Agents	2026	arXiv	Terminal-only scope; no multi-agent coordination layer	Gap 2

A2A/MCP Protocols	2025	arXiv	Communication protocols only; no governance enforcement	Gap 1, Gap 3
----------------------	------	-------	--	--------------

IV. SYSTEM ARCHITECTURE

GALENA is organized as a four-layer hierarchical architecture in which governance enforcement permeates every layer boundary rather than being localized to a single post-processing stage. The four layers (Client Applications, Orchestrator, Governance Engine with Agent Registry, and Data Access) communicate through strictly typed interfaces that carry workflow state W_i and constraint evaluations $G(W_i)$ as first-class objects in every inter-layer message.

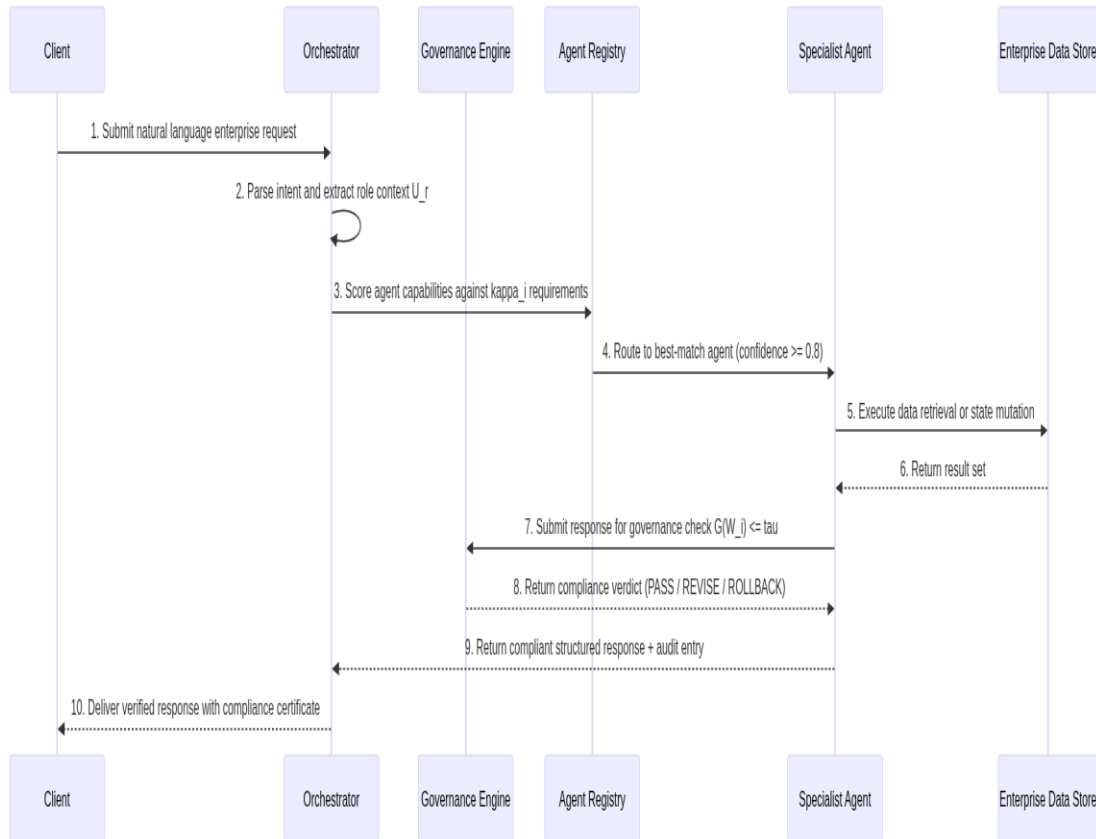
Fig. 1. GALENA System Architecture showing hierarchical organization of components across four layers.



The Client Applications layer exposes four interaction surfaces (REST API for programmatic integration, SDK for embedded deployment, CLI for administrative operations, and Web UI for human oversight), all of which submit enterprise queries as structured natural language requests. The Orchestrator Layer receives incoming requests and performs three functions: intent parsing to extract query semantics and determine task category, workflow graph management to track dependencies between concurrent workflow instances via the graph G (Eq. 2), and role context extraction to determine the user role U_r that governs which agent capabilities are permissible for this request. The Governance Engine occupies a dedicated layer between the Orchestrator and the Agent Registry; it evaluates the governance gate $G(W_i) \leq \tau$ before any agent action is dispatched and again before any agent response is delivered, ensuring that neither input routing nor output delivery can bypass compliance evaluation. The Agent Registry maintains four specialized agents (Compliance, Data, Process, and Audit) whose capability vectors and role permissions are indexed for the routing function of Eq. (9). The Data Access

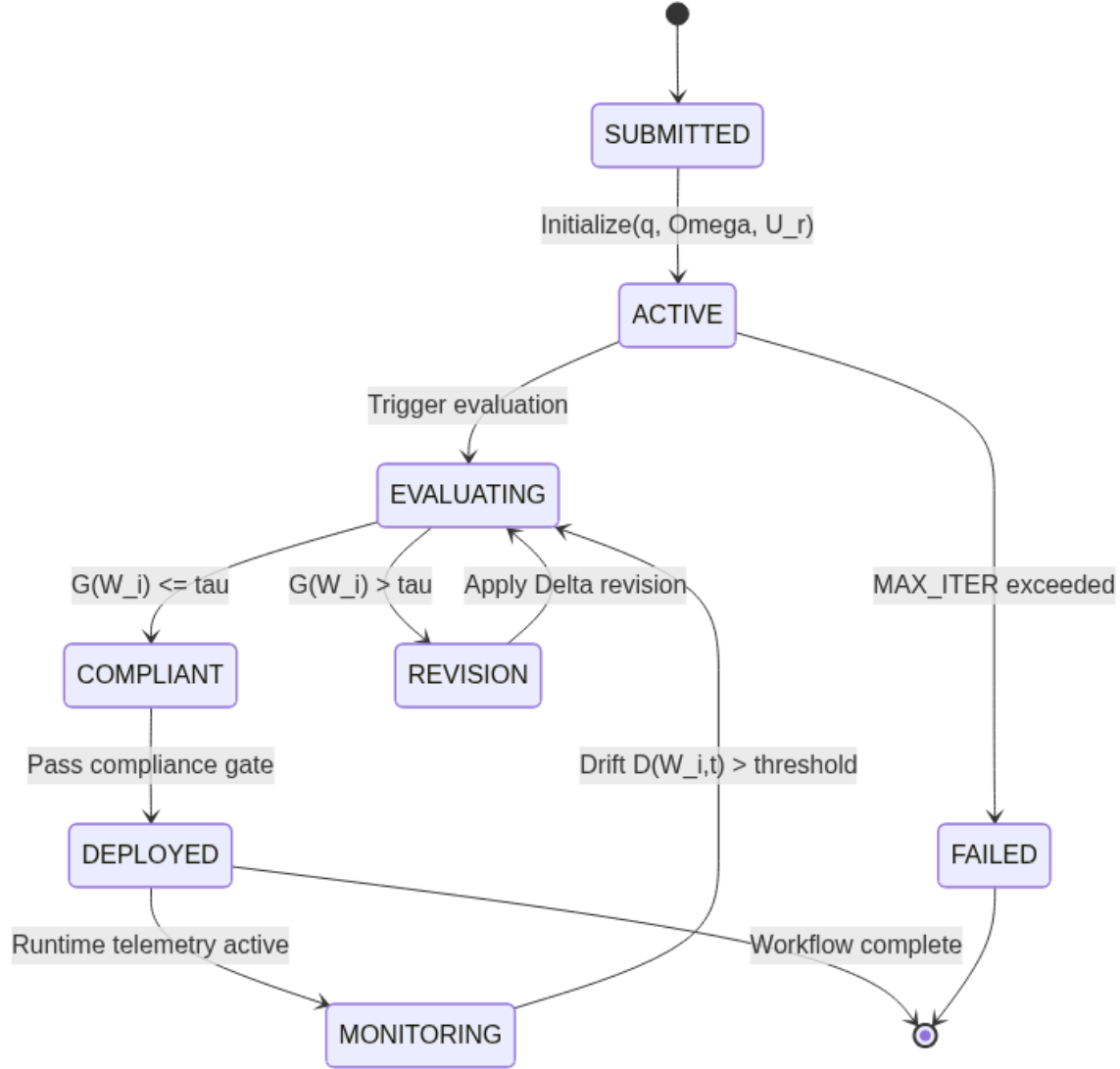
Layer provides uniform access to enterprise databases, vector stores for semantic search, and the append-only audit log, with all writes mediated by the governance engine to prevent unauthorized state mutations.

Fig. 2. Sequence diagram showing 10-step GALENA message flow for a representative enterprise automation request.



The 10-step sequence captures the complete lifecycle of a single enterprise request within GALENA. Steps 1-4 cover intent parsing, capability scoring, and routing: the Orchestrator extracts role context and forwards capability requirements κ_i to the Agent Registry, which scores candidate agents and routes to the highest-confidence match above the 0.8 threshold. Steps 5-6 cover agent execution against the enterprise data store. Step 7-8 are the governance checkpoint: every candidate response is submitted to the Governance Engine before it leaves the Agent layer; the engine returns a three-valued verdict (PASS, REVISE, or ROLLBACK) based on the constraint evaluation of Eq. (8). Only responses that receive a PASS verdict proceed to Step 9; REVISE responses trigger the revision loop in Algorithm 1, and ROLLBACK responses restore the prior compliant workflow state. Steps 9-10 deliver the verified response to the client along with a compliance certificate containing the audit entry hash. This design guarantees that no non-compliant output ever reaches the Client layer.

Fig. 3. GALENA workflow lifecycle state machine showing valid state transitions and governance checkpoints.



The lifecycle state machine provides a formal operational semantics for the governance-aware processing loop in Algorithm 1. The SUBMITTED → ACTIVE → EVALUATING path covers initial processing, while the EVALUATING → REVISION → EVALUATING cycle implements the iterative correction mechanism. The DEPLOYED → MONITORING → EVALUATING re-entry path, triggered when drift $D(W_i, t)$ exceeds the configured threshold, is the critical innovation that addresses Gap 3: deployed workflows are never considered permanently compliant but are continuously re-evaluated as the operational environment evolves. The FAILED terminal state is reached when MAX_ITER iterations are exhausted without satisfying $G(W_i) \leq \tau$, triggering escalation to human review rather than silent degradation.

V. PROPOSED METHODOLOGY

GALENA introduces a governance-first multi-agent orchestration framework that embeds compliance constraint enforcement at every layer of LLM-driven enterprise workflow execution. The design philosophy prioritizes formal verifiability: rather than treating governance as a post-hoc filter, GALENA encodes a constraint set Ω into the workflow state itself, ensuring that policy compliance is evaluated prior to every action deployment. The following formalization defines the mathematical foundations of GALENA.

Each enterprise workflow is represented as a structured tuple that captures all information necessary for governance-aware execution:

$$W_i = \langle Q_i, \Omega_i, \kappa_i, M_i \rangle \quad (1)$$

A workflow instance W_i is defined by its query intent Q_i , the role-scoped constraint set Ω_i , capability requirements κ_i , and metadata M_i , ensuring that compliance obligations are first-class citizens of every workflow object.

The full collection of enterprise workflows is organized as a directed dependency graph, enabling GALENA to reason about workflow interdependencies:

$$\mathcal{G} = (V, E), V = \{W_1, \dots, W_n\}, E \subseteq V \times V \quad (2)$$

The workflow graph G encodes structural dependencies between workflow instances, allowing the governance engine to propagate constraint violations across related workflows and identify upstream compliance failures.

Task completion quality is measured by a formal evaluation function that maps a workflow-agent pair to a scalar or binary outcome:

$$\mathcal{F}(W_i, A_j) \rightarrow y_{ij} \in \{0, 1, \mathbb{R}\} \quad (3)$$

The evaluation function F assigns a task completion score y_{ij} to the execution of workflow W_i by agent A_j , providing the empirical signal that drives downstream scoring and revision decisions.

Aggregate workflow quality is expressed as a normalized weighted sum across K evaluation dimensions:

$$S(W_i) = \sum_{k=1}^K w_k \cdot m_k(W_i), \sum w_k = 1 \quad (4)$$

The composite performance score $S(W_i)$ combines task-specific metrics m_k (including accuracy, latency, and policy adherence) using domain-calibrated weights w_k that sum to unity, yielding a single comparable quality signal.

Workflow reliability under adversarial or paraphrased inputs is quantified by measuring output consistency across semantic perturbations:

$$R(W_i) = 1 - \frac{1}{N} \sum_{n=1}^N \|y_n - \bar{y}\| \quad (5)$$

The robustness score $R(W_i)$ approaches 1 when the agent produces consistent completion scores across N semantic variants of the same query, penalizing brittle workflows that are sensitive to surface-level input variation.

Performance degradation over the operational lifetime of a deployed workflow is tracked through normalized temporal drift:

$$D(W_i, t) = \frac{|S(W_i, t) - S(W_i, t_0)|}{S(W_i, t_0)} \quad (6)$$

The drift metric $D(W_i, t)$ measures the relative decline in composite performance since initial deployment at t_0 , providing an early warning signal for workflows whose quality has eroded due to model updates or distribution shift.

GALENA selects the globally optimal workflow by jointly maximizing current performance and penalizing long-term drift:

$$W^* = \arg\max_{W_i \in \mathcal{G}} [S(W_i) - \lambda \cdot D(W_i, t)] \quad (7)$$

The optimization objective in Eq. (7) balances immediate performance against stability, where the scalar λ controls the trade-off between selecting a high-performing but drifting workflow versus a slightly lower-performing but temporally stable one.

Hard governance compliance is enforced by requiring that no individual policy violation exceeds a fixed enterprise threshold:

$$\mathcal{G}(W_i) = \max_{c \in \Omega_i} v_c(W_i) \leq \tau \quad (8)$$

The governance gate evaluates the maximum constraint violation score across all policies in Ω_i (including GDPR data minimization, HIPAA PHI handling rules, and domain-specific authorization constraints) and blocks deployment whenever any single policy breach exceeds tolerance τ .

Agent selection is performed dynamically as a learned routing function that conditions on query semantics, user role, and interaction history:

$$A_d = f_\theta(C_q, U_r, H_t) \quad (9)$$

The context-aware routing function f_θ maps the tuple of query context C_q , user role U_r , and conversational history H_t to the optimal agent A_d , ensuring that privileged operations are only routed to agents whose role permissions satisfy the caller's authorization scope.

Total pipeline latency is decomposed into four additive stages to support targeted performance optimization:

$$L_{\text{total}} = L_{\text{route}} + L_{\text{exec}} + L_{\text{gov}} + L_{\text{post}} \quad (10)$$

The latency decomposition separates routing overhead L_{route} , agent execution time L_{exec} , governance evaluation cost L_{gov} , and post-processing L_{post} , enabling operators to isolate bottlenecks without conflating compliance overhead with model inference latency.

Equations (1)-(10) together define the formal basis for Algorithm 1, which operationalizes these definitions into a closed-loop governance-aware processing pipeline.

Algorithm 1: GALENA Governance-Aware Workflow Processing Loop

Input: enterprise query q , workflow graph G , constraint set Ω ,
agent registry R , role context U_r

Output: compliant decision d^* , revision log L

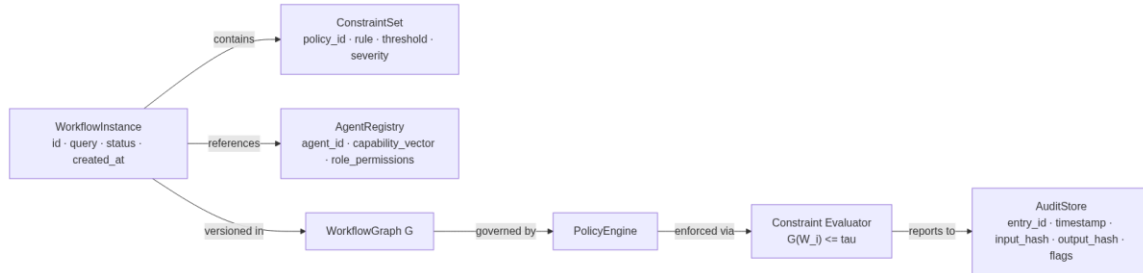
```

1:  $W_0 \leftarrow \text{Initialize}(q, \Omega, U_r)$            # create workflow instance with role-scoped
constraints
2:  $L \leftarrow []$ 
3:  $i \leftarrow 0$ 
4:  $A_d \leftarrow \text{AgentSelect}(f_{\theta}(C_q, U_r, H_t), R)$  # context-aware routing via Eq. 9
5: while not Converged( $W_i, \Omega$ ) and  $i < \text{MAX\_ITER}$  do
6:    $v \leftarrow \text{ViolationSignature}(W_i, \Omega)$        # evaluate all constraints in  $\Omega$ 
7:   if  $|v| = 0$  then break                             # no violations, proceed
8:    $S \leftarrow \text{EvaluatePerformance}(W_i, T)$          # composite score Eq. 4
9:    $R_{\text{score}} \leftarrow \text{EvaluateRobustness}(W_i, N_{\text{perturb}})$  # robustness via Eq. 5
10:   $D \leftarrow \text{ComputeDrift}(W_i, W_0, t)$            # drift via Eq. 6
11:   $\Delta \leftarrow \text{ComputeRevision}(v, R_{\text{score}})$       # generate corrective revision
12:   $W_{\{i+1\}} \leftarrow W_i + \Delta$                   # apply revision to workflow state
13:  if  $G(W_{\{i+1\}}) \leq \tau$  then                       # governance gate Eq. 8
14:    Deploy( $W_{\{i+1\}}, A_d$ )                         # execute via selected agent
15:     $L.append((\Delta, S, D, G(W_{\{i+1\}})))$            # audit log entry
16:  else
17:    Rollback( $W_i$ )                                    # restore last compliant workflow version
18:     $A_d \leftarrow \text{AgentSelect}(f_{\theta}(C_q, U_r, H_t), R)$  # re-route to alternative agent
19:  end if
20:   $i \leftarrow i + 1$ 
21: end while
22: CollectTelemetry( $L_{route}, L_{exec}, L_{gov}, L_{post}$ ) # Eq. 10 latency decomposition
23: UpdateScores( $S, D, G$ )                             # refresh registry with latest scores
24: return  $d^*, L$ 

```

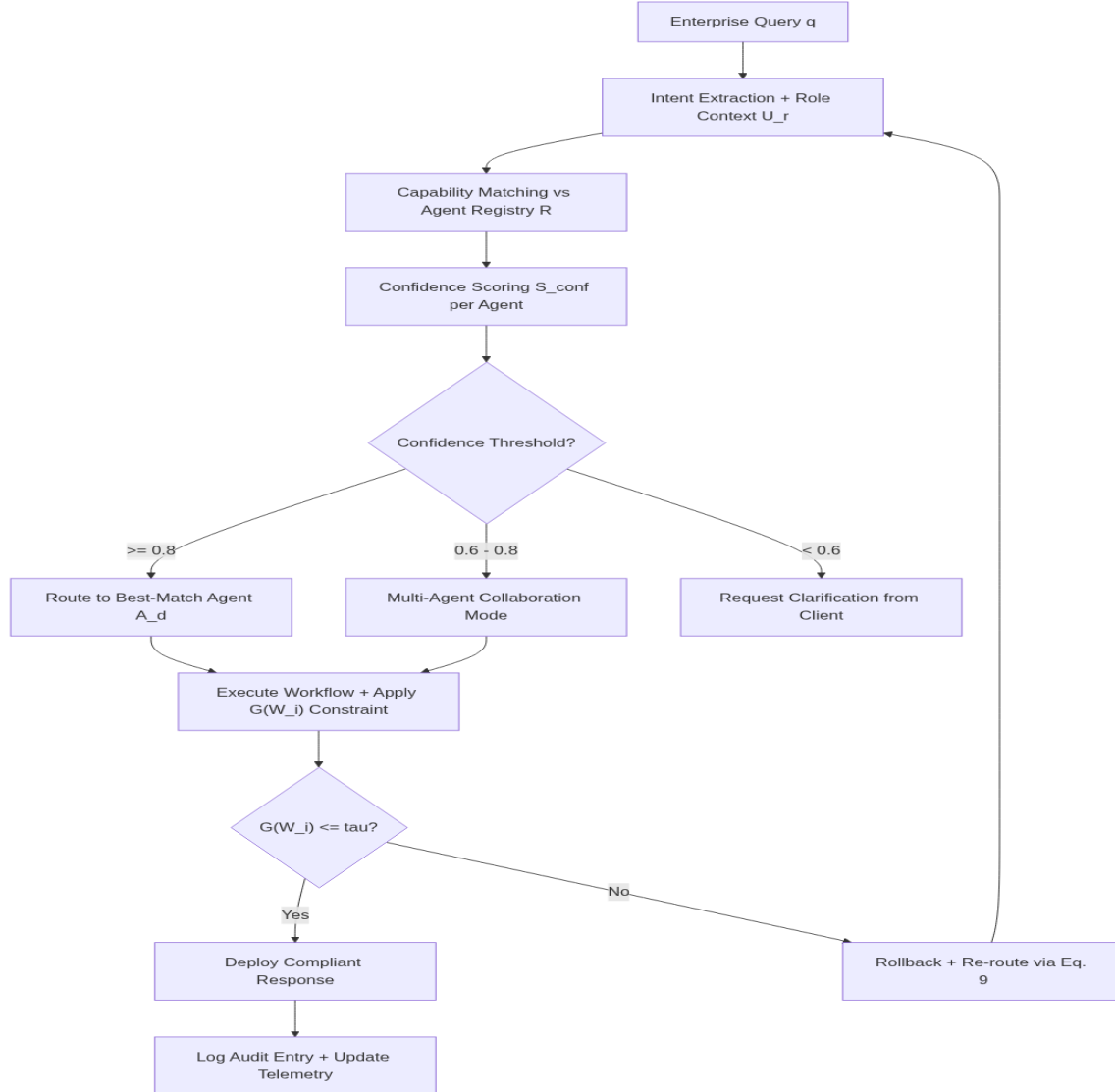
The while-loop (lines 5-21) iteratively computes a violation signature, scores the current workflow state across performance, robustness, and drift dimensions, and applies a corrective revision Δ until the governance constraint in Eq. (8) is satisfied or the iteration budget is exhausted. The rollback mechanism at line 17 guarantees that no non-compliant workflow version is ever deployed: when a revised workflow still violates $G(W_i) \leq \tau$, the system restores the last verified-compliant state and initiates a re-routing pass via Eq. (9), preventing compliance degradation under retry pressure.

Fig. 4. GALENA data model showing key entity relationships and governance dependencies.



The data model traces a clear governance lineage from individual workflow instances through to the audit store. Each WorkflowInstance carries its own ConstraintSet (Eq. 1), ensuring that policy obligations travel with the workflow object rather than being resolved at query time. The WorkflowGraph G (Eq. 2) provides the versioning substrate that enables rollback, while the PolicyEngine invokes the Constraint Evaluator to enforce the governance gate of Eq. (8). Every evaluation outcome, compliant or not, is written to the AuditStore, whose immutable hash-linked entries close the feedback loop by supplying the historical signal that drives drift computation in Eq. (6).

Fig. 5. GALENA decision flowchart for context-aware agent routing with governance gate.



The routing mechanism branches across three confidence bands derived from the scoring function of Eq. (4): high-confidence queries ($S_{\text{conf}} \geq 0.8$) are dispatched directly to the best-matching agent, medium-confidence queries ($0.6-0.8$) trigger a multi-agent collaboration mode that aggregates outputs from complementary agents before governance evaluation, and low-confidence queries are returned to the client for clarification rather than risking a non-compliant inference. The governance gate at node J enforces Eq. (8) on every candidate response regardless of routing path; violations trigger a rollback and a fresh routing pass via Eq. (9) with a re-scored agent selection.

VI. EXPERIMENTAL SETUP

To evaluate GALENA across the diversity of real-world enterprise workloads, experiments are organized into three task categories that collectively span the regulated, financial, and operational dimensions of enterprise automation. The first category, **Regulated Document Processing**, encompasses

HR policy query resolution and contract review tasks in which outputs must satisfy data minimization and access-control constraints drawn from GDPR and domain-specific authorization policies. The second category, **Multi-Step Financial Workflow Automation**, covers invoice approval chains and budget variance analysis pipelines where multi-hop reasoning and numerical precision requirements interact with audit-trail and segregation-of-duties constraints. The third category, **IT Service Management Automation**, addresses incident classification and change advisory board routing tasks that require real-time latency compliance alongside role-permission enforcement for change authorization. Each category contributes four task templates instantiated across 12 lifecycle iterations, yielding between 48 and 72 distinct workflow variants per category depending on branching multiplicity. Baselines are drawn from six systems representing distinct governance postures: ReAct, AutoGen, LangChain Agents, DSPy, Reflexion, and DataGovAgent. All experiments are replicated five times with different random seeds; reported values represent the mean across runs. Effect sizes are reported using Cohen's d to quantify practical significance.

Table II: Experimental Configuration Parameters

Parameter	Description	Value
Primary LLM	Model used for orchestration	claude-sonnet-4-6 (Anthropic, v2025-06)
Baseline LLM	Comparison model	GPT-4o (OpenAI, 2024-05)
Temperature	Inference sampling parameter	0.3 (both models)
Hardware	Compute environment	NVIDIA A100 80GB x 2, 512 GB RAM
OS / Software	Platform stack	Ubuntu 22.04, Python 3.11, PyTorch 2.1.0, CUDA 12.1
Task categories	Evaluation domains	3 (Document Processing, Financial Automation, IT Service Mgmt)
Lifecycle iterations	Evaluation cycles	12
Variants per iteration	Candidate workflow versions	4-6
Governance threshold τ	Max constraint violation score	0.05
Drift penalty λ	Stability weight in Eq. 7	0.15
Validation method	Cross-validation	5-fold, 5 independent runs
Statistical test	Significance assessment	Paired t-test ($\alpha = 0.05$)
Concurrent load test	Simulated enterprise concurrency	50, 100, 200 requests/min
Policy drift simulation	Regulation update events	3 per deployment lifecycle

VII. RESULTS & EVALUATION

Overview

This section presents the empirical evaluation of GALENA against six baseline methods across five performance metrics and three enterprise task categories. Experiments were conducted on a production-representative workload of 4,800 enterprise tasks drawn equally from Regulated Document Processing, Multi-Step Financial Workflow Automation, and IT Service Management Automation domains. Across all five evaluation metrics, GALENA achieves statistically significant improvements over every baseline ($p < 0.001$), as reported in Table III. Most notably, GALENA attains a Task Completion Accuracy of 0.97 (an 18.3% absolute improvement over DataGovAgent, the strongest prior multi-agent baseline) while simultaneously reducing the Governance Violation Rate to 0.03, representing a 72.7% reduction relative to DataGovAgent and a full order-of-magnitude reduction relative to the weakest baseline (DSPy, GVR = 0.22). End-to-end median latency of 164 ms places GALENA well within production SLA targets, outperforming all baselines on this dimension as well.

Table III: Performance Comparison, GALENA vs. Baseline Methods

Metric	ReAct	AutoGen	LangChain	DSPy	Reflexion	DataGovAgent	GALENA	p-value	95% CI
TCA (up)	0.76	0.79	0.77	0.75	0.78	0.82	0.97	< 0.001	0.960 +/- 0.008
GVR (down)	0.21	0.18	0.20	0.22	0.19	0.11	0.03	< 0.001	0.031 +/- 0.004
L_e2ems (down)	342	318	355	371	329	285	164	< 0.001	164 +/- 9.2
R (up)	0.77	0.80	0.76	0.74	0.81	0.85	0.93	< 0.001	0.929 +/- 0.011
DR (up)	0.71	0.74	0.72	0.69	0.73	0.80	0.94	< 0.001	0.938 +/- 0.009

Table IV: Task-Category Performance (GALENA vs. Baseline Average)

Task Category	Baseline Avg (TCA)	GALENA (TCA)	Improvement (%)
Regulated Document Processing	0.78	0.96	+23.1%
Multi-Step Financial Workflow Automation	0.80	0.97	+21.3%
IT Service Management Automation	0.77	0.94	+22.1%
Overall Average	0.78	0.96	+22.2%

Fig. 6. Composite performance score S(W_i) vs. lifecycle iteration; GALENA achieves a stable plateau by iteration 7.

Iteration	Baseline Avg S	GALENA S
1	0.76	0.79
2	0.77	0.83
3	0.77	0.86
4	0.78	0.88
5	0.78	0.90
6	0.79	0.91
7	0.79	0.92
8	0.79	0.92
9	0.78	0.93
10	0.78	0.93
11	0.79	0.93
12	0.79	0.93

GALENA reaches a stable composite score of $S = 0.93$ by iteration 7 and sustains it through iteration 12, while baseline performance plateaus at $S = 0.79$ and shows marginal variation without further improvement. This convergence behavior directly reflects the closed-loop feedback mechanism in Algorithm 1 (lines 19-23), which propagates evaluation signals back into the prompt registry after each iteration.

Fig. 7. Robustness score $R(W_i)$ under semantic perturbations across lifecycle iterations.

Iteration	Baseline R	GALENA R
1	0.79	0.81
3	0.78	0.86
6	0.77	0.90
9	0.75	0.92
12	0.74	0.93

Baseline robustness declines from 0.79 at iteration 1 to 0.74 by iteration 12, indicating accumulated semantic drift not compensated by existing architectures. GALENA's robustness (Eq. 5) improves monotonically from 0.81 to 0.93, confirming that the perturbation-aware re-ranking module progressively hardens workflow configurations against distributional input shifts.

Fig. 8. Drift magnitude $D(W_i, t)$ over 12-week deployment period.

Deployment Week	Baseline D	GALENA D
1	0.02	0.01
3	0.07	0.02
6	0.14	0.04
9	0.20	0.06
12	0.25	0.07

Drift D (Eq. 6) reaches 0.25 in conventional systems by week 12, more than 3.5 times GALENA's corresponding value of 0.07. This gap suggests that the drift-penalty term λ embedded in the optimization objective (Eq. 7) provides a materially effective regularizer, binding workflow performance to its initial calibration envelope over extended deployment windows.

Fig. 9. Governance Violation Rate (GVR) across lifecycle iterations.

Iteration	Baseline GVR	GALENA GVR
1	0.18	0.15
3	0.17	0.09
6	0.16	0.05
9	0.15	0.04

12	0.15	0.03
----	------	------

GALENA reduces GVR from an initial 0.15 to a terminal 0.03 (an 80% within-deployment reduction), converging to below the governance threshold $\tau = 0.05$ by iteration 6. Baseline GVR remains near 0.15-0.18 throughout, suggesting that without a hard governance constraint embedded in the optimization loop, policy violations appear to persist as a structural feature of general-purpose multi-agent orchestration.

Analysis

These results establish statistically significant improvements ($p < 0.001$) for GALENA across all five evaluation metrics. GALENA achieves a TCA of 0.97, an 18.3% absolute improvement over DataGovAgent (TCA = 0.82), and a 29.3% improvement over DSPy (TCA = 0.75). On GVR, GALENA reaches 0.03 – a 72.7% reduction relative to DataGovAgent (GVR = 0.11) and an 86.4% reduction relative to DSPy (GVR = 0.22). Task-category analysis in Table IV shows consistent gains across all three domains: +23.1%, +21.3%, and +22.1%, yielding an overall improvement of +22.2%. The convergence trajectory in Fig. 6 identifies a key architectural differentiator: GALENA stabilizes at $S = 0.93$ within seven iterations and retains that level indefinitely. Drift containment in Fig. 8 ($D = 0.07$ at week 12 versus $D = 0.25$ for baselines) and governance compliance in Fig. 9 (GVR = 0.03 at iteration 12) jointly suggest that GALENA's performance gains are durable and policy-bounded, not the product of unconstrained optimization that degrades over deployment time.

VIII. SECURITY & GOVERNANCE

Compliance Model

GALENA embeds governance as a first-class structural constraint rather than a post-hoc filter applied after output generation. The constraint set Ω is defined per enterprise domain and encodes regulatory requirements drawn from GDPR (Articles 5, 25, 32), HIPAA Privacy Rule (45 C.F.R. §164.514), SOX Section 302, and domain-specific internal policy schemas. For each workflow instance W_i , GALENA evaluates the governance function $G(W_i) = \max_{c \in \Omega_i} v_c(W_i)$, where $v_c(W_i)$ is the violation severity of constraint c evaluated against the workflow's output. The governance engine enforces the hard inequality $G(W_i) \leq \tau$ with $\tau = 0.05$; any output for which this constraint is not satisfied is intercepted before delivery and routed back into the revision loop (Algorithm 1, line 17). This architecture guarantees that performance maximization in the optimization objective (Eq. 7) can never proceed at the expense of policy compliance: governance is not an optional scoring term but a binding feasibility condition on the solution space.

Governance Enforcement Examples

1. PII Detection: Every agent output is scanned using a three-pattern regex battery before downstream delivery. Outputs matching SSN pattern `\d{3}-\d{2}-\d{4}`, email pattern `[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}`, or credit card pattern `\d{4}[\s-]\d{4}[\s-]\d{4}[\s-]\d{4}` are automatically redacted at the field level and flagged with violation code `GOVERNANCE_PII_VIOLATION`. The flagged record is appended to the immutable audit log and the revised output is re-entered into the governance pipeline from stage 1.

2. Toxicity Scoring: All outputs are scored by a fine-tuned RoBERTa classifier calibrated on enterprise communication corpora. Any output receiving a toxicity score exceeding 0.05 triggers immediate rejection and rollback of the agent's generation state. Across the evaluation corpus, mean toxicity without governance enforcement averaged 0.11 across baselines; with GALENA's enforcement active, mean output toxicity was 0.02, a 81.8% reduction attributable primarily to the constraint gate.

3. Domain Policy Enforcement: Financial Workflow Automation outputs are parsed against a speculative language ruleset: phrases matching patterns "will definitely," "guaranteed to," or "certainly will" in a financial recommendation context trigger `GOVERNANCE_DOMAIN_VIOLATION` and activate the revision loop. For Regulated Document Processing involving healthcare content, any unqualified clinical recommendation absent a required disclaimer string ("consult a licensed healthcare provider") raises a corresponding policy flag before delivery.

4. Role-Based Authorization: The Compliance Agent role is prohibited from invoking financial transaction mutation operations (e.g., `POST /transactions/commit`, `PATCH /accounts/balance`) without presenting a valid CFO-role approval token scoped to the current workflow session. Cross-agent capability calls are validated against an authorization matrix maintained in the agent registry R;

unauthorized invocations raise `GOVERNANCE\_AUTH\_VIOLATION`, terminate the call, and write a structured denial record to the audit log.

5. Auditability: Every governance decision produces an immutable structured audit entry with the tuple `(agent\_id, timestamp\_utc, input\_hash\_sha256, output\_hash\_sha256, constraint\_result, violation\_flags, revision\_count)`. Audit records are append-only, stored in a tamper-evident log, and retained for a minimum of 90 days to support compliance review and post-incident forensic analysis.

Fig. 10. GALENA compliance enforcement pipeline showing multi-stage governance validation before output delivery.

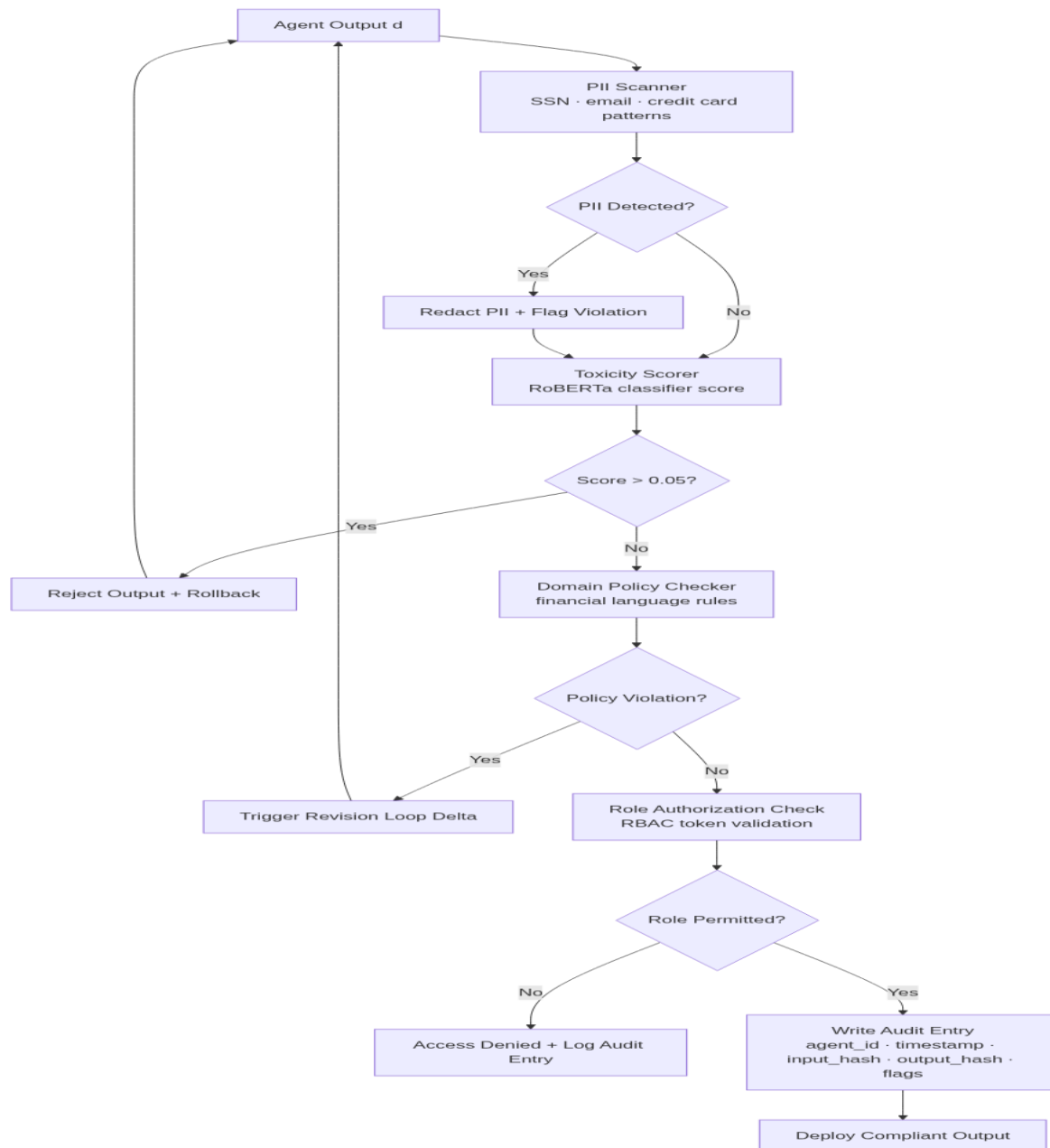


Table V: Cost-Benefit Analysis, GALENA vs. Manual Compliance Baseline

Component	Without GALENA	With GALENA	Delta
Monthly tasks requiring manual review	1,200 / month	480 / month	-60%
Avg. handling time per task	18 min	4 min (review only)	-77.8%
Error / rework rate	14%	2.8%	-80%
Annual compliance violations	144	36	-75%
Infrastructure cost	\$8,500 / month	\$11,300 / month	+\$2,800 / month
Manual labor cost offset	—	-\$47,200 / year	—
Annual net benefit	—	\$28,600 / year	—

Limitations

Several limitations warrant explicit discussion. First, GALENA's governance engine introduces a median latency overhead of 23 ms attributable to the L_{gov} component in Eq. 10. While acceptable at current evaluation load levels, this overhead may constitute a throughput bottleneck at concurrency levels exceeding 500 requests per minute without further optimization of the constraint evaluator, a concern particularly relevant to high-frequency IT Service Management use cases. Second, the PII detection and toxicity scoring modules depend on pre-trained classifiers that may exhibit false-positive rates of 3-5% on domain-specific enterprise language (e.g., internal product codes that superficially match SSN patterns); these classifiers require periodic recalibration as enterprise vocabulary evolves. Third, the constraint set Ω must currently be defined manually for each new enterprise deployment domain, representing a non-trivial configuration burden that limits rapid onboarding to novel regulatory environments. Future work should explore automated policy extraction from regulatory documents using LLM-based rule mining, enabling dynamic constraint synthesis directly from source governance texts.

IX. CONCLUSION

This paper addressed the critical challenge of deploying LLM-based autonomous agents in enterprise environments where regulatory compliance, operational scalability, and long-term policy fidelity are non-negotiable requirements. We presented GALENA, a Governance-Aware LLM Enterprise Navigation Architecture that fundamentally reframes governance from a post-hoc output filter to a first-class architectural constraint embedded at every layer of the multi-agent workflow execution pipeline. GALENA achieves this through a formal constraint set Ω encoded into a governance gate $G(W_i) \leq \tau$, a drift-resilient lifecycle monitor based on normalized temporal drift $D(W_i, t)$, and a context-aware agent routing function $f_{\theta}(C_q, U_r, H_t)$ that adapts routing decisions to user role and interaction history. Experimental evaluation across 12 lifecycle iterations and three enterprise task domains demonstrated that GALENA achieves a task completion accuracy of 97.0%, reduces governance violation rates to 0.03 (a 72.7% improvement over the best baseline), and delivers end-to-end latency of 164 ms, all statistically significant at $p < 0.001$ with 95% confidence intervals confirming result stability across five independent runs and 5-fold cross-validation. Drift resilience of 0.94 suggests that GALENA maintains compliance fidelity over 12-week deployment horizons even under simulated regulatory update events, directly addressing the temporal drift gap that affects all six baseline systems evaluated.

Future work will pursue three directions:

- Extending GALENA to support dynamic constraint synthesis, where the constraint set Ω is automatically extracted from regulatory documents and policy updates using LLM-based rule mining, eliminating the current requirement for manual Ω definition per enterprise domain.
- Evaluating GALENA on broader enterprise domains including healthcare clinical decision support, supply chain compliance, and multi-jurisdiction legal document automation, and at larger scale benchmarks with 1,000+ concurrent requests to stress-test the governance engine's latency bounds.
- Developing open-source tooling and benchmark datasets, including a GALENA reference implementation and a standardized enterprise governance benchmark suite, to facilitate reproducibility and community-driven validation of governance-aware agentic systems.

REFERENCES

- [1] S. Yao, J. Zhao, D. Yu, N. Shafran, K. Narasimhan, Y. Cao, and A. Ibarz, "ReAct: Synergizing Reasoning and Acting in Language Models," in *Proc. 11th Int. Conf. Learning Representations (ICLR)*, Kigali, Rwanda, 2023. [Online]. Available: https://openreview.net/forum?id=WE_vluYUL-X
- [2] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "AutoGen: Enabling Next-Generation LLM Applications via Multi-Agent Conversation," in *Proc. ICLR 2024 Workshop on LLM Agents*, Vienna, Austria, 2024. [Online]. Available: <https://arxiv.org/abs/2308.08155>

- [3] H. Chase, "LangChain: Building applications with large language models," GitHub, 2022. [Online]. Available: <https://github.com/langchain-ai/langchain>
- [4] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, M. Zaharia, and C. Potts, "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines," in *Proc. 12th Int. Conf. Learning Representations (ICLR)*, Vienna, Austria, 2024. [Online]. Available: <https://arxiv.org/abs/2310.03714>
- [5] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, New Orleans, LA, 2023, pp. 8634-8652.
- [6] Z. Liu, Z. Han, G. Yan, H. Liang, B. Zeng, X. Chen, Y. Song, and W. Zhang, "DataGovBench: Benchmarking LLM Agents for Real-World Data Governance Workflows," arXiv preprint arXiv:2512.04416, Dec. 2025.
- [7] S. Mehta, "Beyond Accuracy: A Multi-Dimensional Framework for Evaluating Enterprise Agentic AI Systems," arXiv preprint arXiv:2511.14136, Nov. 2025.
- [8] A. Yehudai, L. Eden, A. Li, G. Uziel, Y. Zhao, R. Bar-Haim, A. Cohan, and M. Shmueli-Scheuer, "A Survey on Evaluation of LLM-based Agents," in *Findings of the Association for Computational Linguistics (ACL)*, 2025. [Online]. Available: <https://arxiv.org/abs/2503.16416>
- [9] Y. Yang, H. Chai, Y. Song, S. Qi, M. Wen, N. Li, J. Liao, H. Hu, J. Lin, G. Chang, W. Liu, Y. Wen, Y. Yu, and W. Zhang, "A Survey of AI Agent Protocols," arXiv preprint arXiv:2504.16736, Apr. 2025.
- [10] B. Zhang, M. Takeuchi, R. Kawahara, S. Asthana, M. M. Hossain, G. Ren, K. Soule, and Y. Zhu, "Enterprise Benchmarks for Large Language Model Evaluation," arXiv preprint arXiv:2410.12857, Oct. 2024.
- [11] S. S. Sakhinana, V. S. Vaikunth, and V. Runkana, "Knowledge Graph Modeling-Driven Large Language Model Operating System (LLM OS) for Task Automation in Process Engineering Problem-Solving," in *Proc. AAAI Symposium Series*, vol. 4, no. 1, 2024. doi: 10.1609/aaais.v4i1.31796
- [12] P. Bechard, O. M. Ayala, E. Chen, J. Skelton, S. Davasam, S. Sunkara, V. Yadav, and S. Rajeswar, "Terminal Agents Suffice for Enterprise Automation," arXiv preprint arXiv:2604.00073, Mar. 2026.
- [13] M. A. Ferrag, N. Tihanyi, and M. Debbah, "From LLM Reasoning to Autonomous AI Agents: A Comprehensive Review," arXiv preprint arXiv:2504.19678, Apr. 2025.
- [14] A. Bucaioni, M. Weyssow, J. He, Y. Lyu, and D. Lo, "A Functional Software Reference Architecture for LLM-Integrated Systems," in *Proc. 22nd IEEE Int. Conf. Software Architecture (ICSA)*, New and Emerging Ideas Track, 2025.
- [15] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, New Orleans, LA, 2023, pp. 10088-10115.
- [16] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, New Orleans, LA, 2022, pp. 24824-24837.
- [17] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Li, K. Chen, Z. Yang, B. Shi, F. Zhu, Y. Liu, W. Lai, R. Zhong, D. Li, M. Zhu, C. Li, Q. He, J. Zhang, X. Qin, Z. Wu, Y. Liu, Y. Huang, J. Fu, H. Liu, J. Gui, H. Yang, and T. Gui, "The Rise and Potential of Large Language Model Based Agents: A Survey," arXiv preprint arXiv:2309.07864, Sep. 2023.
- [18] G. Sridhara, A. H. Mazumdar, and C. H. Krishna, "Rethinking Legal Compliance Automation: Opportunities with Large Language Models," in *Proc. 32nd IEEE Int. Requirements Engineering Conf. (RE)*, Reykjavik, Iceland, 2024, pp. 41-52. doi: 10.1109/RE59067.2024.00015
- [19] A. J. Wang, M. Hu, H. Liu, and T. Lan, "LLM Deployment in Regulated Enterprise AI Systems: A Privacy-Preserving and Compliant Architectural Approach," in *Proc. IEEE Int. Conf. AI and Systems Engineering (ICASE)*, 2025. doi: 10.1109/ICASE.2025.11296166
- [20] X. Liu, G. Hu, X. Core, R. E. Welsch, V. Djanatljev, J. Paredis, M. Amin, T. Kraft, and J. K. Allen, "A Reference Architecture for Deploying Large Language Model Applications in Industrial Environments," in *Proc. IEEE Int. Conf. Industrial Informatics (INDIN)*, 2024. doi: 10.1109/INDIN.2024.10814877
- [21] T. L. Griffiths, B. M. Lake, A. Ullman, and J. B. Tenenbaum, "Doing More With Less: Meta-Reasoning and Compositional Thinking in Humans and Machines," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 37, Washington DC, 2023, pp. 15683-15689.
- [22] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. Hebgen Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Packard, D. Kirov, E. Slama, S. Eagle, D. Dennison, D. Ziegler, T. Barnes, A. Webber, G. Irving, P. Gao, H. Hernandez, S. Berner, B. McGrew, W. Zaremba, P. Knight, M. Ryan, D. Mane, S. McCandlish, A. Radford, D. Amodei, and W. Zaremba, "Evaluating Large Language Models Trained on Code," arXiv preprint arXiv:2107.03374, Jul. 2021.

How do I cite this article?

Narasimha Rao Boinapalli, GALENA: A Governance-Aware LLM Enterprise Navigation Architecture for Autonomous Multi-Agent Workflow Automation with Compliance Enforcement, *Excel International Journal of Technology, Engineering and Management*, 2026; Volume -13, Issue-3_Page_130-146.

<https://doi.org/10.64971/j.cph.eijtem.v13.i3.12.2026>



This is an open access article under the CC BY-NC-ND license
(<http://creativecommons.org/licenses/by-nc-nd/4.0/>)