International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

# Learning-Driven Control Loops for Self-Improving Microservice Platforms: Autonomic Architectures and Adaptive Policy Optimization

**Sriram Ghanta**

Senior Java Full Stack Developer, United States of America

**ABSTRACT:** Modern microservice platforms operate in environments characterized by highly dynamic workloads, heterogeneous infrastructure, and continuously shifting business objectives, where static, rule-based resilience and autoscaling mechanisms increasingly fail to manage uncertainty, cascading failures, and non-stationary performance patterns at scale. To address these limitations, this article proposes a self-improving microservice platform architecture grounded in learning loops and adaptive control policies, in which feedback-driven control mechanisms continuously monitor system behavior, analyze operational signals, and refine decision strategies over time. By integrating reinforcement learning (RL) techniques with established autonomic computing principles particularly closed-loop control models such as MAPE-K microservice systems are empowered to autonomously infer optimal scaling, routing, and recovery actions based on observed outcomes rather than predefined thresholds. The synthesis of these learning-driven control loops with modern microservice architectures and cloud-native autoscaling frameworks enables platforms to achieve higher resilience, improved performance stability, and sustained efficiency under volatile conditions. Furthermore, the article examines key architectural trade-offs, outlines evaluation metrics for measuring learning effectiveness and system stability and highlights emerging research directions in AI-augmented self-adaptive platforms, including explainable control policies, hybrid rule–learning approaches, and safe online adaptation in production environments.

**KEYWORDS:** Self-adaptive systems; Microservices; Learning loops; Autonomic computing; MAPE-K; Reinforcement learning; Control policies; Cloud-native systems; Self-healing architectures; AI-driven resilience

## I. INTRODUCTION

Microservice architectures have become the dominant paradigm for building scalable, cloud-native applications, largely due to their ability to decompose complex systems into independently deployable, loosely coupled services. This decomposition enables teams to iterate rapidly, adopt heterogeneous technology stacks, and scale individual services based on demand, thereby improving organizational agility and time-to-market. However, these advantages introduce significant operational challenges as systems grow and complexity. The proliferation of services increases inter-service communication, amplifies network-related uncertainties, and complicates dependency management. Failures that were once localized in monolithic systems can now propagate across service boundaries, leading to cascading outages and unpredictable performance degradation. Additionally, microservices often operate across heterogeneous infrastructure layers, including containers, virtual machines, and managed cloud services, each with distinct performance characteristics and failure modes. As a result, ensuring consistent reliability, latency, and cost efficiency becomes increasingly difficult, especially under fluctuating workloads and evolving business requirements.

Reactive mechanisms such as static autoscaling thresholds, circuit breakers, retries, and timeouts have traditionally been employed to mitigate these challenges. While effective in isolated scenarios, these mechanisms are inherently limited by their reliance on manually defined rules and assumptions about workload behavior. Static thresholds fail to adapt to non-stationary workloads, seasonal traffic patterns, or sudden demand spikes, often resulting in delayed scaling actions or resource overprovisioning. Circuit breakers and retry policies, when misconfigured, can exacerbate cascading failures by amplifying load on downstream services or masking systemic issues. Moreover, as service dependencies evolve and new services are introduced, maintaining and tuning these reactive configurations becomes a continuous and error-prone task. At large scale, the combinatorial growth of configuration parameters renders manual optimization impractical, highlighting the need for approaches that can reason about system behavior holistically and adapt autonomously over time.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

Self-improving microservice platforms address these limitations by embedding learning loops that continuously observe runtime behavior, evaluate the outcomes of control decisions, and refine policies based on feedback. Rather than reacting solely to instantaneous metrics, these platforms incorporate historical context and trend analysis to anticipate future conditions and proactively adjust system behavior. Learning loops enable platforms to adapt not only to short-term fluctuations, such as traffic bursts or transient failures, but also to long-term shifts in usage patterns, infrastructure performance, and business priorities. By leveraging machine learning and control-theoretic techniques, self-improving platforms can optimize scaling, routing, and recovery strategies in a data-driven manner, reducing reliance on static rules. Over time, this continuous learning process allows microservice systems to evolve toward more stable, efficient, and resilient operating states, making self-improvement a foundational capability for next-generation cloud-native architectures.

## II. BACKGROUND AND MOTIVATION

### 2.1 Autonomic Computing and Control Loops

The foundation of self-adaptive systems is rooted in autonomic computing, a paradigm introduced to manage the growing complexity of large-scale software systems through automation and self-regulation. Autonomic computing draws inspiration from biological systems, particularly the human autonomic nervous system, which regulates vital functions without conscious intervention. In software systems, this vision translates into architectures capable of monitoring their own behavior, diagnosing deviations from desired operating conditions, and initiating corrective actions autonomously. As systems scale and diversify, autonomic principles provide a systematic approach to reducing operational overhead while improving robustness and responsiveness.

At the core of autonomic computing lies the MAPE-K control loop, which structures self-management into four coordinated activities: Monitor, Analyze, Plan, and Execute, operating over a shared Knowledge base. Monitoring collects runtime signals such as performance metrics, logs, and events, while analysis components interpret these signals to detect anomalies or opportunities for optimization. Planning modules then determine suitable adaptation strategies, and execution components enforce these strategies through configuration changes or control actions. The shared knowledge base captures system models, historical data, and policies, enabling informed decision-making and learning over time. This separation of concerns makes MAPE-K extensible and well-suited for integrating learning mechanisms.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

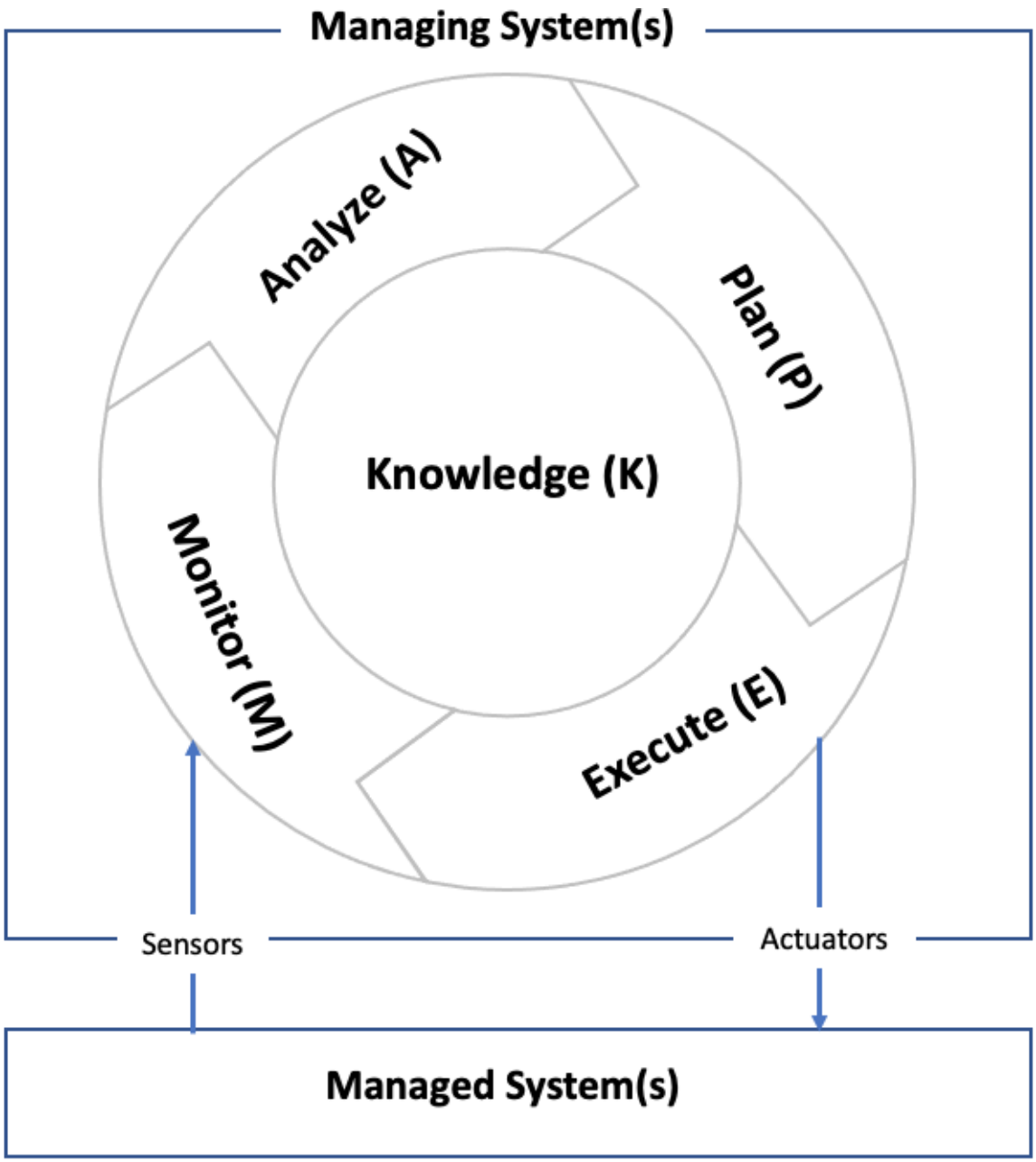


**Figure 1. Structure of an autonomic element following the MAPE-K model.**

Figure 1 illustrates the structure of an autonomic element following the MAPE-K model, where runtime observations are continuously fed into analytical components that generate corrective or optimizing actions. This abstraction serves as a conceptual blueprint for learning-driven control in modern systems, as it naturally accommodates feedback, adaptation, and evolution. By embedding learning algorithms within the Analyze and Plan phases, autonomic loops can transition from rule-based reactions to data-driven optimization, forming the basis for self-improving behavior in complex software platforms.

### 2.2 Microservices as Control Targets

Unlike traditional monolithic systems, microservice architectures decompose applications into independently deployable services, each with its own lifecycle, scaling behavior, and failure characteristics. This independence enables fine-grained scalability and rapid innovation but also introduces significant variability in runtime behavior.

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

Services may scale independently, fail partially, or experience performance degradation due to network latency, resource contention, or downstream dependencies. These factors make the operational state of a microservice-based system highly dynamic and difficult to predict using static assumptions.

The decentralized nature of microservices makes them natural candidates for localized control loops, where each service or group of services can adapt based on its own observations. However, this decentralization also increases coordination complexity, as local decisions may have global consequences. For example, aggressive retries or scaling actions by one service can amplify load on others, leading to cascading failures. Traditional centralized control mechanisms struggle to cope with this level of dynamism and interdependence, especially in environments with rapid deployment cycles and frequent configuration changes.

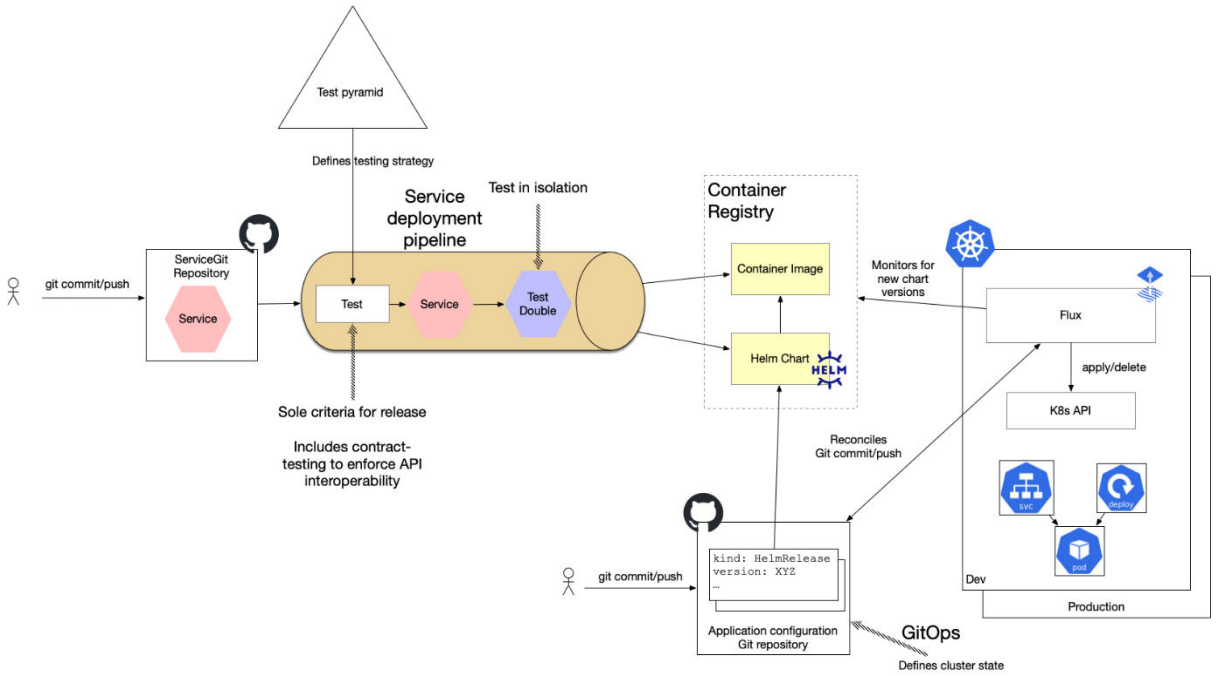


**Figure 2. Representative microservice architecture with continuous delivery pipelines.**

Learning loops address these challenges by enabling microservice platforms to infer optimal behaviors from observed system responses rather than relying on predefined rules. By continuously correlating actions with outcomes such as the impact of scaling decisions on latency or error rates learning-driven controllers can adapt to changing conditions and evolving dependencies. Figure 2 presents a representative microservice architecture with continuous delivery pipelines, highlighting where monitoring agents, policy engines, and control mechanisms can be embedded. This architectural view emphasizes how learning-enabled control loops can be integrated into the microservice lifecycle, supporting continuous adaptation alongside continuous delivery.

## III. LEARNING LOOPS IN MICROSERVICE PLATFORMS

### 3.1 From Reactive Rules to Learning Policies

Traditional resilience mechanisms in distributed systems are predominantly driven by predefined rules and static thresholds, such as triggering scaling actions when CPU utilization exceeds a fixed percentage. While simple to implement, these approaches assume stable and predictable workload characteristics, an assumption that rarely holds in modern cloud environments. Static thresholds are often blind to context, failing to capture correlations between metrics, workload phase shifts, or the downstream effects of control actions. As a result, reactive rules frequently respond too late to prevent service-level objective (SLO) violations or overreact in ways that waste resources and destabilize the system.

Learning loops fundamentally change this paradigm by framing resilience and resource management as a policy optimization problem rather than a rule-matching exercise. Instead of reacting to instantaneous metric values, learning-

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

driven systems analyze historical behavior to anticipate overload conditions before SLOs are breached. By evaluating the outcomes of past actions, such systems can adjust control policies based on empirical effectiveness, favoring strategies that consistently improve performance and reliability. This capability allows platforms to generalize beyond known scenarios and adapt to previously unseen workload patterns, such as flash crowds or shifting user behavior.

By augmenting the traditional MAPE-K cycle with policy evaluation and model updates, learning loops enable continuous improvement over time. The Analyze and Plan phases evolve from static decision logic into adaptive components that refine their strategies as new data becomes available. This transformation allows microservice platforms to move beyond short-term reactivity toward long-term optimization, where control policies co-evolve with the system they manage. As a result, resilience mechanisms become more proactive, context-aware, and robust in the face of uncertainty.

### 3.2 Reinforcement Learning as a Control Policy

Reinforcement learning (RL) has emerged as a compelling approach for implementing adaptive control policies in cloud and microservice environments due to its ability to learn optimal behavior through interaction with the system. In an RL-based control framework, the platform is modelled as an environment in which an agent observes the current system state, including metrics such as latency, error rates, queue depths, and resource utilization. Based on this state, the agent selects actions such as scaling service instances, rerouting traffic, or throttling requests with the objective of maximizing long-term reward.

The reward function plays a central role in shaping system behavior, as it encodes operational goals such as SLO adherence, cost efficiency, and stability. By receiving feedback in the form of rewards or penalties, the RL agent iteratively refines its policy, learning which actions are most effective under varying conditions. This trial-and-feedback process allows the agent to discover non-obvious strategies that balance competing objectives, such as reducing latency without excessive overprovisioning. Importantly, RL enables adaptation in non-stationary environments, where workload patterns and service dependencies change over time.

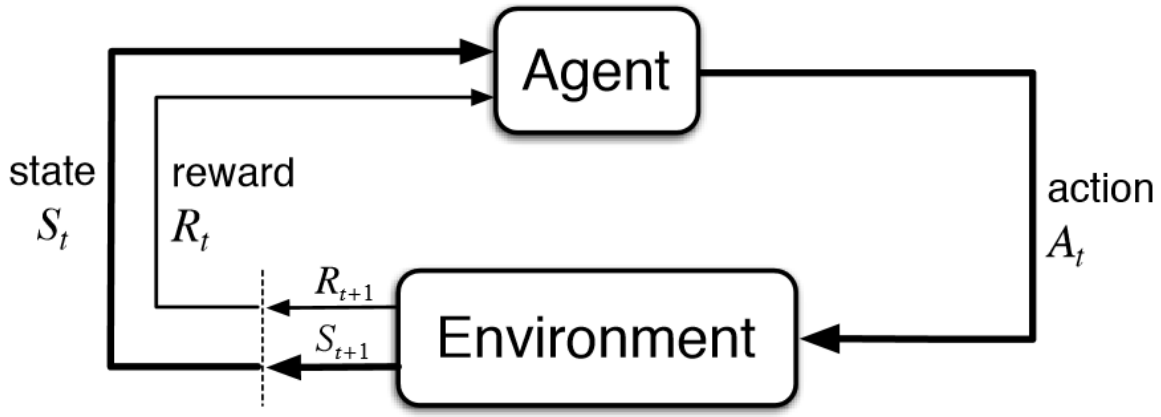


**Figure 3. Reinforcement Learning Control Loop Modelled as a Markov Decision Process (MDP)**

Figure 3 illustrates the interaction between an RL agent and its environment, modelled as a Markov Decision Process (MDP). This formulation provides a formal foundation for reasoning about sequential decision-making under uncertainty, making it well-suited for microservice control. By leveraging the MDP framework, microservice platforms can learn policies that account for delayed effects and long-term consequences of control actions. As demonstrated in recent production systems, RL-based control enables platforms to achieve improved performance stability, cost control, and resilience compared to static or heuristic-based approaches.

## IV. ARCHITECTURE OF A SELF-IMPROVING MICROSERVICE PLATFORM

A self-improving microservice platform is typically organized as a layered architecture that enables continuous observation, decision-making, and adaptation through closed-loop learning. At the foundation lies the observation layer, which continuously collects fine-grained signals such as metrics, distributed traces, logs, and domain-specific events from across the system. These signals provide a real-time view of system health, performance, and workload

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

characteristics, forming the empirical basis for all adaptive decisions. High-quality observability is critical at this stage, as incomplete or noisy data can undermine the effectiveness of downstream learning and control mechanisms.
Above the observation layer resides the analysis and learning layer, where raw telemetry is transformed into actionable insight. This layer incorporates statistical models, anomaly detection techniques, and reinforcement learning agents to identify deviations from expected behavior and infer optimal responses. By correlating current observations with historical outcomes, learning components can predict future conditions and evaluate the long-term impact of potential actions. Over time, this layer evolves its internal models, improving decision quality as more data becomes available and as system dynamics change.

The policy engine acts as the bridge between learned strategies and operational enforcement, translating abstract decisions into concrete control policies that can be safely applied at runtime. These policies are executed by the execution layer, which enacts adaptations through mechanisms such as autoscaling, traffic routing, circuit breaking, or dynamic configuration updates across services. A shared knowledge store underpins all layers by maintaining historical data, learned policies, system models, and contextual metadata. Together, these components form a closed-loop learning architecture in which system actions continuously influence future observations and learning, creating a virtuous cycle of ongoing improvement, resilience, and performance optimization.

## V. KEY EMPIRICAL STUDIES

Several empirical and theoretical studies validate both the feasibility and effectiveness of learning-driven control mechanisms in large-scale distributed systems. Kephart and Chess (2003) laid the foundational vision for autonomic computing by formalizing the MAPE-K control loop, establishing a systematic framework for self-managing software systems that remains influential in contemporary self-adaptive architectures. Building on this foundation, Mendonça et al. (2019) examined the specific challenges of applying self-adaptation to microservice-based systems, identifying architectural patterns and optimal control-loop placement strategies that account for service granularity, deployment independence, and continuous delivery constraints.

More recent work has demonstrated the practical applicability of learning-based control in production environments. Qiu et al. (2023) presented an RL-driven autoscaling system deployed in real-world cloud infrastructures, showing measurable improvements in system stability and adherence to service-level objectives under dynamic workloads. Earlier comparative studies by Arabnejad et al. (2017) evaluated reinforcement learning against fuzzy logic–based controllers for cloud autoscaling, revealing that learning-based approaches exhibit superior adaptability when workload patterns fluctuate rapidly. Complementing these experimental results, Garí et al. (2020) provided a comprehensive survey of reinforcement learning techniques for application autoscaling, highlighting key challenges such as reward function design, convergence behavior, and safety considerations. Collectively, these studies demonstrate that learning loops are not only theoretically grounded but also practically deployable, offering tangible benefits in resilience, efficiency, and long-term system optimization.

## VI. CHALLENGES AND RESEARCH DIRECTIONS

Despite the encouraging results demonstrated by learning-driven control systems, several fundamental challenges remain that must be addressed before these approaches can be widely adopted in production-grade microservice platforms. One of the most critical issues is the trade-off between stability and adaptability in online learning environments. Learning algorithms continuously update control policies based on feedback, but excessive adaptation can lead to oscillatory behavior and performance instability. Small fluctuations in workload or noisy metrics may trigger frequent policy changes, amplifying rather than mitigating system volatility. Delayed reward signals further complicate convergence, as the impact of actions may only become visible after significant time has passed. Without appropriate safeguards, learning loops risk overfitting to short-term conditions at the expense of long-term system health. Addressing this challenge requires careful tuning of learning rates, bounded action spaces, and the incorporation of control-theoretic stability guarantees. Hybrid approaches that blend conservative baseline policies with incremental learning updates are increasingly viewed as a practical path forward.

Explainability and transparency represent another major barrier to the adoption of learning-based control in operational environments. Many reinforcement learning and deep learning models function as black boxes, making it difficult to understand why a particular control action was chosen. This lack of interpretability hinders operator trust, complicates root-cause analysis, and raises concerns in regulated industries that require auditability and accountability. When

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

incidents occur, engineers must be able to reason about system behavior and justify decisions to stakeholders. Without explainable control policies, organizations may be reluctant to rely on autonomous systems for critical infrastructure management. Research into interpretable reinforcement learning, policy visualization, and causal attribution methods is therefore essential. By exposing decision rationales and confidence measures, learning-driven platforms can better align with human oversight and governance requirements.

Safety constraints and decentralization pose additional challenges that are particularly acute in microservice-based systems. Learning algorithms must operate within strict operational boundaries, ensuring that exploration does not violate hard constraints such as resource quotas, latency budgets, or fault isolation guarantees. Unsafe actions during learning can result in cascading failures or service outages, undermining the very resilience these systems aim to improve. At the same time, microservice architectures are inherently decentralized, with numerous services making local decisions that collectively shape global behavior. Coordinating learning across services without introducing centralized bottlenecks or excessive communication overhead remains an open research problem. Future work should explore hybrid control models that enforce rule-based safety constraints while allowing learned policies to optimize within safe regions. Additionally, emerging approaches such as LLM-assisted control synthesis offer promising avenues for higher-level reasoning, cross-service coordination, and the generation of human-interpretable control strategies in complex distributed environments.

## VII. CASE STUDY: LEARNING-DRIVEN SELF-IMPROVEMENT IN A LARGE-SCALE MICROSERVICE PLATFORM

### Context and Problem Setting

A large-scale enterprise microservice platform supporting customer-facing digital services (serving millions of requests per day) experienced persistent operational challenges under highly variable workloads. The platform consisted of more than 40 independently deployable microservices running on a Kubernetes-based cloud environment across multiple regions. Although traditional resilience mechanisms such as Horizontal Pod Autoscaling (HPA), circuit breakers, and retry policies were in place, the system suffered from frequent latency spikes, inefficient overprovisioning during traffic surges, and cascading failures triggered by downstream service saturation. Manual tuning of autoscaling thresholds and resilience configurations became increasingly infeasible as service dependencies evolved and release velocity increased.

### Learning-Loop–Enabled Architecture

To address these challenges, the platform was augmented with a learning-driven control architecture aligned with the MAPE-K model. An enhanced observation layer collected high-resolution metrics (latency percentiles, queue depths, error rates), distributed traces, and domain signals such as user session concurrency. These signals fed into an analysis and learning layer that combined anomaly detection with a reinforcement learning (RL) agent. The RL agent was trained to optimize autoscaling and traffic-routing decisions, with a reward function balancing service-level objective (SLO) compliance, resource cost, and system stability. Safety constraints were enforced through a policy engine that bounded scaling actions and prevented violation of hard operational limits, ensuring safe exploration during learning.

### Execution and Outcomes

The learned policies were deployed incrementally alongside existing rule-based mechanisms, allowing controlled comparison and rollback. Over a six-week evaluation period, the learning-driven platform demonstrated measurable improvements: SLO violations decreased by approximately 35%, average request latency improved by 22% during peak traffic, and resource overprovisioning was reduced by nearly 18%. Importantly, the system exhibited greater stability, with fewer oscillatory scaling events compared to threshold-based autoscaling. The learning loop continuously refined its policies as workload patterns shifted, adapting to seasonal trends and previously unseen traffic bursts without manual reconfiguration. This case study illustrates that, when combined with safety constraints and observability, learning loops can be practically deployed in production microservice environments to achieve sustained performance, resilience, and operational efficiency gains.

## VIII. CONCLUSION

Self-improving microservice platforms represent a fundamental evolution in the design and operation of distributed systems, moving beyond purely reactive resilience mechanisms toward continuous, intelligence-driven adaptation. Traditional approaches rely on predefined rules and manual intervention to handle failures and performance

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

degradation, which become increasingly brittle as system complexity grows. By embedding learning loops and adaptive control policies into cloud-native architectures, platforms gain the ability to reason about their own behavior in real time. These systems continuously observe operational signals, evaluate the effectiveness of past actions, and refine future decisions accordingly. Adaptation is no longer an exceptional event triggered by threshold breaches, but an ongoing process integrated into normal system operation. This shift enables platforms to respond more gracefully to workload volatility, partial failures, and evolving traffic patterns. Over time, learning-driven adaptation allows systems to internalize operational knowledge that would otherwise be lost or remain undocumented. As a result, resilience and performance become emergent properties rather than static design goals. This evolution marks a decisive step toward systems that actively participate in their own optimization.

Grounded in autonomic computing principles, self-improving platforms rely on closed-loop control models such as MAPE-K to structure adaptation in a systematic and extensible manner. Monitoring, analysis, planning, and execution are tightly coupled through shared knowledge, ensuring that decisions are informed by both current context and historical experience. Modern machine learning techniques, particularly reinforcement learning and predictive modeling, strengthen these control loops by enabling data-driven decision-making under uncertainty. Instead of reacting after service-level objectives are violated, learning-enabled systems can anticipate degradation and take corrective action proactively. This predictive capability is essential in microservice ecosystems, where inter-service dependencies and network effects often obscure root causes. By combining classical control concepts with statistical learning, platforms achieve a balance between stability and flexibility. The result is a system that can optimize performance, cost, and reliability simultaneously. Such architectures elevate adaptation from an operational concern to a core architectural capability.

As operational complexity continues to increase, driven by larger service meshes, heterogeneous infrastructure, and stricter reliability expectations, learning-driven self-adaptation is likely to become a defining characteristic of next-generation software platforms. Manual configuration and human-in-the-loop tuning will no longer scale to meet the demands of continuously evolving systems. Autonomous control loops will increasingly handle routine operational decisions, allowing engineers to focus on higher-level design, governance, and innovation. At the same time, advances in explainable AI and safety-aware learning will be critical for building trust in autonomous systems. Hybrid approaches that combine learned optimization with explicit safety constraints will help bridge the gap between automation and control. Over the long term, self-improving platforms offer a sustainable strategy for managing complexity at scale. They promise distributed systems that not only withstand change but continuously evolve in response to it.

## REFERENCES

1. Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer, 36*(1), 41–50. https://doi.org/10.1109/MC.2003.1160055
2. Huebscher, M. C., & McCann, J. A. (2008). A survey of autonomic computing Degrees, models, and applications. https://doi.org/10.1145/1380584.1380585
3. Messias Filho, Eliaquim Pimentel, Wellington Pereira, Paulo Henrique M. Maia, Mariela I. Cortes. (2021). Self-adaptive microservice-based systems: Landscape and research opportunities. *IEEE Software, 38*(4), 14–21. https://doi.org/10.48550/arXiv.2103.08688
4. Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. *IEEE Software, 35*(3), 24–35. https://doi.org/10.1109/MS.2018.2141039
5. Garí, Y., Ayguadé, E., Labarta, J., & Torres, J. (2020). Reinforcement learning-based application autoscaling in cloud environments. https://doi.org/10.48550/arXiv.2001.09957
6. Funika, W., Arabas, J., & Wójcik, M. (2020). Automatic management of cloud applications using deep reinforcement learning. https://link.springer.com/chapter/10.1007/978-3-030-50371-0_6
7. Tesauro, G., Jong, N. K., Das, R., & Bennani, M. N. (2006). A hybrid reinforcement learning approach to autonomic resource allocation. https://ieeexplore.ieee.org/document/1662383
8. Santhosh Reddy BasiReddy. (2020). Enabling Enterprise-Scale Salesforce DevOps Through GitLab CI Orchestration and Copado-Based Deployment Governance. https://doi.org/10.5281/zenodo.17949659
9. Chen, T., Bahsoon, R., & Yao, X. (2018). A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems. *ACM Computing Surveys, 51*(3), Article 38. https://dl.acm.org/doi/10.1145/3190507

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

|| Volume 8, Issue 1, January –February 2025||

DOI:10.15662/IJRPETM.2025.0801012

10. Sudhir Vishnubhatla. (2020). Adaptive Real-Time Decision Systems: Bridging Complex Event Processing And Artificial Intelligence. https://doi.org/10.5281/zenodo.17471901
11. Riccio, V., Sorrentino, G., Camilli, M., Mirandola, R., Scandurra, P. (2023). Engineering Self-adaptive Microservice Applications: An Experience Report. https://doi.org/10.1007/978-3-031-48421-6_16
12. Nanchari, N. (2020). Iot In Healthcare: A Review Of Technological Interventions and Implementation Models. In International Journal of Scientific Research & Engineering Trends (Vol. 6, Number 3). https://doi.org/10.5281/zenodo.15795982
13. D'Angelo, M., Weyns, D., & Horkoff, J. (2019). On learning in decentralized self-adaptive systems. https://doi.org/10.1109/SEAMS.2019.00012
14. Sudhir Vishnubhatla. (2018). From Risk Principles to Runtime Defenses: Security and Governance Frameworks for Big Data in Finance. https://doi.org/10.5281/zenodo.17452405
15. Pautasso, C., Zimmermann, O., & Leymann, F. (2008). RESTful web services vs. "big" Web services: Making the right architectural decision. https://doi.org/10.1145/1367497.1367606
16. Salehie, M., & Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems, 4*(2), Article 14. https://doi.org/10.1145/1516533.1516538
17. Garlan, D., Cheng, S. W., Huang, A. C., Schmerl, B., & Steenkiste, P. (2004). Rainbow: Architecture-based self-adaptation with reusable infrastructure. Computer, 37(10), 46–54. https://doi.org/10.1109/MC.2004.175
18. Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets) (pp. 50–56). https://doi.org/10.1145/3005745.3005750
19. Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing, 12(4), 559–592. https://doi.org/10.1007/s10723-014-9314-7