Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Explainable Agentic Decision Support for Project Governance in Agile–DevOps: A Multi-Agent Governance Framework for Project Managers

Suresh Kandasamy
Dayananda Sagar University
Suresh Arumugam
Dayananda Sagar University
Cynthia Jayapal
cynthiaj@rvu.edu.in

RV University

Article

Keywords:

Posted Date: July 8th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9834249/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Explainable Agentic Decision Support for Project Governance in Agile–DevOps: A Multi-Agent Governance Framework for Project Managers

Suresh Kandasamy[1], Suresh Arumugam[2], and Cynthia Jayapal[3,*]

[1]Department of AI & ML, Dayananda Sagar University, Harohalli, Bengaluru 562112, India, and DBA in Emerging Technologies with a Concentration in Generative AI, Golden Gate University, San Francisco, CA 94105, USA
[2]Department of CSE–Data Science, Dayananda Sagar University, Harohalli, Bengaluru 562112, India
[3]School of Computer Science and Engineering, RV University, Bengaluru 560059, India
[*]cynthiaj@rvu.edu.in

ABSTRACT

Modern Agile–DevOps environments continuously produce operational data through deployment pipelines, observability platforms, cloud infrastructure, and security tooling. While this telemetry is critical for governance activities, the information is often distributed across multiple operational domains and represented through different technical perspectives. As a result, Project Managers may struggle to interpret reliability issues, release risks, infrastructure costs, and security concerns in a unified and actionable manner. The disconnect between low-level operational telemetry and governance-level interpretation remains a persistent challenge in Agile–DevOps project management.This paper introduces the AgileOps Agentic Framework (AAF), a governance-oriented multi-agent decision-support framework designed to translate fragmented operational evidence into explainable governance recommendations. The framework combines specialised reasoning agents representing DevOps, Site Reliability Engineering (SRE), FinOps, and DevSecOps with consensus evaluation, Re-Grounded Agentic Reasoning (RAR) for uncertainty handling, utility-based governance scoring, and evidence-grounded explanation generation. Rather than performing autonomous remediation, AAF focuses on supporting managerial decision making through traceable and operationally grounded governance interpretation.The framework was evaluated using three complementary studies: (1) controlled cross-domain governance scenarios, (2) Project Manager-oriented prompt evaluations, and (3) a runtime telemetry pilot using operational artifacts collected from the Sock Shop microservices benchmark. In the controlled evaluation involving 120 curated governance cases, the full AAF configuration achieved a domain-match rate of 0.875 and governance-action match of 0.792, outperforming baseline and ablation configurations. During the Project Manager prompt evaluation using 100 governance-oriented prompts, the framework achieved a 0.78 domain-match rate in blind evidence mode and full domain consistency under aligned evidence conditions, while maintaining strong acceptable governance-action alignment across ambiguous operational scenarios. A grounded explanation evaluation further showed that evidence-constrained explanation generation could preserve domain consistency, action consistency, and evidence coverage across evaluated cases. The runtime telemetry pilot, which included 20 controlled runtime cases and 733 collected operational artifacts, demonstrated that heterogeneous runtime evidence could be transformed into structured governance interpretations under controlled conditions.Collectively, the results suggest that integrating domain-specific agentic reasoning with utility-weighted governance evaluation, live telemetry processing, and evidence-constrained explanation generation holds promise for advancing decision support at the project-governance level in Agile–DevOps contexts. A further contribution of this work lies in the reproducible experimental pipeline it establishes—one that creates an explicit, auditable link between raw operational telemetry and governance-oriented interpretations intended for Project Managers.

Introduction

Software delivery in Agile and DevOps environments is very fast and complex. All the tools supporting the software delivery pipeline for a company generate operational telemetry data which can be used to optimise the software delivery process. Such telemetry data is produced by CI and testing tools, cloud-native pipelines and workflows, observability and monitoring tools, and automated security tools, to name a few. These tools facilitate fast software delivery and handle unprecedented complexity in operational software systems[1–4]. The telemetry data, however, poses governance problems that are hard to manage. The data is scattered across multiple tools and applications, and provides very different insights for development teams, reliability teams, infrastructure teams, cost-management teams, and security teams. Project Managers currently lack visibility into release readiness, operational risk, infrastructure cost, and security posture.

The telemetry data that flows through the stack is what governance decisions are made from. Are features not meeting

goals going to cause delayed releases to customers? Is the total cost of owning a service increasing because of factors such as aggressive auto-scaling and increased usage of spot instances? A large number of telemetry signals are surfaced for engineering and platform teams to operate on a day-to-day basis. Those signals also allow for additional insights to inform governance decisions related to release timing. Unfortunately, current processes are often slow and limit the insights that can be gathered across key aspects of delivery.

Although many research studies and operational practices have been conducted for automation and reliability management in DevOps and SRE environments[5–8], methods and techniques related to deployment automation, observability pipeline setup and management, incident response, anomaly detection, and operational analytics have already been extensively discussed. Many AIOps studies, such as machine-learning-based log analysis, event correlation, and operational support systems, have also been explored[9–11]. Practices such as FinOps and DevSecOps, which extend governance into cloud-cost management and security operations, have recently gained significant attention[12–15]. However, most of the existing approaches remain highly focused on engineering operations and provide limited managerial interpretation. There is comparatively little emphasis on integrating insights from multiple operational perspectives to support Project Manager decision making under uncertain conditions.

Although explainability in general-purpose LLMs remains an open challenge[16–18], many operational reasoning and explanation capabilities have become increasingly relevant and practically achievable within the context of multi-agent systems, Agentic AI, and LLMs[19,20]. Grounded language generation further improves explainability by ensuring that the output of a language model remains tightly linked to structured evidence[21]. However, generating operationally explainable LLM-based systems is far from trivial and introduces both technical and governance challenges related to model reliability, infrastructure operations, security, and release management.

Although many approaches, solutions, and tools have been proposed to address these challenges, there remains a strong need for governance-oriented approaches, processes, and tools that can operate effectively within Agile–DevOps environments. This paper presents the AgileOps Agentic Framework (AAF), a governance-oriented decision-support framework designed for modern software delivery, deployment, runtime, and lifecycle operations. The AAF consists of four reasoning agents: DevOps RA, Site Reliability Engineering RA, FinOps RA, and DevSecOps RA. Each reasoning agent generates governance claims, corresponding confidence levels, and summaries of supporting evidence computed from operational telemetry data. The utility of proposed governance actions is evaluated in terms of performance improvement, cost efficiency, and risk reduction. The Cross-domain agreement between governance claims is also evaluated. When agreement falls below a predefined threshold, additional evidence is processed through re-ground agentic reasoning (RAR) processes to improve agreement and reduce uncertainty. The explanation layer produces evidence-based governance summaries intended to support Project Managers through operational interpretation and governance traceability.

AAF is not intended to function as a standalone remediation system. Running AAF does not require separate infrastructure or independent operational environments to address the identified issues. Instead, the purpose of AAF is to help Project Managers understand operational conditions, identify cross-cutting issues, and guide governance responses in uncertain delivery environments. The AAF is intended to improve governance interpretation by helping Project Managers identify appropriate governance responses under different operational conditions.

We conducted three separate studies to validate the feasibility and usability of the proposed framework. In the first study, we applied the framework to cross-domain governance scenarios focusing on governance interpretation, governance-action selection, agreement evaluation, explainability, and usability. The second and third studies evaluated Project Manager-oriented prompts generated by the framework in different operational contexts. These studies also incorporated a pilot telemetry using the Sock Shop microservices benchmark application[22]. We generated 20 runtime cases with an average of 36.9 operational data objects per case, resulting in a total of 733 artifacts. The framework successfully collected telemetry artifacts that included running-container information, runtime log messages, deployment metadata, service-query results, and resource-consumption metrics. Together, these studies provide a reproducible basis for examining governance-related operational decisions.

The primary contributions of this paper are summarized as follows:

- A governance-oriented multi-agent system integrating DevOps, Site Reliability Engineering, FinOps, and DevSecOps reasoning for Agile–DevOps operational governance.
- A structured consensus and Re-Grounded Agentic Reasoning mechanism for managing uncertainty across multiple operational domains.
- A utility-based governance evaluation approach that incorporates performance improvement, cost efficiency, and risk reduction into governance-action selection.
- An evidence-based explainability layer that supports the operational interpretation of the project manager with improved governance traceability.

- A reproducible multi-stage evaluation framework supporting governance-scenario analysis, Project Manager prompt evaluation, grounded explanation assessment, and telemetry-pilot experimentation using microservices-based runtime environments.

The remainder of this paper is organized as follows. Section II reviews related work in DevOps governance, AIOps, multi-agent systems, explainable AI, and grounded language reasoning. Section III presents the architecture and workflow of the AgileOps Agentic Framework. Section IV describes the experimental methodology and evaluation design. Section V presents and discusses the experimental results. Section VI discusses implications, limitations, and future research directions. Section VII concludes the paper.

Related Work

DevOps, SRE, and Operational Governance

The operational governance of software systems is significantly impacted by DevOps practices that have fundamentally changed the way software systems and their production infrastructure are built and maintained over the last decade [1–4]. Many practices such as continuous integration, automated deployment, continuous delivery, production monitoring, continuous observability, and automated incident management heavily rely on automation. Automated procedures process telemetry into operational evidence that informs governance decisions. The systems producing this telemetry—deployment pipelines, observability tools, cloud and virtualised infrastructure, and incident-management platforms—have themselves become increasingly automated. This paper explores how operational governance is influenced not only by automation itself, but also by the complex operational evidence generated across modern production infrastructures.

Site Reliability Engineering (SRE) extends traditional reliability engineering practices into the day-to-day operation of software systems through service-level objectives (SLOs), observability pipelines, incident-response procedures, and fault-management strategies[5,6]. In this paper, particular attention is given to cloud-native and microservices environments, where applications consist of multiple independently deployable services with continuously evolving runtime interactions[23]. Monitoring such systems requires mechanisms capable of gathering, storing, and analysing telemetry data collected from multiple operational sources.

Anomaly detection, log mining, and reliability monitoring for root-cause analysis have been approached using both statistical and machine-learning techniques. Systems such as DeepLog[9] and Drain[10] demonstrate how operational logs can be analysed to identify abnormal system behaviour and support operational diagnostics. Trace-based approaches and root-cause localisation in distributed and microservices systems have also attracted significant attention[8,24,25]. However, while substantial progress has been made in improving the ability of infrastructure operations teams to detect and respond to operational issues, comparatively less work has focused on transforming operational evidence into governance-oriented recommendations suitable for Project Managers and delivery stakeholders.

AIOps and Intelligent Operational Support

Artificial Intelligence for IT Operations (AIOps) has emerged as an active research area for operational monitoring, anomaly detection, fault diagnosis, infrastructure optimisation, and intelligent operational support[11]. Many AIOps systems apply machine-learning techniques to correlate events, identify failures, predict anomalies, and guide operational actions. With the rapid growth of native cloud systems and large-scale telemetry streams, interest in AIOps has continued to increase. In this paper, we outline an initial effort toward a governance-oriented AIOps framework for monitoring cloud-native microservices environments.

Rapid developments have been reported in commercial and research-oriented AIOps platforms. However, most existing solutions primarily support infrastructure operators, Site Reliability Engineering (SRE) teams, and platform-engineering teams. Although these systems provide automation and remediation capabilities, explainability and governance-oriented interpretation often remain secondary concerns. As a result, Project Managers may still struggle to interpret operational telemetry in a form suitable for governance-level decision making.

Unlike many existing AIOps approaches that focus on infrastructure orchestration and autonomous remediation, the framework presented in this paper focuses on governance interpretation. Rather than automatically executing operational changes, the framework generates evidence-based and explainable governance recommendations intended to support managerial decision making under uncertain operational conditions.

FinOps and DevSecOps Integration

As the cloud-native delivery environment matures, effective governance is needed to reconcile different perspectives and priorities. Practices like FinOps provide teams with cloud-cost visibility, financial accountability, and optimization techniques

to ensure a reliable and cost-effective use of cloud resources[12,26,27]. DevSecOps practices incorporate security and compliance-based best practices into CI/CD pipelines and other workflows[13–15]. The CIO and engineering leaders must weigh reliability, performance, cost, security, and time constraints in their governance decisions.

Scaling decisions are typically made within individual technical domains. Allowing a service auto-scale to increase to mitigate reliability issues may cost more than intended in the cloud. Decisions to delay deployment to fix a security issue have implications for scheduled delivery dates and release commitments. Current operational tooling allows decisions to be made within individual technical domains, but does not facilitate understanding of the governance implications across operational domains.

This paper extends the operational reasoning applied by DevOps/SRE teams to also consider the FinOps and DevSecOps dimensions. We provide governance recommendations that involve cross-domain operational trade-offs and go beyond typical focus on optimizing for infrastructure-related metrics.

Multi-Agent Systems and Agentic AI

Multi-agent systems for distributed reasoning, coordination, and decision making have been studied for a long time[28–31]. With the emergence of Agentic AI and large language models (LLMs), decomposing complex tasks into specialized agents has gained renewed attention[19,20]. In governance-oriented operational environments, different objectives often require different forms of expertise and evaluation.

Recent work has applied LLMs to building agents for planning, orchestration, collaborative reasoning, operational support, and modeling the behavior of other agents[32]. These systems have been explored in areas such as task planning, decision making, argument generation, interactive reasoning, and state-space exploration. However, many of these approaches rely heavily on unconstrained natural-language generation, making generated plans difficult to verify and increasing the risk of unsupported outputs, commonly referred to as hallucinations[33]. In governance-oriented operational environments, such hallucinated outputs may have serious implications for release management, reliability, infrastructure operations, and information security.

In this work, structured operational reasoning is separated from language-model-based explanation generation. Domain interpretation, consensus evaluation, and governance scoring are handled independently of the explanation layer in order to improve traceability and reduce unsupported claims.

Explainability, Grounding, and Hallucination Control

Recent research in retrieval-augmented generation, explainable AI, and hallucination analysis shares a common objective: ensuring that AI-generated outputs remain grounded in verifiable evidence[21]. Previous studies have shown that unconstrained language generation can introduce unsupported or misleading statements, particularly in operational environments where accountability is critical[33]. These concerns become even more significant when AI systems contribute to governance-related decisions that require transparency, traceability, and operational accountability.

There is also increasing interest in explainable AI, particularly in the context of human-centered decision-support systems[16–18]. Explainability is especially relevant in Agile–DevOps governance environments, where Project Managers are often required to justify decisions related to release management, reliability, security, and operational cost across multiple stakeholder groups.

To address these concerns, the framework presented in this paper incorporates an evidence-based explanation layer that constrains language-model outputs to structured AAF evidence, including domain claims, supporting evidence, confidence values, utility scores, and consensus results. The explanation layer improves readability, transparency, and evidence traceability while reducing the likelihood of unsupported operational claims.

Research Gap

Although there have been significant advances in DevOps automation, observability, AIOps, cloud governance, operational analytics, and intelligent operational support[1–6,8–15,23–27], comparatively limited work has focused specifically on governance-oriented operational interpretation for Project Managers operating within Agile–DevOps environments. Most existing research continues to emphasize infrastructure automation, anomaly detection, incident response, and engineering-centric operational workflows. Comparatively fewer studies investigate how operational evidence from multiple domains can be consolidated into explainable governance recommendations that remain reproducible, traceable, and useful for the support of managerial decisions[16–21,33].

The AgileOps Agentic Framework addresses this gap by combining multi-agent operational reasoning, structured uncertainty handling, utility-based governance evaluation, and evidence-grounded explanation generation within a governance-oriented decision-support architecture. Unlike many autonomous operational systems, the proposed framework is designed to support Project Managers by helping them interpret operational telemetry and evaluate governance trade-offs under controlled but realistic operational conditions.

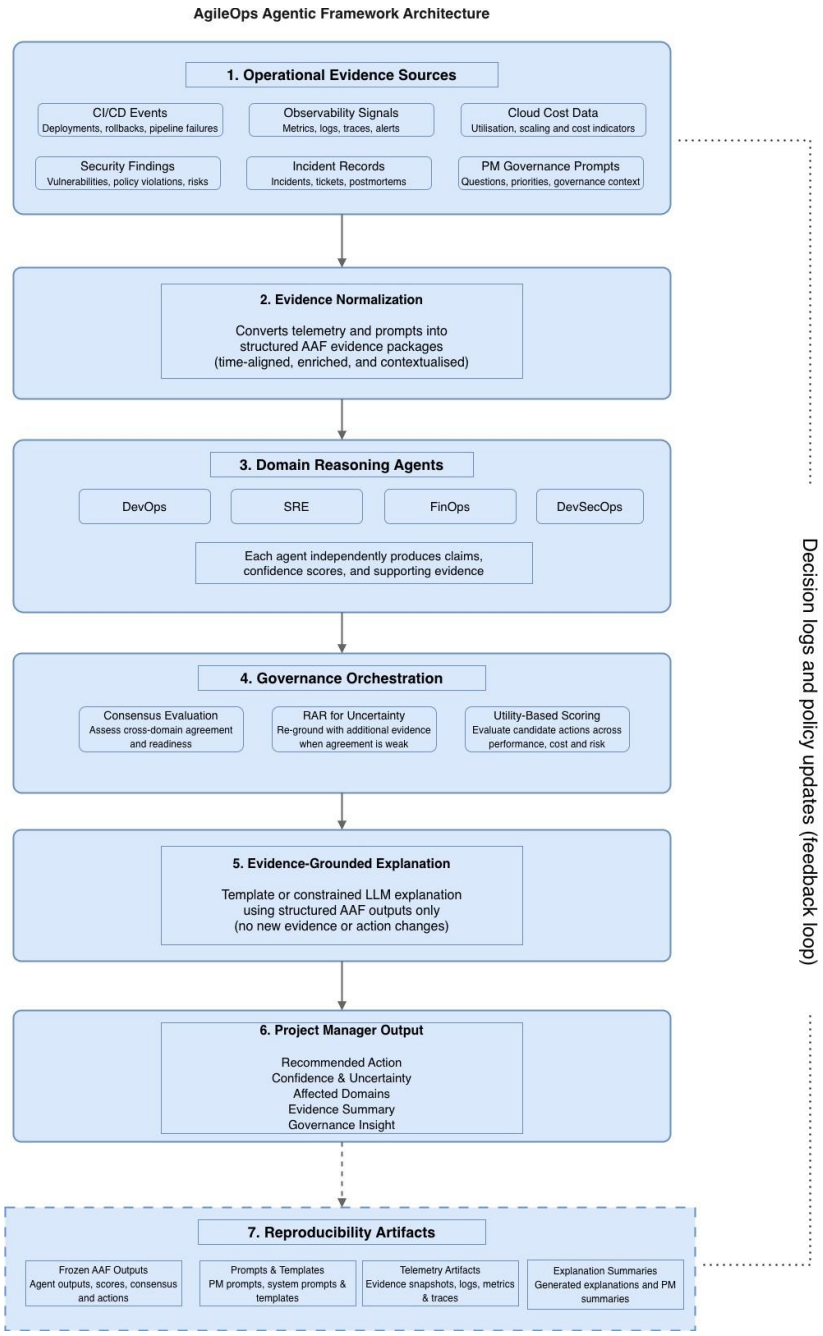


Figure 1. High-level architecture of the AgileOps Agentic Framework (AAF). The framework integrates four domain-specific reasoning agents (DevOps, SRE, FinOps, DevSecOps) with consensus evaluation, Re-Grounded Agentic Reasoning (RAR), utility-based governance scoring, and an evidence-grounded explanation layer.

AgileOps Agentic Framework

Framework Overview

This section presents the AgileOps Agentic Framework (AAF), a governance-oriented decision-support framework designed for Agile–DevOps environments. AAF is not intended to perform autonomous remediation or directly orchestrate infrastructure changes. Instead, the framework is designed to support governance and operational decision-making by providing Project Managers with a structured understanding of operational conditions across multiple delivery domains. The framework combines operational reasoning, uncertainty handling, utility-based governance evaluation, and evidence-based explainability within a unified workflow.

AAF continuously monitors telemetry generated from deployment pipelines, production observability platforms, cloud and virtualised infrastructure services, and security-monitoring systems. Operational telemetry is collected and transformed into structured evidence packages that are subsequently analyzed by domain-specific reasoning agents. Each reasoning agent independently evaluates the evidence and generates operational claims, associated confidence estimates, and supporting evidence summaries.

After domain agents generate their outputs, the framework evaluates the level of cross-domain agreement through a consensus-evaluation process. When sufficient agreement is not achieved, the framework activates the Re-Grounded Agentic Reasoning (RAR) process, where additional operational evidence is collected and re-evaluated in an attempt to improve alignment and reduce uncertainty.

Subsequently, governance decisions are evaluated using a utility-based scoring model that considers the impact of performance, cost efficiency, and risk reduction. The framework then generates evidence-based governance explanations intended for Project Managers and operational stakeholders. These explanations remain fully traceable to the underlying operational evidence.

The framework operates through five major stages:

- operational evidence ingestion,
- domain-agent interpretation,
- consensus and re-grounding evaluation,
- utility-based governance scoring,
- grounded explanation generation.

Unlike many operational-analysis platforms that optimize a single technical objective, AAF is designed to support governance interpretation across multiple operational domains simultaneously. This enables the framework to evaluate operational trade-offs across DevOps, SRE, FinOps, and DevSecOps perspectives within a unified governance-oriented workflow.

Operational Evidence Representation

Unlike many operational analytics systems that primarily expose logs or dashboards, AAF represents telemetry as structured operational evidence that supports or contradicts governance recommendations. This design improves reproducibility and allows governance decisions to be traced directly to the operational data used during the evaluation.

The operational evidence package is represented as:

$$E = \{e_1, e_2, \ldots, e_n\} \tag{1}$$

The framework uses structured operational signals collected from multiple sources, including deployment systems, observability platforms, cloud infrastructure, and software-defined networking environments. Representative evidence types include:

- deployment rollback events,
- service restart counts,
- latency and availability measurements,
- infrastructure utilization metrics,
- vulnerability findings,
- policy violations,
- cloud-cost indicators.

The framework does not require a fixed observability or deployment stack prior to evaluation. In controlled studies, evidence packages were dynamically generated while representative governance scenarios were evaluated. During the telemetry pilot, the Sock Shop microservices benchmark application[7] was used to generate and analyze operational artifacts in runtime.

Domain-Specific Reasoning Agents

AAF incorporates four deterministic reasoning agents representing DevOps, Site Reliability Engineering (SRE), FinOps, and DevSecOps. For each operational evidence package, an agent generates an operational interpretation represented as

$$A_i = (c_i, s_i, \varepsilon_i) \tag{2}$$

where A_i denotes the agent's output i, c_i represents the operational claim, s_i denotes the confidence score and ε_i represents the set of supporting evidence.

The SRE agent evaluates reliability, latency, service availability, restart behavior, and operational stability. The DevOps agent evaluates the stability of deployment, CI/CD failures, rollback behavior, and release conditions. The FinOps agent evaluates operational cost indicators and infrastructure-utilization efficiency. The DevSecOps agent evaluates vulnerabilities, policy violations, and operational security risks.

Each agent produces deterministic outputs derived from structured operational evidence rather than unrestricted language-model reasoning. This design improves reproducibility and ensures that operational claims remain grounded in evidence. A representative agent output is shown below:

```
{
  "agent_type": "SRE",
  "claim": "Service reliability degradation detected",
  "confidence": 0.85,
  "evidence": [
    "P95 latency elevated: 650 ms",
    "Availability dropped to 98.40%"
  ]
}
```

Consensus Evaluation

After the agent-level interpretation is completed, the governance orchestration layer computes a consensus score between agents that estimates the degree of agreement among participating agents. The framework does not require complete agreement across all domains, but instead aims to achieve sufficient consistency to support governance recommendations.

The consensus score is defined as:

$$C = \lambda \cdot \bar{s} + (1 - \lambda) \cdot D \tag{3}$$

where C denotes the consensus score, $\bar{s}$ represents the mean confidence of the agent, D denotes the agreement of the domain-alignment, and λ is a weighting parameter that balances confidence and agreement. For the experiments presented in this article, λ was fixed at 0.5 to balance the mean confidence and the agreement between domains equally.

A fixed threshold τ is then used to determine whether additional reasoning steps are required. If the consensus score falls below the threshold:

$$C < \tau \tag{4}$$

the framework activates the Re-Grounded Agentic Reasoning process. For the experiments reported in this article, the consensus threshold was fixed at $\tau = 0.65$.

The consensus mechanism is intended to estimate governance readiness rather than probabilistic truth inference. Individual agents contribute structured preferences based on operational evidence, and the orchestration layer evaluates the degree of agreement between domains before governance recommendations are finalized.

Re-Grounded Agentic Reasoning

Re-Grounded Agentic Reasoning (RAR) is designed to manage uncertainty in situations where agent agreement is insufficient or operational evidence is ambiguous. RAR attempts to improve alignment by incorporating additional operational context into the evidence package.

RAR is triggered when $C < \tau$, where C is the consensus score and τ is the minimum consensus threshold. During the RAR process, the framework enriches the evidence package with additional operational context and re-evaluates the agent outputs. The updated interpretation is accepted only if the consensus improvement exceeds a minimum increment:

$$C' - C \geq \delta_{\min} \tag{5}$$

where C' represents the updated consensus score, C represents the original consensus score, and $\delta_{\min}$ denotes the minimum acceptable improvement. If the consensus improvement remains insufficient, the original interpretation is retained, and the case

is classified as an unresolved uncertainty. Consequently, RAR functions primarily as an uncertainty-management mechanism, rather than as a performance-optimization mechanism. In the experiments presented in this article, $\delta_{\min} = 0.05$. To preserve deterministic behavior, RAR iterations were limited to a maximum of two re-grounding loops per evaluation case.

Utility-Based Governance Evaluation

After the consensus evaluation is completed, AAF applies a utility-based model to evaluate candidate governance actions according to performance, cost efficiency, and risk reduction.

The utility of governance action a_i is defined as:

$$U(a_i) = w_p P_i + w_c C_i + w_r R_i \tag{6}$$

where $U(a_i)$ is the utility score of action a_i, P_i represents the performance score, C_i represents the cost-efficiency score, R_i represents the risk-reduction score, and w_p, w_c and w_r are weighting coefficients. During the evaluation, fixed weights were used: $w_p = 0.4, w_c = 0.3, w_r = 0.3$.

Performance was assigned slightly higher weight because operational stability and release readiness are central governance concerns in Agile–DevOps environments. Cost efficiency and risk reduction were weighted equally in order to maintain the balance of governance between domains.

The selected governance action is determined as:

$$a^* = \underset{a_i}{\operatorname{argmax}}\, U(a_i) \tag{7}$$

Representative governance actions include rollback to stable deployment; mitigate and monitor; scale adjustment; patch or block release; and no action (observe). The utility layer evaluates governance trade-offs across operational conditions where multiple technically valid actions may exist. Different governance actions may improve reliability while increasing operational cost, or reduce security risk while affecting deployment timelines. The utility model provides a structured mechanism for balancing these competing objectives.

Evidence-Grounded Explanation Layer

In the final stage of AAF, the structured operational analysis is transformed into governance-oriented explanations intended for Project Managers and other stakeholders. The explanation layer uses a constrained language-model prompt that operates only on structured AAF output.

The explanation layer receives: domain claims; supporting evidence; confidence estimates; consensus score; Status of the RAR; utility evaluation; and selected governance action. The language model is not permitted to introduce new operational evidence or modify the selected governance action. This constraint reduces unsupported operational claims while preserving readability and interpretability.

The generated explanation summarizes what happened; why it happened; which operational domains were affected; what governance action is recommended; confidence and uncertainty status; and supporting evidence. Appendix A contains representative prompt templates and explanation examples.

Explainability Index

The quality of the explainability within AAF is evaluated using the Explainability Index (XI), a composite metric that estimates readability, clarity of evidence, and traceability. The XI score is defined as

$$XI = \alpha R + \beta E + \gamma T \tag{8}$$

where R denotes readability, E denotes evidence clarity, T denotes traceability, and α, β, and γ are weighting coefficients. During evaluation, equal weighting was used: $\alpha = \beta = \gamma = \frac{1}{3}$. All sub-scores were computed automatically using deterministic heuristic scoring rules and normalised to the range $[0, 1]$.

Readability estimates structural clarity and simplicity of sentences. The clarity of evidence estimates the degree to which the explanations are supported by concrete operational evidence. Traceability estimates whether governance conclusions can be traced back to supporting evidence and operational actions.

XI is intended as a lightweight heuristic for the estimation of explainability rather than as a human-subject evaluation metric. The metric does not evaluate semantic correctness directly; instead, it estimates how effectively operational evidence and governance reasoning are communicated to governance stakeholders.

End-to-End Workflow

The general AAF workflow proceeds as follows.

- operational telemetry is transformed into structured evidence,
- domain agents independently interpret the evidence,
- consensus evaluation estimates cross-domain agreement,
- RAR is triggered when agreement is insufficient,
- utility scoring evaluates candidate governance actions,
- the highest-utility governance action is selected,
- the explanation layer generates a grounded governance summary.

This workflow transforms fragmented operational telemetry into explainable and traceable governance recommendations intended to support Project Managers and governance stakeholders.

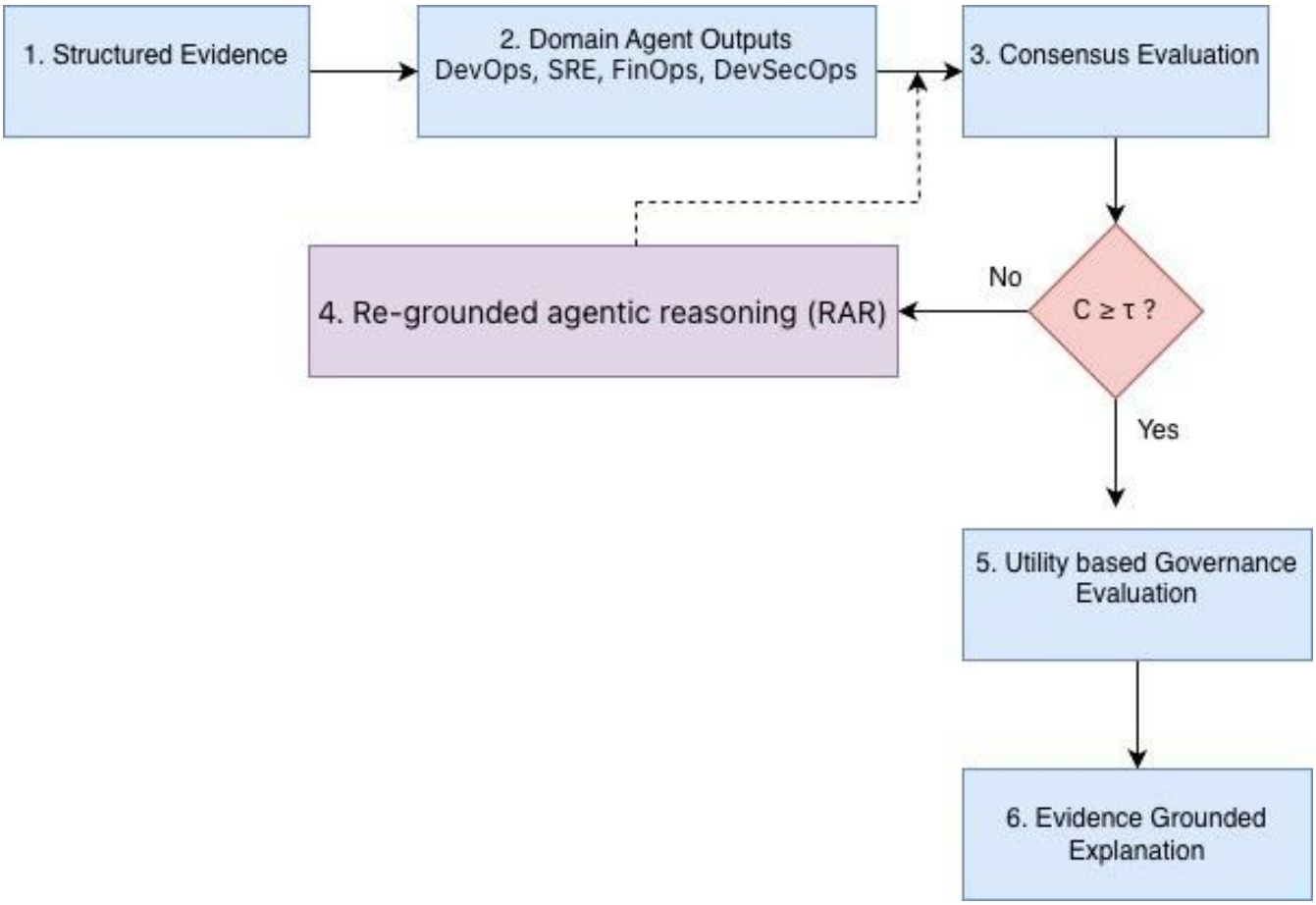


Figure 2. Governance decision workflow and utility-based evaluation process. The diagram illustrates how operational evidence flows through domain-agent interpretation, consensus evaluation, Re-Grounded Agentic Reasoning, utility scoring, and grounded explanation generation.

Experimental Methodology and Evaluation Design

Evaluation Objectives

This evaluation assesses the extent to which the AgileOps Agentic Framework (AAF) supports governance-oriented operational interpretation in Agile–DevOps environments across multiple delivery domains while maintaining explainability and traceability.

The evaluation focused on four primary objectives:

- assessing cross-domain operational interpretation,
- examining governance-action selection under uncertainty,
- evaluating the clarity and reliability of Project Manager-oriented explanations,
- Assess runtime operational artifacts as governance evidence.

The goal of the evaluation was not to assess infrastructure automation, autonomous remediation, or readiness for production deployment. Instead, the evaluation was designed as a controlled and reproducible study of governance-oriented operational reasoning and explainability.

Evaluation Design

The AAF evaluation followed a three-stage experimental protocol consisting of: controlled cross-domain governance scenarios; Project Manager-oriented prompt evaluation and grounded explanation assessment; and a controlled runtime telemetry pilot using the Sock Shop benchmark environment. Table 1 summarizes the evaluation structure and the corresponding outputs.

Table 1. Evaluation design summary.

Study	Objective	Input Type	Output Evaluated
Study 1	Cross-domain governance evaluation	Structured operational evidence	Governance interpretation and action selection
Study 2	PM prompt and explanation evaluation	Governance-oriented prompts	Grounded explanations and governance consistency
Study 3	Runtime telemetry pilot	Runtime operational telemetry	Runtime governance interpretation feasibility

The governance scenarios were designed to evaluate both operational interpretation and governance-action selection under representative operational conditions. Each scenario represented a sequence of operational events corresponding to realistic software-delivery situations. The PM prompt and explanation evaluation assessed the quality of grounded governance explanations generated for Project Managers, focusing on consistency, evidence traceability, and governance interpretability. The runtime telemetry pilot evaluated the feasibility of transforming runtime telemetry artifacts into structured governance evidence suitable for operational interpretation.

Study 1: Controlled Governance Scenarios

The first evaluation stage used controlled governance scenarios to evaluate the framework's operational interpretation capabilities across DevOps, SRE, FinOps, and DevSecOps domains. The evaluation scenarios included:

- deployment rollback events,
- elevated latency conditions,
- restart-loop behavior,
- infrastructure scaling anomalies,
- cloud-cost spikes,
- vulnerability findings,
- policy violations,
- release-readiness conflicts,
- cross-domain governance conditions involving multiple operational domains simultaneously.

Operational evidence was transformed into structured AAF evidence packages prior to evaluation. Each domain agent generated operational claims, confidence values, and supporting evidence summaries. The generated outputs were then processed through consensus evaluation, Re-Grounded Agentic Reasoning (RAR), utility-based governance evaluation, and grounded explanation generation. The evaluation data set consisted of 120 governance scenarios representing both isolated operational conditions and cross-domain governance situations.

Two operational-evaluation modes were used for the PM-oriented governance assessment.

Blind Evidence Mode. In this mode, operational evidence was derived primarily from prompt interpretation and runtime logic without predefined evidence alignment. The framework was therefore evaluated under incomplete or partially ambiguous operational conditions.

Aligned Evidence Mode. In this mode, curated governance labels and structured evidence packages were used to test the internal consistency of the framework under controlled evidence conditions.

Study 2: Project Manager Prompt and Grounded Explanation Evaluation

The second stage evaluated the explainability, traceability, and evidence-based characteristics of governance explanations generated for Project Managers. Representative governance-oriented prompt categories included release reviews; reliability investigations; deployment-governance discussions; cloud-cost assessments; and security and compliance reviews. Representative prompts included:

- "Should the release proceed despite elevated latency?"
- "Why was rollback recommended?"
- "Which operational domains contributed to the governance decision?"
- "Security patches are released to fix vulnerabilities before they can be exploited. If deployment is delayed too long, operational risk may increase."

The explanation layer did not use unrestricted natural-language prompting. Instead, explanations were generated using structured outputs from the AAF operational analysis pipeline, including domain claims, supporting evidence, confidence values, consensus scores, RAR status, utility-evaluation results, and selected governance actions. The language model was constrained so that it could not introduce unsupported operational evidence or alter governance decisions.

The generation of explanations was evaluated using a stratified sample of governance cases from the aligned PM-prompt data set. All explanations were generated using fixed-temperature settings with the GPT-4o-mini model. This stage evaluated domain consistency; governance-action consistency; evidence coverage; unsupported-claim risk; readability; evidence clarity; governance traceability; and heuristic composite explainability quality.

Study 3: Runtime Telemetry Pilot Using Sock Shop

The third evaluation stage introduced runtime operational realism using the Sock Shop microservices benchmark application[7]. Sock Shop is a cloud-native microservices benchmark environment commonly used for observability and operational experimentation. Unlike the earlier controlled-scenario studies, this stage relied on runtime telemetry collected while the system was actively operating under controlled perturbations.

A Docker Compose-based environment was used to simulate runtime operational conditions. Controlled runtime perturbations included:

- service stop pages,
- restart-loop behavior,
- elevated CPU utilization,
- request-pressure conditions,
- autoscaling anomalies,
- runtime configuration inspection,
- release-governance conflicts,
- security and release-gate conditions,
- multi-domain operational trade-offs.

Representative runtime artifacts included: Docker Compose status outputs; container logs; runtime inspection metadata; resource utilization metrics; HTTP service responses; deployment-state observations; and runtime configuration artifacts. The runtime telemetry pilot evaluated 20 controlled runtime cases. A total of 733 runtime operational artifacts were collected and transformed into structured operational evidence packages. The telemetry pilot was intended as a controlled feasibility study rather than a production operational deployment study.

Evaluation Metrics

The framework was evaluated using governance-oriented interpretation and explainability metrics rather than infrastructure-performance metrics. The reported confidence intervals represent descriptive uncertainty for the controlled datasets evaluated in this study and should not be interpreted as production-scale inferential estimates.

The domain match rate evaluates the accuracy of operational-domain identification:

$$DA = \frac{N_{\text{correct}}}{N_{\text{total}}} \tag{9}$$

where N_{correct} denotes the number of operational domains correctly identified and N_{total} denotes the total number of evaluated cases. The confidence intervals were estimated using the descriptive normal approximation.

Governance Action Match evaluates whether the selected governance action matches the governance recommendation defined during data set preparation.

$$AM = \frac{A_{\text{matched}}}{A_{\text{total}}} \tag{10}$$

where A_{matched} denotes the number of matched governance actions and A_{total} denotes the total number of governance actions evaluated. Two variants of governance-alignment were evaluated: (1) *Primary Governance Action Match*, which evaluates the exact governance action agreement; and (2) *Acceptable Governance Action Match (AGA Match)*, which allows equivalent governance actions in operationally ambiguous cross-domain situations.

The consensus score follows the formulation introduced earlier: $C = \lambda \cdot \bar{s} + (1 - \lambda) \cdot D$. Higher consensus values indicate stronger agreement between agent confidence levels and operational interpretation between domains.

The explainability index (XI) is evaluated as described in the Framework section. For grounded explanation evaluation, an additional Composite Explanation Score (CES) was computed using the same dimensions directly on the generated explanations. Both XI and CES were automatically calculated using deterministic scoring rules and should be interpreted as heuristic explainability measures rather than human-subject evaluation metrics.

Unsupported Claim Risk. A lightweight unsupported Claim Risk metric was used to evaluate whether generated explanations introduced unsupported statements not grounded in the operational evidence package. This metric was intended primarily as a heuristic estimate of hallucination risk rather than as semantic factuality.

Baseline and Ablation Configurations

To evaluate the contribution of individual components of the framework, controlled baseline and ablation configurations were implemented. The evaluated configurations included:

- full AAF system,
- AAF without RAR,
- AAF without utility scoring,
- single-agent baseline,
- prompt-only explanation baseline.

The ablation study isolated the contribution of multi-agent operational reasoning, uncertainty handling through RAR, utility-based governance evaluation, and evidence-based explanation generation.

Reproducibility and Implementation Setup

The evaluation pipeline was implemented using Python-based orchestration workflows that generated structured JSON operational evidence artifacts for each evaluation stage. Representative reproducibility artifacts included: structured evidence packages; governance summaries; Project Manager prompt libraries; runtime telemetry snapshots; Docker Compose runtime artifacts; evaluation outputs; configuration records; and explanation-generation prompts. The evaluation used controlled operational and runtime data only. Live production data was not incorporated into the experiments.

All experiments were carried out using deterministic settings, including fixed governance scenarios, fixed prompt sets, fixed consensus thresholds, and deterministic evaluation procedures. All framework parameters remained fixed throughout the evaluation: $\lambda = 0.5, \tau = 0.65, \delta_{\min} = 0.05, w_p = 0.4, w_c = 0.3, w_r = 0.3$. The RAR iterations were limited to a maximum of two loops per evaluation case. All utility weights, consensus thresholds, and RAR trigger parameters were predefined and kept constant in all experiments.

Results

Study 1: Controlled Governance Scenario Evaluation

Study 1 evaluated AAF using controlled governance-oriented operational scenarios spanning DevOps, SRE, FinOps, DevSecOps and cross-domain operational conditions. Table 2 presents the results of the controlled governance scenario and Table 3 presents the paired mean differences between the full AAF and the baseline or ablation configurations.

Table 2. Controlled governance scenario results. Domain match rate and governance action match are reported with 95% confidence intervals.

Method	Domain Match	95% CI	Action Match	95% CI
Traditional Baseline	0.633	[0.547, 0.720]	0.000	[0.000, 0.000]
Single-Agent Baseline	0.692	[0.609, 0.774]	0.000	[0.000, 0.000]
AAF Full	0.875	[0.816, 0.934]	0.792	[0.719, 0.864]
AAF without Consensus	0.808	[0.738, 0.879]	0.792	[0.719, 0.864]
AAF without RAR	0.808	[0.738, 0.879]	0.792	[0.719, 0.864]
AAF without Utility	0.875	[0.816, 0.934]	0.000	[0.000, 0.000]

Table 3. Paired mean differences between AAF Full and baseline or ablation configurations (descriptive uncertainty estimates).

Comparison	Mean Difference	95% CI
AAF Full domain match − Traditional Baseline	0.242	[0.138, 0.345]
AAF Full domain match − Single-Agent Baseline	0.183	[0.080, 0.287]
AAF Full domain match − AAF without Consensus	0.067	[0.016, 0.117]
AAF Full domain match − AAF without RAR	0.067	[0.016, 0.117]
AAF Full action match − AAF without Utility	0.792	[0.719, 0.865]

The complete AAF configuration achieved a domain-match rate of 0.875 and a governance-action match of 0.792 under the controlled governance evaluation. Compared with traditional and single-agent baselines, the framework improved operational-domain interpretation while also producing substantially stronger governance-action alignment.

The most important ablation result was observed in the AAF-without-Utility configuration. Although domain-match performance remained high, governance-action match dropped to 0.00. This suggests that identifying the operational domain alone is insufficient for governance support. The utility layer plays a central role in translating operational interpretation into governance-oriented action selection.

The consensus and RAR ablations produced comparatively smaller changes in aggregate performance. In the controlled evaluation, RAR primarily functioned as an uncertainty-identification mechanism rather than a major performance-improvement mechanism. Low-consensus operational cases were preserved as unresolved when additional evidence did not adequately improve alignment.

Framework Metrics and Utility Profile

Table 4 summarizes the operational metrics at the framework-level observed during the controlled governance evaluation, and Table 5 presents the components of the utility-profile.

Table 4. Framework-level metrics across ablation configurations.

Method	Consensus Mean	Utility Mean	XI Mean
AAF Full	0.74	0.81	0.81
AAF without Consensus	1.00	0.81	0.82
AAF without RAR	0.68	0.81	0.82
AAF without Utility	0.74	0.00	0.81

The complete AAF configuration produced balanced results across consensus behavior, governance utility, and explainability quality. The lower consensus score observed in the AAF-without-RAR configuration suggests that re-grounding contributes

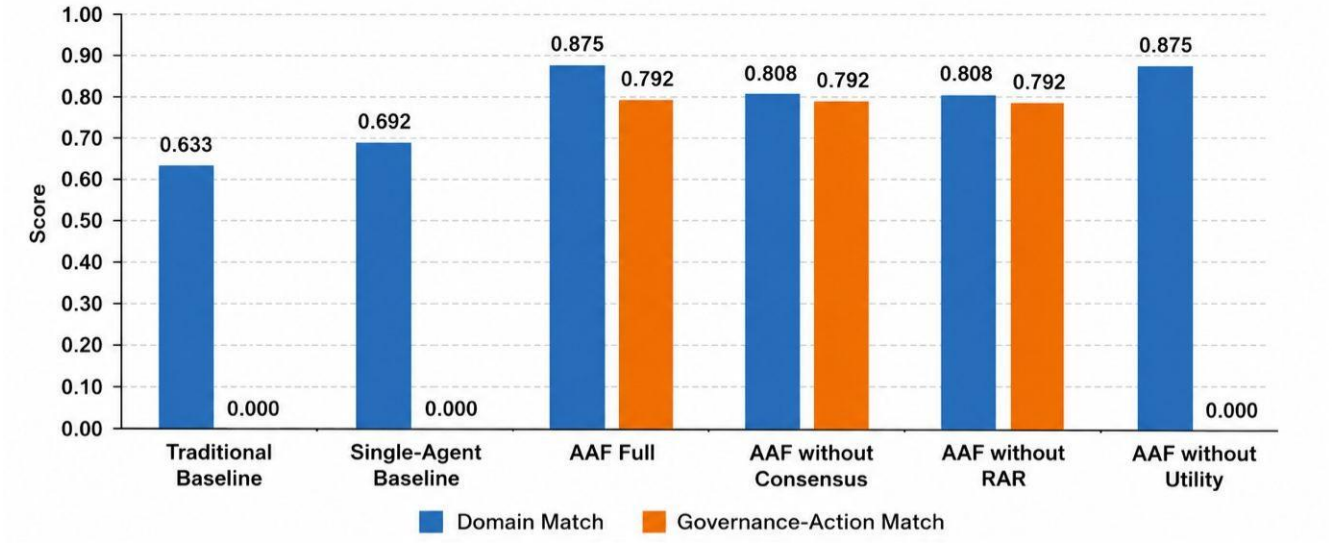


Figure 3. Controlled governance evaluation results. The utility layer is necessary for governance-action selection, while the multi-agent architecture improves operational-domain interpretation relative to baseline approaches.

to operational alignment when evidence across domains is uncertain or incomplete. However, the utility layer remained the dominant contributor to the selection of governance-action. Without utility evaluation, the framework could still interpret operational conditions, but could not reliably select governance actions aligned with the curated evaluation criteria. The utility profile suggests that AAF balances operational performance, cost efficiency, and risk reduction rather than optimizing only a single operational objective.

Table 5. Utility component scores during controlled governance evaluation.

Component	Mean Score
Performance	0.49
Cost Efficiency	0.68
Risk Reduction	0.52
Overall Utility	0.81

Study 2: Project Manager Prompt Evaluation

Study 2 evaluated whether AAF could process Project Manager-oriented governance prompts and generate operational-domain interpretations and governance actions under both blind and aligned evidence conditions. This evaluation includes 100 governance-focused questions that focus on release governance, reliability operations governance, cost governance, security governance, prioritization, escalation policies, and governance in cross-cutting scenarios.

Table 6 presents results under blind evidence conditions.

Blind evaluations were conducted successfully, reaching a 0.78 domain-match rate. The primary governance-action match rate was 0.37; however, the acceptable governance-equivalent action match rate reached 0.79. This is a strong result given the number of Project Manager prompts written with an incomplete, cross-domain, or ambiguous operational context. RAR was activated in 66% of the cases, highlighting the difficulties of cross-domain governance when faced with ambiguous circumstances and the lack of sufficient evidence.

Table 7 presents results under aligned evidence conditions.

Under aligned evidence conditions, the framework achieved full domain-match consistency and full acceptable governance-action consistency. The primary governance action match remained lower at 0.54, again reflecting the operational ambiguity associated with selecting a single correct governance action under cross-domain governance conditions. The aligned evaluation should therefore be interpreted primarily as a controlled consistency analysis rather than as evidence of unrestricted operational reasoning performance.

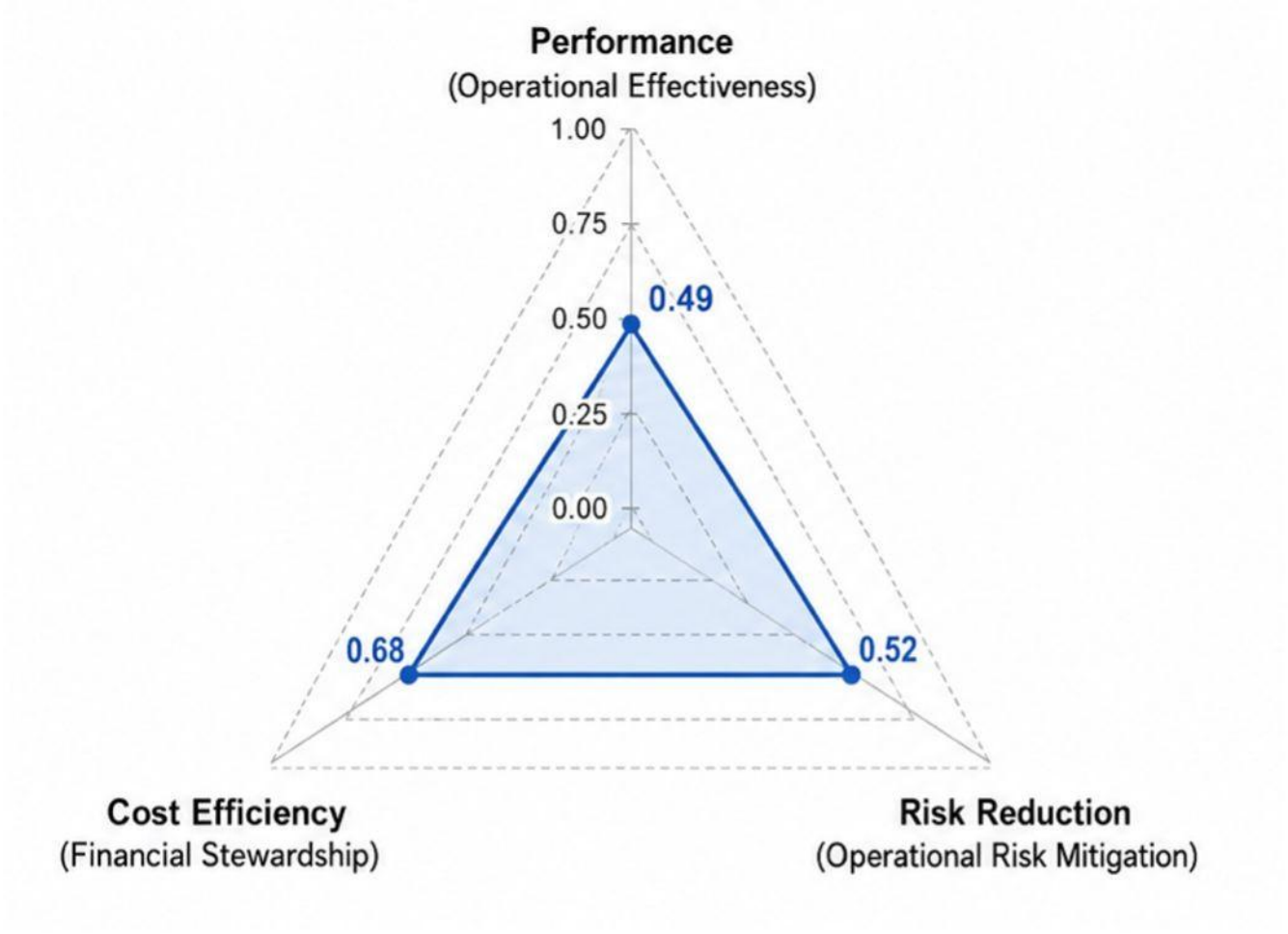


Figure 4. Utility profile across governance dimensions. AAF balances performance, cost efficiency, and operational risk reduction during governance-action selection.

Table 6. Project Manager prompt evaluation results—blind evidence mode.

Metric	Value
Number of Prompts	100
Domain Match Rate	0.78
Primary Governance Action Match	0.37
Acceptable Governance Action Match	0.79
Consensus Mean	0.701
Utility Mean	0.720
XI Mean	0.829
RAR Trigger Rate	0.66

Table 7. Project Manager prompt evaluation results—aligned evidence mode.

Metric	Value
Number of Prompts	100
Domain Match Rate	1.00
Primary Governance Action Match	0.54
Acceptable Governance Action Match	1.00
Consensus Mean	0.773
Utility Mean	0.973
XI Mean	0.816
RAR Trigger Rate	0.21

The contrast between blind and aligned modes is important because it demonstrates that governance interpretation quality is highly dependent on evidence completeness and operational grounding.

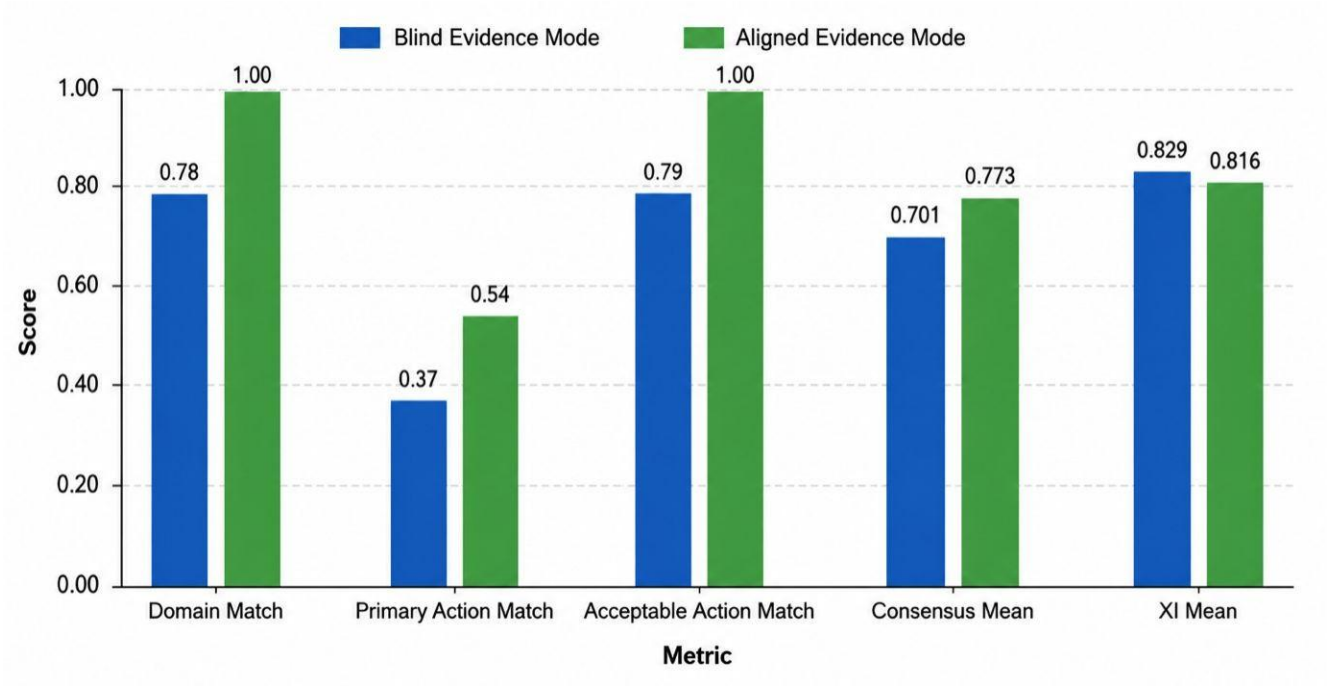


Figure 5. Project Manager prompt evaluation under blind and aligned evidence conditions. Domain match rate and acceptable governance-action match are compared across both evaluation modes.

Grounded LLM Explanation Evaluation

The grounded explanation evaluation examined whether the explanation layer could generate readable and traceable governance summaries while remaining constrained to structured AAF outputs. The evaluation used a stratified subset of 30 governance cases derived from the aligned PM-prompt dataset. The results are presented in Table 8.

Table 8. Grounded explanation evaluation results.

Metric	Value
Number of Cases	30
Domain Consistency	1.00
Action Consistency	1.00
Evidence Coverage	1.00
Unsupported-Claim Risk	0.281
CES Heuristic Mean	0.573
Readability Mean	0.676
Evidence Clarity Mean	0.477
Traceability Mean	0.561

These metrics should be interpreted as heuristic explainability indicators rather than formal human-subject interpretability measures. The grounded explanation layer preserved both the predicted operational domain and the governance action selected across all evaluated cases. The coverage of the evidence also remained fully aligned with the structured operational evidence package provided to the explanation layer. This suggests that constraining the language model to structured operational evidence can preserve governance consistency while still producing readable Project Manager-facing summaries.

The unsupported-claim risk metric should not be interpreted as a direct hallucination rate. Instead, it functions as a lightweight reproducible proxy that estimates whether explanations introduced vocabulary or operational assertions not grounded in the structured evidence package. The CES heuristic score should be interpreted similarly as a composite explainability estimate rather than a validated human-subject interpretability metric.

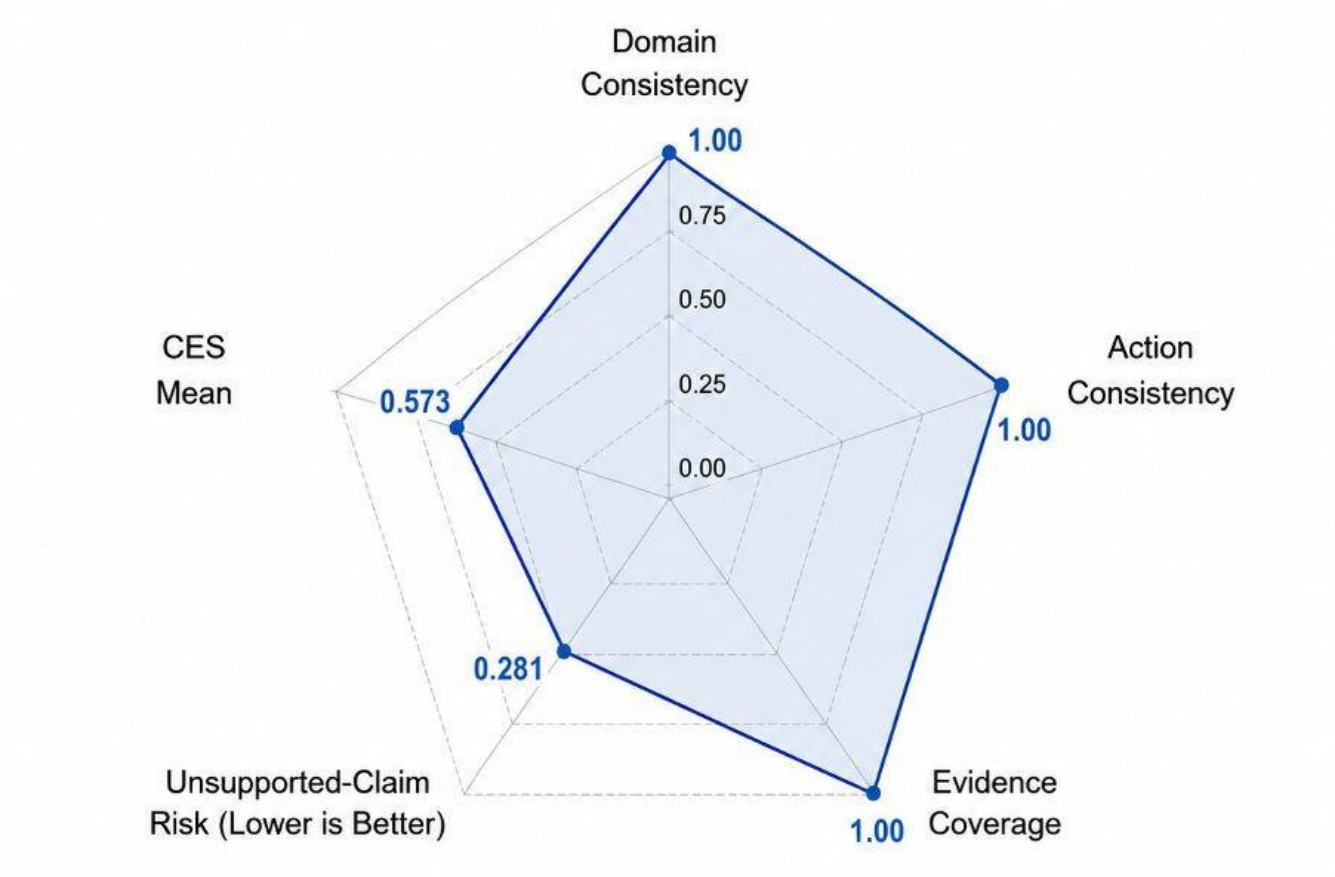


Figure 6. Grounded explanation evaluation across explainability metrics, including domain consistency, action consistency, evidence coverage, CES heuristic mean, readability, evidence clarity, and traceability.

Study 3: Runtime Telemetry Pilot Using Sock Shop

Study 3 evaluated AAF using runtime operational artifacts collected from the Sock Shop microservices benchmark application[7]. Unlike the earlier controlled evaluations, this study used operational artifacts derived from deliberate runtime perturbations executed within a Docker Compose deployment environment. The pilot included 20 controlled runtime operational cases involving service stoppages, restart-loop behavior, elevated CPU utilization, request-pressure conditions, runtime configuration inspection, release-governance conflicts, security-release gate conditions, and multi-domain operational trade-offs. A total of 733 runtime artifacts were collected in the 20 runtime cases. Table 9 presents the results.

Table 9. Runtime telemetry pilot results using Sock Shop operational artifacts.

Metric	Value
Benchmark	Sock Shop
Runtime Cases	20
Runtime Artifacts	733
Domain Match Rate	0.65
Governance Action Match	0.80
Consensus Mean	0.829
Utility Mean	0.962
XI Mean	0.817
RAR Trigger Rate	0.05

The runtime telemetry pilot achieved a 0.65 operational-domain match and 0.80 governance-action match across the controlled runtime cases. The lower domain-match rate indicates that runtime-derived evidence is more difficult to classify than curated governance evidence. Error cases were concentrated in mixed or cross-domain perturbations where runtime artifacts contained overlapping reliability, deployment, and resource usage signals. For example, service stoppages and restart behavior may be interpreted as either DevOps release instability or SRE reliability degradation depending on the available evidence. Similarly, resource-pressure conditions may overlap with FinOps scaling concerns when runtime utilization signals are not accompanied by explicit cost indicators.

This result is important because it shows that runtime operational artifacts are noisier and less explicitly labeled than controlled scenario evidence. The framework retained a stronger governance-action match than the domain-match rate, suggesting that action selection can remain useful even when the dominant domain is ambiguous. Despite this reduction in operational-domain interpretation performance, governance-action consistency remained comparatively strong. RAR activation was limited during the runtime pilot, with only one runtime case triggering unresolved low-consensus behavior.

The telemetry pilot should not be interpreted as operational validation on a production-scale. Rather, it demonstrates that runtime operational artifacts collected from a live microservices environment can be transformed into structured governance evidence and processed consistently under controlled runtime conditions.

Results Summary

Throughout all three stages of the evaluation, consistent general patterns emerged. The multi-agent approach improves operational interpretation within a real-time governance simulation environment when compared with traditional and single-agent approaches. The results obtained from the traditional and single-agent baselines were consistently weaker in governance-oriented operational interpretation.

The utility ablation study showed that simply identifying operational behavior was not sufficient to provide governance guidance. Grounded explanation generation maintained consistency between operational domains and governance actions when constrained to structured AAF output.

It was harder to decide on governance actions than to identify operational domains when the commands lacked sufficient operational context. Performance also decreased under blind evidence conditions compared to aligned evidence conditions. The runtime telemetry pilot exposed additional complexity in interpreting operational domains as we moved from fully controlled governance scenarios to analyzing live runtime components and operational data within a distributed microservices environment.

Rather than forcing high-confidence decisions under incomplete evidence, AAF preserves uncertainty when operational conditions remain ambiguous.

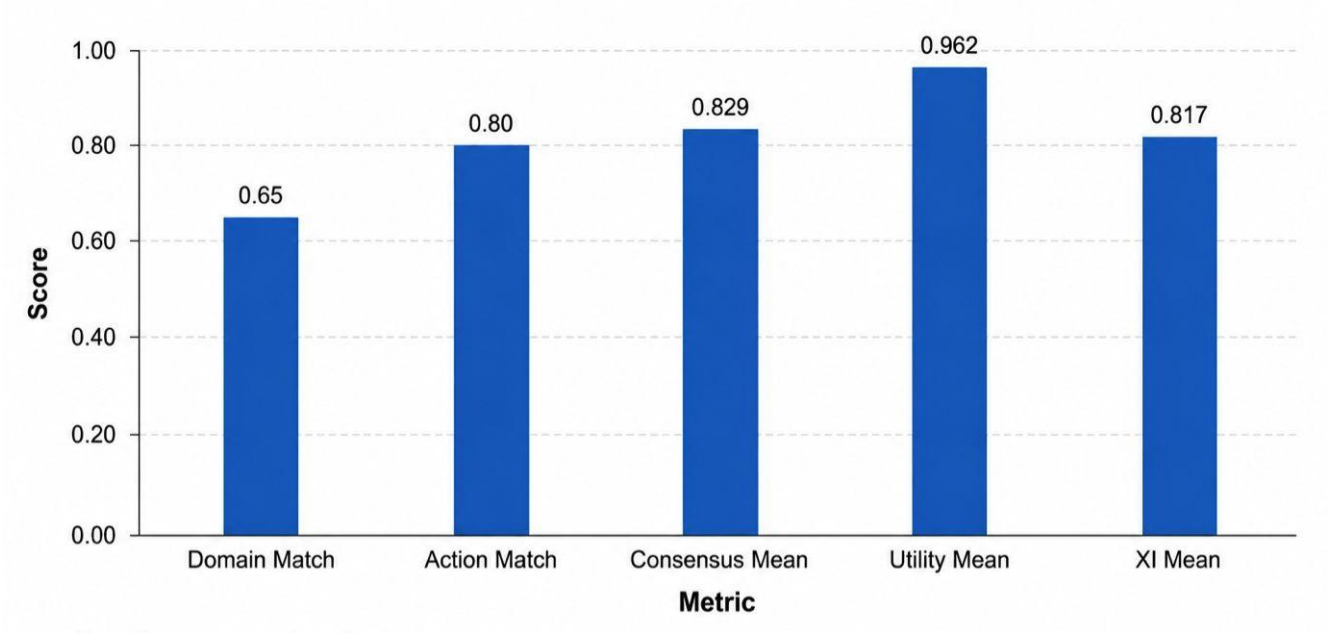


Figure 7. Runtime telemetry pilot results using Sock Shop operational artifacts. Domain match rate, governance action match, consensus mean, and utility mean are reported across the 20 controlled runtime cases.

Discussion

This paper shares experience with the use of the AgileOps Agentic Framework (AAF) to support governance-oriented interpretations of operational telemetry in Agile–DevOps settings. To validate the results produced by AAF, the framework was applied to a set of controlled operational scenarios, and the quality of the telemetry-to-governance mappings it produced was assessed. This paper investigates the extent to which a fully configured AAF can produce stronger operational-to-governance mappings while preserving interpretability and evidence grounding.

To validate our approach, we performed a series of ablation experiments in which we removed the utility-scoring layer from the governance evaluation flow. Removing the utility-scoring layer had only a modest effect on identifying operational domains, but a much greater effect on governance-action alignment. Selecting governance actions to satisfy operational goals such as reliability, delivery time, operational cost, and security risk is an inherent challenge because these objectives are often competing and interdependent[2,2]. The utility-scoring layer therefore acted as an organizational scorecard for evaluating governance trade-offs in a structured and reproducible manner.

During the Project Manager prompt evaluation, even under blind prompt-derived telemetry conditions, the framework achieved a 0.78 domain-match rate and a 0.79 acceptable governance-action match across more than 100 governance-oriented prompts targeting the Project Manager role. Under aligned evidence conditions, both the domain consistency score and the acceptable governance-action alignment reached 1.00. However, the primary action-match score remained lower because many prompts represented ambiguous operational conditions or situations that span multiple operational domains, where several governance responses were operationally appropriate. These findings suggest that evaluating governance utility is more effective than relying solely on operational classification.

The constrained explanation layer consistently preserved governance consistency and maintained coverage of relevant supporting evidence while reducing unsupported claims across explanatory cases. This architectural decision –using the language model for explanation generation while restricting governance decisions to deterministic framework components –improves governance accountability and traceability while reducing the risks associated with unrestricted language model reasoning.

In addition, we conducted a runtime telemetry pilot using running Sock Shop services in a controlled but perturbed environment. We collect operational telemetry including container status information, logs, restart behavior, CPU utilization, memory allocation, latency metrics, and service failure rates. The collected runtime telemetry was subsequently transformed into governance-oriented operational insights while preserving interpretability, evidence traceability, and stability of operational interpretation.

Implications for Project Governance

Governance in Agile DevOps environments is often complicated not only by limited operational visibility, but also by fragmented evidence distributed across specialized deployment pipelines, observability platforms, cloud-management systems, and security-monitoring tools. Project Managers currently review telemetry and organizational data manually across multiple disconnected tools and workflows. These workflows often lack an integrated engineering interpretation from DevOps, SRE, FinOps, and DevSecOps perspectives. AAF extends beyond conventional telemetry analysis by introducing a governance-oriented interpretation layer that combines reasoning across these operational domains to produce evidence-linked claims, confidence estimates, utility scores, uncertainty indicators and grounded governance summaries within a unified workflow[?,?]. Applying this governance-oriented operational interpretation to areas such as release evaluation, reliability coordination, cloud-cost governance, operational prioritization, and security-risk assessment can provide useful operational intelligence and improve managerial interpretability in Agile–DevOps governance environments.

Role of Re-Grounded Agentic Reasoning

Re-Grounded Agentic Reasoning (RAR) is designed primarily as an uncertainty-management mechanism rather than an optimization mechanism. RAR identifies low-consensus operational conditions and prevents weakly supported reasoning from producing poorly justified high-confidence governance recommendations[?]. After re-grounding, some low-consensus cases intentionally remained unresolved. This should not be interpreted as a framework failure. In governance-oriented operational environments, preserving uncertainty is often preferable to producing unsupported high-confidence governance recommendations.

RAR currently operates through controlled evidence enrichment. Future work should explore integration with live observability tools, deployment telemetry, cloud-cost monitoring platforms, and security-event pipelines to better understand how uncertainty resolution behaves under real-world operational conditions.

Grounded Explanation Generation and Evidence Consistency

AAF separates deterministic operational reasoning from language-model-based explanation generation. Governance interpretation, consensus evaluation, and utility scoring are completed before the language model is introduced into the workflow. The language model receives structured output from the framework rather than raw operational telemetry. These structured outputs include: predicted operational domain; selected governance action; supporting evidence summaries; consensus values; utility scores; and uncertainty indicators.

The explanation layer is constrained so that it cannot alter governance decisions or introduce unsupported organizational evidence. Its primary purpose is to improve the readability and interpretability for Project Managers and other governance stakeholders. This architectural separation is important because unsupported language-model outputs could negatively affect release governance, reliability coordination, security evaluation, and cloud-cost decisions. Our experiments showed that constrained explanation generation improved usability and interpretability while maintaining strong evidence consistency[?,?]. Future work should include semantic factual evaluation, human-centered usability studies, and governance studies that involve practicing project managers and operational stakeholders.

Limitations

This work has several limitations. First, the evaluation scenarios were conducted in controlled environments and therefore do not fully represent the organizational complexity and governance variability present in real-world operational settings. Similarly, the Project Manager prompt dataset does not capture all communication styles and governance processes used across organizations and software-delivery pipelines.

Second, the evaluation of grounded explanations was limited to a relatively small number of cases and a single language-model configuration. A broader evaluation involving additional models, datasets, and prompting strategies is required to strengthen the generalizability of the findings and further evaluate unsupported-claim reduction approaches.

Third, the runtime telemetry pilot was intentionally conducted in a controlled environment rather than at the production scale. The objective of the pilot was to evaluate the feasibility of integrating runtime telemetry into governance workflows rather than validating production deployment readiness.

Fourth, the framework has not yet been empirically validated through human-centered studies involving practicing Project Managers. Consequently, conclusions about usability, trust, workload reduction, and governance effectiveness should be interpreted cautiously until validated through real-world evaluation studies.

Finally, this work used fixed utility weights and governance thresholds throughout the evaluation. In practice, organizations operate with different priorities that include reliability, speed of delivery, operational cost, and security requirements. Future work should therefore investigate adaptive utility weighting, expert-labeled governance datasets, longer-duration runtime telemetry collection, and realistic evaluations involving operational Agile–DevOps governance workflows.

Conclusion

The AgileOps Agentic Framework (AAF), a governance-oriented multi-agent decision-support framework for Agile–DevOps operational environments. The framework integrates DevOps, Site Reliability Engineering, FinOps, and DevSecOps reasoning through structured operational evidence, consensus-driven evaluation, grounded agentic reasoning (RAR), utility-based governance scoring, and evidence-based explanation generation.We evaluated the framework through three complementary studies. The first study evaluated operational interpretation, governance-action selection, governance justification, and consensus behavior using controlled governance scenarios. The second study evaluated Project Manager-oriented governance prompts under both blind-evidence and aligned-evidence conditions. The third study evaluated runtime operational telemetry generated during controlled perturbations of the Sock Shop microservices benchmark environment.

This research demonstrated that combining multi-agent operational reasoning with utility-based governance evaluation improves governance-action alignment compared with simpler operational-classification approaches. The ablation analysis further showed that effective governance-action selection requires more than operational-domain identification alone; it also requires a structured evaluation of trade-offs involving reliability, performance, operational cost, and security risk.The Project Manager prompt evaluation demonstrated strong governance consistency under aligned evidence conditions while also revealing the challenges associated with ambiguous and cross-domain operational situations. These findings support a utility-oriented interpretation of governance rather than a purely classification-oriented approach.

Also evaluated grounded explanation generation under constrained evidence conditions. The results showed that the generation of evidence-based explanations can preserve domain consistency, governance-action consistency, and evidence coverage while reducing unsupported claims. These findings support the architectural separation between deterministic operational reasoning and the generation of language-model explanations.Beyond controlled synthetic governance scenarios, the runtime telemetry pilot evaluated 20 controlled runtime operational cases and processed 733 runtime telemetry artifacts collected from the Sock Shop benchmark environment. The results demonstrated that runtime operational telemetry can be transformed into governance-oriented operational interpretations while maintaining explainability, stability, and governance traceability.

Overall, by combining multi-agent operational reasoning, utility-based governance scoring, uncertainty handling, and grounded explanation generation can improve governance-oriented interpretation of operational telemetry in Agile–DevOps environments.Future work will focus on larger-scale runtime telemetry studies, integration with live observability and CI/CD platforms, adaptive governance weighting mechanisms, broader language-model evaluation, semantic factuality assessment, and human-centered studies involving practising Project Managers operating within realistic governance workflows.

Acknowledgements

The authors would like to thank AppTestify Global Services Pvt. Ltd., Pune, Maharashtra, India for providing the authors with access to cloud infrastructure and compute resources on which the experimental workloads were run to gather the data for this article.

Author contributions statement

S.K. conceived the framework, designed the evaluation methodology, implemented the AgileOps Agentic Framework, conducted the experiments, and drafted the manuscript. S.A. supervised the research, provided critical guidance on the evaluation design and explainability framework, and reviewed and edited the manuscript. C.J. contributed to the analysis of results, interpretation of findings, and review of the manuscript. All authors reviewed the final manuscript.

Additional information

Competing interests: The authors declare that they have no competing interests.

Data availability: Implementation code, evaluation scripts, runtime telemetry artifacts, Project Manager prompt datasets, grounded explanation evaluations, and reproducibility outputs are available at: `https://github.com/Ksuresh/AgileOps-Agentic-Framework`.

References

1. N. Forsgren, J. Humble, and G. Kim, Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. Portland, OR, USA: IT Revolution Press, 2018.
2. G. Kim, P. Debois, J. Willis, and J. Humble, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. Portland, OR, USA: IT Revolution Press, 2016.

3. J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston, MA, USA: Addison-Wesley Professional, 2010.
4. G. Kim, K. Behr, and G. Spafford, The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win, 5th Anniversary ed. Portland, OR, USA: IT Revolution Press, 2018.
5. B. Beyer, C. Jones, J. Petoff, and N. Murphy, Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA, USA: O'Reilly Media, 2016.
6. B. Beyer, N. Murphy, D. Rensin, K. Kawahara, and S. Thorne, The Site Reliability Workbook: Practical Ways to Implement SRE. Sebastopol, CA, USA: O'Reilly Media, 2018.
7. N. J. Gunther, Guerrilla Capacity Planning: A Tactical Approach to Planning for Highly Scalable Applications and Services. Heidelberg, Germany: Springer, 2007.
8. B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, "Dapper, a large-scale distributed systems tracing infrastructure," Google Research, Tech. Rep., 2010.
9. M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS), 2017, pp. 1285–1298.
10. P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in Proc. IEEE Int. Conf. Web Services (ICWS), 2017, pp. 33–40.
11. S. Dang, L. Wu, Z. Li, and H. Zhang, "AIOps: Real-world challenges and research innovations," in Proc. IEEE/ACM 42nd Int. Conf. Software Engineering: Companion Proceedings (ICSE-Companion), 2020, pp. 4–5.
12. FinOps Foundation, Cloud FinOps: Collaborative, Real-Time Cloud Financial Management. Portland, OR, USA: FinOps Foundation Press, 2021.
13. M. Myrbakken and R. Colomo-Palacios, "DevSecOps: A multivocal literature review," in Proc. IEEE/ACM Int. Conf. Software and System Processes (ICSSP), 2017, pp. 17–24.
14. R. Rahman and L. Williams, "Software security in DevOps: Synthesizing practitioners' perceptions and practices," in Proc. IEEE/ACM Int. Workshop on Continuous Software Evolution and Delivery (CSED), 2016, pp. 70–76.
15. National Institute of Standards and Technology (NIST), Application Container Security Guide, NIST Special Publication 800-190, 2017.
16. A. B. Arrieta et al., "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," Inf. Fusion, vol. 58, pp. 82–115, 2020.
17. A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence," IEEE Access, vol. 6, pp. 52138–52160, 2018.
18. D. Gunning and D. Aha, "DARPA's explainable artificial intelligence (XAI) program," AI Mag., vol. 40, no. 2, pp. 44–58, 2019.
19. L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen, "A survey on large language model-based autonomous agents," Frontiers of Computer Science, vol. 18, no. 6, pp. 1–26, 2024.
20. W. X. Zhao et al., "A survey of large language models," ACM Comput. Surveys, vol. 56, no. 9, pp. 1–38, 2024.
21. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 9459–9474.
22. Weaveworks, "Sock Shop: A microservices demo application," GitHub repository. [Online]. Available: https://github.com/microservices-demo/microservices-demo.
23. N. Dragoni, S. Giallorenzo, A. Lluch-Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: Yesterday, today, and tomorrow," in Present and Ulterior Software Engineering, M. Mazzara and B. Meyer, Eds. Cham, Switzerland: Springer, 2017, pp. 195–216.
24. A. Bogatinovski, J. Nedelkoski, J. Cardoso, and O. Kao, "Self-supervised anomaly detection from distributed traces," in Proc. Web Conf. (WWW), 2021, pp. 1795–1805.
25. T. Chen, X. Zhang, S. Kandula, and D. Katabi, "Root cause localization for failures in large-scale distributed systems," in Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI), 2020, pp. 337–350.

26. J. Barr, “Cloud computing economics,” ACM Queue, vol. 13, no. 9, pp. 20–31, 2015.

27. A. Capozucca, S. Pastore, and A. Tamburri, “Towards sustainable cloud cost governance,” Future Generation Computer Systems, vol. 124, pp. 1–14, 2021.

28. M. Wooldridge, An Introduction to Multi-Agent Systems, 2nd ed. Hoboken, NJ, USA: Wiley, 2009.

29. G. Weiss, ed., Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA, USA: MIT Press, 1999.

30. M. d’Inverno and M. Luck, Understanding Agent Systems, 2nd ed. Berlin Heidelberg, Germany: Springer, 2004.

31. S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Upper Saddle River, NJ, USA: Pearson, 2021.

32. J. S. Park, J. O’Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, “Generative agents: Interactive simulacra of human behavior,” in Proc. 36th Annu. ACM Symp. User Interface Software and Technology (UIST), 2023, pp. 1–22.

33. S. Ji, T. Yu, S. Xu, and E. Cambria, “Survey of hallucination in natural language generation,” ACM Comput. Surveys, vol. 55, no. 12, pp. 1–38, 2023.

34. E. Ries, The Lean Startup: How Today’s Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York, NY, USA: Crown Business, 2011.

35. W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, “Detecting large-scale system problems by mining console logs,” in Proc. ACM SIGOPS 22nd Symp. Operating Systems Principles (SOSP), 2009, pp. 117–132.

36. P. Leitner and J. Cito, “Patterns in the chaos—A study of performance variation and predictability in public IaaS clouds,” ACM Trans. Internet Technol., vol. 16, no. 3, pp. 1–23, 2016.

37. M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, and S. Gil, “Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud,” in Proc. IEEE 10th Colombian Computing Conf. (CCC), 2015, pp. 583–590.

Appendix A: Prompt Templates, Runtime Telemetry, and Reproducibility Details

A.1 LLM System Prompt

```
You are a Project Manager-facing AgileOps governance assistant.

Your task is to explain the governance decision using only the structured
AAF evidence provided.

Strict rules:
- Do not invent facts.
- Do not introduce telemetry, incidents, risks, causes, users, teams,
  systems, or impacts that are not present in the input.
- Preserve the selected action exactly as provided.
- Preserve the predicted primary domain exactly as provided.
- Use the same evidence phrases from the agent outputs wherever possible.
- If evidence is missing or incomplete, state that evidence is incomplete.
- Do not recommend a different action.
- Do not mention tools, dashboards, logs, or production systems unless
  they appear in the input.
- Keep the explanation concise and evidence-grounded.
```

A.2 LLM User Prompt Template

```
Generate a Project Manager-facing governance explanation from the
structured AAF output below.

Use this exact output format:

1. What happened
   Predicted primary domain: <copy predicted_primary_domain exactly>.
```

```
   Selected action: <copy selected_action exactly>.

2. Why it happened
   Summarize the cause using only agent claims and evidence from the input.

3. Cross-domain impact
   Mention only domains that appear in the agent outputs.
   Do not add new domains.

4. Recommended governance action
   Repeat the selected action exactly: <copy selected_action exactly>.
   Explain why it is appropriate using utility values from the input.

5. Confidence and uncertainty
   Mention the consensus score and RAR status from the input.

6. Evidence used
   List 3 to 6 evidence items. Copy evidence phrases from the agent
   outputs as closely as possible.

7. PM decision implication
   Give one concise sentence for the Project Manager. Do not add new facts.

Structured AAF output:
{payload_json}
```

A.3 Representative Structured AAF Output


```
{
  "scenario_id": "pm001",
  "predicted_primary_domain": "DevSecOps",
  "selected_action": "Patch or block release",
  "consensus_score": 0.86,
  "rar": {
    "triggered": false,
    "accepted": true,
    "before": 0.86,
    "after": 0.86,
    "loops": 0
  },
  "utility": {
    "selected_action": "Patch or block release",
    "best_utility": 1.0494,
    "performance_score": 0.4325,
    "cost_efficiency_score": 0.70,
    "risk_reduction_score": 0.888
  },
  "agents": [
    {
      "agent_type": "DevOps",
      "claim": "Deployment signal indicates a possible release or
               configuration issue",
      "confidence": 0.78,
      "evidence": [
        "Rollback marker present in release telemetry",
        "Moderate restart loops observed: 8"
      ]
    },
    {
      "agent_type": "SRE",
      "claim": "Reliability signal indicates a possible service health issue",
      "confidence": 0.78,
      "evidence": [
        "P95 latency elevated: 650 ms",
        "Availability dropped to 98.40%"
      ]
    },
    {
      "agent_type": "DevSecOps",
```

```
      "claim": "Security or compliance issue is the likely primary
               operational cause",
      "confidence": 0.92,
      "evidence": [
        "Multiple critical CVEs detected: 2",
        "Policy-as-code violation detected",
        "Compliance evidence gap detected"
      ]
    }
  ]
}
```

A.4 Runtime Telemetry Artifact Structure

The runtime telemetry pilot used controlled operational perturbations executed on the Sock Shop microservices benchmark application[7]. Runtime artifacts were collected from Docker Compose-based executions and transformed into structured AAF evidence packages. Representative runtime artifacts included: container status snapshots; service restart observations; runtime CPU and memory statistics; deployment and image inspection metadata; service health responses; operational logs; release-state observations; and configuration inspection artifacts.

The telemetry pilot included 20 runtime cases and 733 collected runtime artifacts across DevOps, SRE, FinOps, and DevSecOps operational conditions. Representative runtime case categories included:

- service termination scenarios,
- repeated restart conditions,
- runtime resource pressure,
- deployment instability,
- release-governance conflicts,
- cross-domain reliability and cost trade-offs,
- security release-gate conditions.

A.5 Governance Action Equivalence Mapping

Because governance-oriented prompts may allow multiple operationally defensible responses, the evaluation distinguished between strict primary-action matching and acceptable governance-equivalent actions. Table 10 presents representative equivalence mappings. This mapping was used only for governance-equivalent evaluation and not for domain interpretation scoring.

Table 10. Governance action equivalence examples (Table A-I).

Primary Governance Action	Acceptable Governance-Equivalent Actions
Patch or block release	Block release and fix pipeline
Mitigate and monitor	Review operational stability
Scale adjustment	Review scaling policy
Rollback to stable deployment	Controlled rollback
Observe and defer	No immediate operational action

A.6 Reproducibility Artifacts

The implementation code, evaluation scripts, runtime telemetry artifacts, PM prompt datasets, grounded explanation evaluations, and reproducibility outputs are available at: `https://github.com/Ksuresh/AgileOps-Agentic-Framework`.

The repository includes controlled governance scenarios; Project Manager prompt datasets; grounded explanation evaluation outputs; Sock Shop runtime telemetry artifacts; JSONL evaluation records; deterministic evaluation workflows; reproducibility configuration files; figure-generation scripts; and runtime telemetry snapshots. The repository preserves frozen evaluation outputs for all reported experiments, including: Phase 1 controlled governance evaluation; Phase 2A PM prompt evaluation; Phase 2B grounded explanation evaluation; and Phase 3 runtime telemetry pilot. API keys, credentials, and private infrastructure details are not included.

A.7 Threats to Validity

The evaluation was designed to prioritise reproducibility and governance-oriented operational interpretation rather than production-scale deployment validation.Internal validity may be affected by the use of curated governance scenarios and structured prompt datasets. External validity is limited because the runtime telemetry pilot used the Sock Shop benchmark application rather than enterprise-scale operational environments. Construct validity is limited by the use of heuristic explainability measures such as XI and CES, which estimate readability, evidence clarity, and traceability rather than semantic correctness directly. To improve reproducibility, all experiments used deterministic seeds, frozen evaluation datasets, structured telemetry artifacts, and preserved evaluation outputs.

A.8 Funding Declaration

No funding was received from any organization for this work.