**Journal of Information Systems Engineering and Management**
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

**Research Article**

# Autonomous Platform Engineering: A Framework for Eliminating Human-Induced Latency in Cloud Operations

Pradeep Kurra

Trace3, USA

**ARTICLE INFO**

Received: 10 Feb 2026

Revised: 15 Feb 2026

**ABSTRACT**

Many enterprise cloud operations today still rely on human remediation or ticketing, which can lead to increased time to resolution. This architectural effort is to move enterprise cloud operations from human-driven DevOps approaches to policy-driven, closed-loop autonomous operations. It seeks to eliminate talent shortages and queuing within hyperscale cloud growth by enabling self-healing autonomous infrastructure through agentic AI systems. These are goal-driven reasoning agents acting within a governance policy frame. Platform engineering is more than automating rules to remove manual tasks. Platform engineering enables intelligent agents to autonomously determine and execute scaling, security mitigation and compliance as a function of defined scenarios. Further, it separates autonomous versus human-in-the-loop configurations, and quantifies smart behavior. It establishes operational patterns with levels of performance that no human could achieve. By eliminating human-induced latency for routine operational activities and limiting human involvement to high-level decision making and non-routine situations, it enables operational velocity and operational reliability that are impossible with customary approaches. This article helps organizations prepare their environment to safely and effectively use autonomous systems against production workloads, including providing technical and governance frameworks.

**Keywords:** Agentic AI, Autonomous Operations, Closed-Loop Systems, Policy-Driven Intelligence, Self-Healing Infrastructure, Platform Engineering

## I. Introduction

Despite the large-scale deployment of automation tooling, enterprise cloud operations are still predominantly human-centric, with human operators responding to alerts from monitoring tools, reviewing incidents in ticketing systems, and executing remediations in change management workflows. Such human coordination leads to latency between incident detection and the remediation being executed, leading to an operational bottleneck and reduced agility and reliability in the system. Industry studies report that the time between generating the alert and notifying an operator ranges from 15 to 45 minutes for cloud-based on-demand processes for managing incident resolution tickets. An additional 2 to 6 hours are required for diagnosis and implementation of a remedy. As cloud environments grow in size, complexity and speed, human operators define the maximum throughput and minimum latency of cloud operations.

Human factors engineering research demonstrates that individuals' performance deteriorates when faced with high workloads that exceed their cognitive capabilities. The risk of this decreased performance is especially relevant to cloud engineering practices because of the shortage of cloud engineers and cloud operations engineers who are skilled in operating complex distributed systems. It has been shown that performance degrades when people have to work on multiple problems, especially when sustained attention or working memory is involved. Furthermore, even well-staffed organizations often lack adequate preparation for an influx of incidents or those affecting multiple domains within the system. This phenomenon is due to the limitation of human attention, which cannot be increased linearly with the growth of demand for cloud operations [2]. A proposed framework for autonomous platform engineering of cloud operations treats it as a collection of closed-

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

loop control systems, with agentic AI (smart agents that can sense the state of their environment, reason about complex states of operation, pursue a goal, and take action within bounds).

This architecture proposes agentic AI systems as the operational intelligence layer. They operate as agents that continuously monitor the system state using telemetry, reason about complex operational situations, make decisions based on their goal, and take actions within the policies. Whereas script-based automation and reactive AIOps notify human operators, agentic AI systems ingest experience over time, reason given context, and adapt based on unseen experiences while complying with policy constraints [3]. This suggests autonomous platform engineering is an AI-native operating model by placing smart agents in charge of operational reasoning, rather than automating against prescribed workflows. This allows them to balance competing objectives, gracefully handle edge-cases not strictly defined in the policy, tune themselves based on operational data, and still allow human operators to intervene when necessary through confidence thresholds and escalation processes.

## II. Limitations of Human-Centric Cloud Operations

### A. Operational Latency and Bottleneck Analysis

Human-centric cloud operations could introduce several sources of latency when alerting, investigating, and remediating incidents or problems. Human alert notification systems exist for alerting human operators. Human response time is dependent on the operator's availability, workload, and shift coverage. Readily visible yet latent operator load for expert teams exists. Operators must switch tasks, assess alert priority, and choose a response across modalities. Human factors research on monitoring performance and cognitive load describes context switching. Switching between operations incurs a cognitive load penalty of 20-40% of lost productivity. Resumption delays from leftover task context take 10-15 min to fully reinstate working context [4]. In a ticketed workflow, the time it takes to generate tickets, route them to other teams, track the ticket and gain approval delays remediation action.

After diagnosis, operators analyze the causes of problems, the interrelations of symptoms, the potential repair strategies, and the most appropriate solutions to complex, novel problems. For recurring problems with known solutions, the time for diagnosis can add considerable latency. Pattern-of-incident studies indicate that between 60 and 70% of production incidents fall into recurring issue classes that have known remediation runbooks [5]. When the average time to diagnose an incident is 30 to 45 minutes, even with existing runbooks that outline remediation steps, a knowledge distribution problem arises when only one person possesses the expertise for certain steps, resulting in delays as the incident waits for that expert. To reduce these bottlenecks, organizations have been documenting their processes and sharing knowledge, but locating and making sense of the available information is slow, labor-intensive work [5].

Coordination needs between teams and between different organizations can compound lead times on issues that span multiple technical areas and/or require cross-functional collaboration. Hand-offs between teams introduce delays due to the communication overhead, context switching, and misunderstanding that occur when teams exchange responsibility. Empirical studies of distributed system outages have shown that cross-team collaboration adds 45–90 minutes to median resolution latencies. Poor communication has been cited as contributing to 30-40% of the overall downtime [5]. These latencies mean that incidents often take hours or even days to resolve, when in fact they could be rooted out in seconds or minutes using automation. The peak load scenarios create stressful failure modes for the operational model due to human limits and the consequences of incident backlogs.

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

| Operational Aspect | Human-Centric Model | Autonomous Model | Primary Constraint | Scalability Pattern |
|---|---|---|---|---|
| **Alert Response** | Manual acknowledgment and evaluation | Automated detection and action | Human availability and attention | Limited by operator capacity |
| **Issue Diagnosis** | Manual investigation and correlation | Pattern recognition and inference | Cognitive load and expertise | Linear with headcount |
| **Remediation Execution** | Ticket-based workflow approval | Policy-driven automated execution | Process overhead and handoffs | Bounded by team coordination |
| **Knowledge Management** | Documentation and runbooks | Executable automation libraries | Information retrieval time | Dependent on individual expertise |
| **Decision Consistency** | Variable based on operator experience | Uniform policy-driven decisions | Expertise distribution | Fragmented across personnel |
| **Recovery Speed** | Hours to days typical | Seconds to minutes achievable | Human intervention latency | Cannot exceed human throughput |

Table 1: Human-Centric vs Autonomous Operational Paradigms in Cloud Environments [1][2][4]

### B. Talent Scarcity and Scalability Constraints

The cloud engineering talent shortage is a major bottleneck in the operational capacity and growth rate for organizations. Demand exceeds supply for cloud engineering operators with highly technical skills related to infrastructure, network, security, and application. Organizations invest heavily in recruiting, training, and retention, and high salaries, combined with a highly competitive market for rare skills and knowledge, lead to churn that can be disruptive. Cloud operators need years of exposure to problems of distributed systems, organizational context, and domain knowledge to build a repertoire of techniques and heuristics for troubleshooting and repairs. According to human factors, it often takes 5–10 years of deliberate practice and experience to cultivate advanced skills in highly technical domains, with operator performance increasing asymptotically as a function of experience rather than in a linear relationship with role tenure [6].

Classic operations scaling approaches are all based on the linear scaling of headcount. As the organization's cloud footprint scales, it exceeds its ability to hire, train and manage a growing team. This forces organizations to throttle cloud adoption, degrade service levels, or overburden operations teams to the point of burnout and turnover. The similar operations problem of workload sustainability finds that, for operations roles, when weekly workloads are sustained above 45-50 hours, error rates increase by 40-60% and turnover rates increase 2-3 times compared to those who maintain standard working hours [6]. Another operations scalability challenge, in addition to human resource challenges, is the distribution of specialized knowledge, which can lead to operational fragility when key staff are absent for any reason.

Knowledge can be transferred through training, but this imposes costs as the operators are taken away from productive work, and they need to adapt to changes in technology and processes. The cognitive load on the operator is proportional to the environment's complexity, as the number of services, configurations, and service dependencies exceeds the operator's working memory and its capacity for attention. Much of the understanding in cognitive psychology is that there is an intrinsic limit of 7±2

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

discrete pieces of information that can be stored in working memory, and that performance tends to drop sharply if there's a strain [6]. As cognitive load increases, operators will have to fall back on documentation, runbooks, and knowledge bases, which typically require their own cognitive effort to retrieve and interpret.

## III. Autonomous System Architecture Framework

### A. Closed-Loop Control System Design

The autonomous platform engineering framework is an architecture pattern in control systems with continuous loops of functionality between monitoring, analysis, decision-making, and execution in automated actions during normal operations. The monitoring subsystem constantly monitors the state of infrastructure, application, security, and compliance across cloud environments and produces streams of structured telemetry. Rather than displaying raw telemetry for human interpretation, the analysis subsystem applies pattern recognition, anomaly detection, and context awareness algorithms to identify deviations from desired states requiring corrective action. Research on autonomous control systems demonstrates that closed-loop feedback architectures can achieve nominal adjustments to operational parameters within 100-500 milliseconds from detection to action in controlled environments [7].

Instead of simple rule-based triggers or policy matching, the agentic AI reasoning of the decision subsystem considers the operational environment holistically, comparing multiple potential actions to policy objectives, assessing their downstream consequences, dependencies, and choosing the best possible actions to achieve the organization goals. This goal-oriented reasoning allows the agent to operate in an environment with a partial knowledge of the world, in a way that reconciles conflicting requirements (e.g. minimizing downtime vs. maximizing performance). to manage costs, and handles edge cases not fully covered by the policies. The agent also uses confidence scores to escalate to human control if its uncertainty is above a rejection threshold, its predicted consequences exceed an allowable risk threshold, or its observed circumstances go beyond the scope of its prior experiences [8]. The execution subsystem identifies appropriate remediation strategies, based on policies describing operational knowledge, industry best practices, and organizational policies and constraints. The execution subsystem implements remediation using infrastructure application programming interfaces (APIs), configuration management tools, and orchestration engines, while providing defined levels of safety and the ability to perform rollbacks. Within an execution framework with policy constraints, analysis results from representative benchmarks suggest that 99.5-99.9% of assessments (decisions) would be correct in systems trained on similar environments, though performance drops to 15-25% for deployable environments that were not used during training [7]. Execution results are fed into the monitoring frameworks, which allow us to determine whether remediations had the intended effect and unexpected side effects are detected, possibly leading to further action. Explicit control boundaries will determine wherein the architecture has authority to make autonomous decisions and when the control transitions to, or back to, human operators for situations beyond the system's capabilities or that are not authorized by policy.

### B. Agentic AI and Policy-Driven Intelligence Framework

The intelligent layer of autonomously operating systems consists of agentic AI systems that act as goal-directed reasoning engines under organizational policy. The agents use telemetry streams to monitor system state, maintain situational awareness, and reason over multiple remediation paths to meet policy goals, executing decisions within their granted autonomy. Unlike rule-based approaches using preprogrammed condition-response models, agents evaluate contextual information, balance trade-offs between competing objectives (e.g., performance, cost, security, compliance), and learn strategies from environmental feedback and prior experience. In addition, they

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

can reason about actions from policy objectives (even when their processes are not explicitly programmed for the particular scenario), handle uncertainty and partial information, and improve decision-making performance via experience [3].

Policy frameworks are the governance layer that constrain agentic reasoning. They are the base layer of intelligence that makes safe autonomy possible. Policies encode operational knowledge, organizational constraints, and decision logic, machine-readable at the right level of specificity for autonomous action and the right level of permissiveness for valid deviation and change. Policies prescribe optimization goals, permitted actions, escalation thresholds for risk, and constraints that cannot be overridden. Agentic systems use policies to govern autonomous intelligence operating under organizational guardrails, in contrast to boundless unconstrained artificial general intelligence [9]. Hierarchical policies define global organizational, environment-specific and service-specific policy customizations. Governance policies ensure local policies conform to enterprise policies. Empirical studies of policy management systems have found that hierarchies of 3-5 levels of abstraction can keep the overhead of policy evaluation below 10-20 ms for typical operational scenarios [9].

Declarative policy specifications focus on the desired system states, constraints, and objectives. This allows autonomous systems to determine how to achieve a policy goal based on real-time context. As systems grow and the capabilities of the infrastructure evolve, the autonomous system can leverage new features without changing the policy. Organizational policy changes are the only occasions when a policy requires modification. With respect to the pattern of policy evolution, declaratively modeled policy frameworks require only 5-10% updates in the addition of the infrastructure capabilities, while imperative procedural automation frameworks require up to 60-80% updates [9]. The policy framework comprises a variety of policy subtypes spanning functional areas, such as performance management, security enforcement, compliance checking, cost optimization, and capacity management. For multi-domain policies, conflicting policies are present in 15-20% of scenarios that require conflict resolution. Reported studies indicate automated conflict resolution can operate within 70-85% of these scenarios without human intervention [9].

| Policy Aspect | Declarative Approach | Imperative Approach | Adaptability Profile | Governance Model |
|---|---|---|---|---|
| **Specification Style** | Desired state definition | Procedural instruction | High flexibility | Constraint-based |
| **Update Requirement** | Minimal for infrastructure evolution | Frequent for capability changes | Self-adapting | Centralized oversight |
| **Conflict Resolution** | Automated constraint satisfaction | Manual procedure reconciliation | Systematic handling | Precedence hierarchies |
| **Domain Coverage** | Cross-cutting concerns | Isolated functional areas | Coordinated decisions | Federated policies |
| **Versioning Strategy** | Controlled evolution testing | Script modification tracking | Gradual rollout | Change management integration |
| **Abstraction Layers** | Hierarchical organizational to technical | Flat implementation-focused | Multi-level complexity management | Layered authority delegation |

Table 2: Policy Framework Characteristics and Design Considerations [9]

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

### C. Self-Healing Infrastructure Mechanisms

Self-healing enables infrastructure to automatically monitor itself, diagnose the root cause of failures, and make changes to return to normal operation, without requiring human intervention. Health monitoring of the system evaluates the functionality, availability, and performance of component parts, including hardware and software, against defined baselines or performance criteria. Automated diagnostics gather failure symptoms and correlate events from multiple components. They also derive root causes by pattern matching against known failure signatures and causal inference from observations of system behavior. Observed in production self-healing systems, automated diagnostics can determine root causes for component failures with 75–85% accuracy [10].

Remediation action libraries encode executable recovery procedures for common failure modes and institutionalize patterns observed in past incidents as automation rather than just documentation, but these patterns may be challenging to understand by a human at run-time. Reported data from self-healing production deployments indicates that this approach reduces human intervention in 60 to 75% of infrastructure failures. Mean time to recovery is reduced from the 45-90 minutes required for manual labor to 2-5 minutes for autonomous remediation of failures identified in these and other automated systems [10]. Verification measures are used to ensure self-healing actions have restored the system state to a healthy baseline, and customer-facing services are available. With learning-based self-healing systems, it was found that recovery repertoires could grow by 10–15% annually with experience. Newly added recovery procedures had an initial success rate of 40-50% that increased over time to 70-80% success after 6-12 months of operation [10].

| Architectural Component | Primary Function | Intelligence Type | Human Interface | Failure Mode Handling |
|---|---|---|---|---|
| **Monitoring Subsystem** | Comprehensive observability and telemetry | Continuous sensing | Passive visibility | Graceful degradation |
| **Analysis Subsystem** | Pattern recognition and anomaly detection | Diagnostic inference | Alert escalation | Confidence scoring |
| **Decision Subsystem** | Policy-driven remediation strategy selection | Constrained reasoning | Policy refinement | Boundary enforcement |
| **Execution Subsystem** | Automated action implementation | Procedural execution | Audit review | Rollback capabilities |
| **Feedback Loop** | Outcome verification and learning | Adaptive refinement | Retrospective analysis | Alternative strategy triggering |
| **Escalation Mechanism** | Uncertainty and exception handling | Confidence assessment | Direct intervention request | Human handoff protocols |

Table 3: Autonomous System Architecture Component Functions and Interactions [7] [9] [10]

### D. Agentic AI Differentiation from Traditional Approaches

The agentic AI model is key to cloud operations automation as the number of rules that can be

**Journal of Information Systems Engineering and Management**
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

**Research Article**

implemented in the cloud environment is finite and cannot cover all operational situations. Rule-Based Automation executes scripts that have already been set in response to a system event without reasoning capabilities. If the condition holds, the script is run (e.g. add instance); else nothing happens. These systems are brittle: they cannot generalize to new situations, they cannot balance competing goals, and they fail silently to situations not predicted by the conditions. They are reactive systems based on an "if-then" model without intelligence [11]. AIOps (Algorithmic IT Operations) uses machine learning to identify anomalies, correlate events and predict incidents and usually alerts human operations or creates tickets. Although AIOps improves detection and diagnosis, the final loop to problem remediation is typically closed by the human operator. Autonomic Computing (self-managing systems) uses monitor-analyze-plan-execute loops and involves a control loop. It uses user-defined policies or state machines, but lacks contextual reasoning and goal optimization for self-management. They can self-configure and self-optimize within some well defined and limited range but lack the adaptive reasoning ability to make trade-off decisions in novel situations [12].

Agentic AI Systems include perception (thorough observability across layers of infrastructure), reasoning (contextual reasoning, causal reasoning, and multi-objective optimization), goal-directed behavior (policy-constrained, yet context-sensitive), autonomous agency (within authority and confidence thresholds), and continuous learning (strategy improvement from operational experience). Agents reason through uncertainty, partial information, and competing priorities, rather than taking fixed actions according to a hard-coded logic. Contextual reasoning, multi-objective optimization, strategy adaptation, and confidence-based operation are key distinguishing features of agentic systems [11].

Contextual reasoning assists agents to go beyond knowing that CPU usage is high to understanding the context of that operation such as whether it is a batch job attack), what else is being affected (memory, network, dependent services), and the impact on the business (customer-facing vs. internal), and what constraints are at play (cost limits, compliance requirements, maintenance windows). When there are multiple objectives, it can be necessary to balance trade-offs between the different objectives [11]. For example, after a security incident, the agent may attempt to balance threat severity, business continuity impact, compliance requirements and operational constraints to decide whether to quarantine the machine, gradually reduce it or escalate it for human judgment.

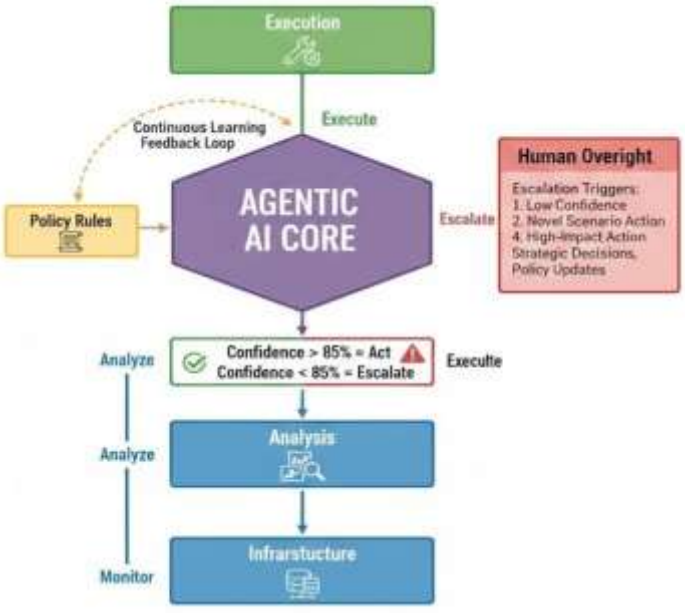


Figure 1: Agentic AI: Autonomous Operations Framework [9] [10] [11]

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

## IV. Implementation Strategy and Organizational Transition

### A. Capability Maturity Progression

Organizational learning in autonomous platform engineering is supported stepwise by models of capability maturity. Autonomy is introduced first where organizational confidence and technical maturity are lowest and is further extended as maturity increases. Early scope can be small and low risk, e.g., scaling services on predictable workloads or renewing security certificates to avoid outages when they expire. These exploratory applications serve the dual purpose of creating value from autonomy systems and training the organization to create policies, monitor autonomous performance, and intervene when necessary. Research on technology adoption shows that organizations adopting automation capabilities incrementally are 60–70% more likely to succeed than those pursuing a wholesale approach due to their iterative nature and reduced risk of change [13].

Intermediate maturity stages lead to independent actions based on more complicated decision-making processes. For example, performance optimization may involve optimizing against a set of conflicting objectives. In security, a response to a threat may need to balance the severity of the threat and business impact. Organizations develop progressively more advanced policies that encapsulate operational knowledge and organizational constraints to such a degree that they can govern autonomous agents' decisions in a broad range of situations. Operational confidence (confidence in the systems' ability to operate acceptably) is developed from experience gained by acceptance of systems' handling of different situations and knowledge of limits and escalation of those cases beyond their capability. Recent studies have found it takes organizations 18–24 months to move from pilot implementations to intermediate levels of automation maturity, where capability development is limited by policy evolution and organizational learning rather than technology [13].

Deep maturity levels have broad autonomous support for most major operational domains and human monitoring of calculated, policy, or exception tasks that fall outside the capabilities of the autonomous system. At this level, organizations govern autonomous system activities to align them with business strategies, regulations, and acceptable risk appetites. When examining the operations at the mature autonomous levels, organizations can automate 70 to 85 percent of operational decisions, with the only human inputs being governing policies, novel situations, and continuing improvement [13].

### B. Technical Architecture Requirements

Technical capabilities required for autonomous platform engineering include observability, programmable infrastructure controls or actions, and policies. Observability platforms need to enable the collection of telemetry across all layers of infrastructure, application components, and operational processes. Evidence demonstrates the importance of observability for autonomous systems, as it encompasses more structured data streams that automation systems can utilize. As such, telemetry must be collected at a rate of 1-10 data points/second/component and retained for 30-90 days in order to effectively use pattern recognition and anomaly detection algorithms [14].

Rich APIs allow for safe and systematic control of operations in infrastructure by autonomous systems; this reduces reliance on direct console access and ad hoc scripting to provide programmatic control of infrastructure. To avoid catastrophic impact to large-scale deployment due to implementation errors, several safety features (dry run, progressive rollout and automated rollback) are provided. Research on safety measures for autonomous systems shows that limiting the initial deployment to 1 to 5% of the total infrastructure capacity is effective at exposing bugs, preventing 85 to 95% of the worst-case scenarios that would occur with a full deployment [14]. Policy engines allow for the specification of operational knowledge and organizational policy in a declarative form, which is partially machine-readable, while evaluation engines are responsible for determining the appropriate actio

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

ns given the system state and policy.

| Architecture Domain | Core Capability | Implementation Characteristics | Integration Requirements | Safety Mechanisms |
|---|---|---|---|---|
| **Observability** | Comprehensive telemetry capture | Structured high-frequency data streams | Multi-layer instrumentation | Degradation detection |
| **Programmability** | Infrastructure control APIs | Consistent programmatic interfaces | Service mesh integration | Dry-run validation |
| **Policy Engine** | Declarative rule evaluation | Low-latency decision processing | Workflow system coupling | Constraint verification |
| **State Management** | Configuration tracking | Version control and history | Change management alignment | Rollback capabilities |

Table 4: Technical Architecture Foundational Requirements [14]

## V. Governance and Safety Frameworks

### A. Autonomous Decision Boundaries

The authority of the autonomous system relative to a human operator is an important requirement in the governance of autonomous systems. Autonomous system policies specify what an autonomous system is allowed to do, the scope or level of the permission, and the triggers for such permissions. Boundaries of authority balance the need for the autonomous system to be granted more autonomy at the local level, with the need to limit its authority in potentially life-critical decisions. Research on human-automation interaction shows that having well-defined decision boundaries reduces operator confusion and inappropriate reliance on automation by 50-70% compared to system boundaries that are fuzzy [15].

In the case of agentic AI systems in particular, it is important to account for the agent's reasoning because agents can derive new actions from the knowledge about their goal and their environment. This is very different from a rule-based system, which cannot derive new actions in this way. The governance of agentic systems involves specifying both permissions and constraints, allowing for policies that dictate the actions an agent can take as well as the reasoning it cannot pursue. Likewise, it may specify what tradeoffs or outcomes it cannot accept even if they meet other goals. For example, a policy may specify an agent can optimize for performance within cost but not at the cost of security posture [11]. An agent should know what it can and cannot act on and how these relate.

Trade-off boundaries need to be constructed when an agent is required to make a trade-off of some set of entities. For example, a policy could permit an agent to reduce the frequency of scanning by a security scanner to improve performance under high load, but no further. Bounded optimization only considers technically feasible solutions and rules out optimal solutions that would contradict organizational policy. Agent transparency (i.e. the ability of the agent to explain their decisions) ensures that operators can verify that an autonomous procedure is in line with the organization's intent, despite the task itself not being bound to policy. Audit logs must, furthermore, record how the agent arrived at such a decision, what alternatives were considered, and what trade-offs were made [11]. Agents have to be able to detect novel situations, such as when they are acting outside the domain of their experience or coverage, rather than extrapolating inappropriately. This requires reasoning

about the limits of its knowledge and its confidence in its inferences.

Escalation includes scenarios in which the confidence of the autonomous system (AS) is below a certain threshold, the risk of the action exceeds a specified threshold, or the current situation resembles a scenario where the autonomous action has previously failed. Escalation is graduated, so the AS can automatically request permission for actions with low confidence or risk or defer decision-making for more critical actions. Risk assessment mechanisms model the impact of potential autonomous actions and escalate decisions beyond autonomous authority when the predicted impacts exceed risk tolerance. Simulations of autonomous systems responding to risky situations suggest that appropriately set thresholds would lead to human escalation rates of 5–15% during normal operations and 25–40% during abnormal operations or transitions to a new operational mode where uncertainty is considerably increased [15].

Novel situation detection identifies situations outside of previous operational experience and the autonomous system's experience and policy. This lowers the risk of applying existing policy to operational situations with different policy requirements and deferring human decision-making to operational situations that are truly novel in nature. To counteract this, boundary enforcement mechanisms impose technical restrictions that prevent the system from operating outside of authorized limits regardless of its policy logic.

### B. Audit, Compliance, and Continuous Improvement

An audit capability can capture all of the decisions and actions made by an autonomous system and can support governance and regulatory oversight, incident investigations, and performance analysis. Audit logs capture the rationale for autonomous decisions made by the agent and can include the policy rules, their evaluations, and alternative plans considered. Structured log formats may enable developers to automatically analyze logs, and identify patterns of autonomous behavior or flag violations of policy or unusual environmental conditions. Research on requirements for the auditing of autonomous systems indicates that log retention times of 12-36 months may be necessary for regulatory and forensic purposes. Structured logging can also lead to automatic compliance checks, potentially reducing 60–80% of the time spent on audits compared to unstructured logging solutions [16].

Compliance monitoring approaches ensure that an autonomous system's actions comply with regulations, organizational policies, and industry standards in the execution of operational activities. Automated compliance checking techniques ensure that proposed actions comply with the applicable compliance constraints and can prevent the autonomous system from executing technically feasible actions that violate compliance. Continuous improvement processes monitor system performance to identify opportunities for increasing automation, improving policies, and improving decision-making policies. Automated metrics include the number of successful autonomous resolutions, escalations, and false-positive detections. Time-to-resolution improvements compared to the manual baseline are also tracked. The benefits of continuous improvement of autonomous operations are illustrated by organizations that rely on established feedback loops, with reported annual gains in accuracy of decision-making in autonomous operations between 15 and 25% and reduced unnecessary escalations between 10 and 20% through policy refinement [16].

## Conclusion

Once autonomous platform engineering is viewed as a policy-governed agentic control plane, the focus on human-in-the-loop operational intelligence shifts to an understanding of closed-loop autonomous operational intelligence. This paper synthesizes agentic AI systems characterized by contextual reasoning, goal-directed decision making, and adaptive learning to form the operational

intelligence layer governed by declarative policies. It differs from rule-based automation, AIOps, and customary autonomic computing in terms of adaptation and the ability to deal with complexity, uncertainty, and novelty within human-defined governance boundaries. This helps address the cloud skills shortage, and can scale to far greater levels than human labor. With a formal progression up the capability maturity model, and appropriate observability infrastructure and operating model governance (decision boundaries and a formal approach to continuous improvement), organizations can build operational confidence. Consequently, autonomous platform engineering is transformed from an experimental capability into an operational foundation, with evolved policies governing clever agents at scale, leaving human decision-makers for planned governance.

## References

[1] Joel Scheuner and Philipp Leitner, "Function-as-a-Service Performance Evaluation: A Multivocal Literature Review," Journal of Systems and Software, 2020. Available: https://doi.org/10.1016/j.jss.2020.110708

[2] Daniel P. Jenkins et al., "Human Factors Methods A Practical Guide for Engineering and Design (second edition)," Ashgate, 2013. Available: https://www.researchgate.net/publication/267624130_Human_Factors_Methods_A_Practical_Guide_for_Engineering_and_Design_second_edition

[3] Ranjan Sapkota et al., "Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges," arXiv preprint arXiv, 2025. Available: https://arxiv.org/pdf/2505.10468

[4] Mary Czerwinski et al., "A Diary Study of Task Switching and Interruptions," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2004. Available: https://dl.acm.org/doi/10.1145/985692.985715

[5] Peter Bodik et al., "Characterizing, Modeling, and Generating Workload Spikes for Stateful Services," in Proceedings of the 1st ACM Symposium on Cloud Computing, 2010. Available: https://dl.acm.org/doi/10.1145/1807128.1807166

[6] Ericsson, K. Anders, et al., "The Role of Deliberate Practice in the Acquisition of Expert Performance," Psychological Review, 1993. Available: https://psycnet.apa.org/record/1993-40718-001

[7] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," Computer, 2003. Available: https://ieeexplore.ieee.org/document/1160055

[8] Jerry Swan et al., “The road to general intelligence,” Springer Nature, 2022. Available: https://library.oapen.org/bitstream/handle/20.500.12657/57384/1/978-3-031-08020-3.pdf

[9] M. Sloman and E. Lupu, "Security and Management Policy Specification," IEEE Network, 2002. Available: https://ieeexplore.ieee.org/document/993218

[10] D. Garlan et al., "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," Computer, 2004. Available: https://ieeexplore.ieee.org/document/1350726

[11] Soodeh Hosseini and Hossein Seilani, "The role of agentic AI in shaping a smart future: A systematic review," Array, 2025. Available: https://www.sciencedirect.com/science/article/pii/S2590005625000268

[12] Rahul Gaikwad et al., "A framework design for algorithmic it operations (AIOps)," Design Engineering, 2021. Available: https://www.researchgate.net/profile/Dr-Rahul-Gaikwad/publication/354632690_Design_Engineering_A_Framework_Design_for_Algorithmic_IT_Operations_AIOPS/links/61435e5e5c251a5966486172/Design-Engineering-A-Framework-Design-

for-Algorithmic-IT-Operations-AIOPS.pdf

[13] Everett M. Rogers, "Diffusion of Innovations," 3rd ed., Free Press, 1983. Available: https://teddykw2.wordpress.com/wp-content/uploads/2012/07/everett-m-rogers-diffusion-of-innovations.pdf

[14] Benjamin H. Sigelman et al., "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure," Google Technical Report, 2010. Available: https://static.googleusercontent.com/media/research.google.com/en//archive/papers/dapper-2010-1.pdf

[15] R. Parasuraman et al., "A Model for Types and Levels of Human Interaction with Automation," IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2000.Available: https://ieeexplore.ieee.org/document/844354

[16] Nancy G. Leveson and Clark S. Turner, "An Investigation of the Therac-25 Accidents," Computer, 1993. Available: https://ieeexplore.ieee.org/document/274940