Runtime Uncertainty in Microservice Systems:

The Need for Explainable, Reasoning-Driven Self-Adaptation

Abstract

Microservice architectures have become the dominant paradigm for building large-scale, cloud-native systems due to their scalability, modularity, and deployment flexibility. However, the operational benefits of microservices come at the cost of increased runtime complexity and uncertainty. Contemporary self-adaptation mechanisms—such as rule-based autoscaling, threshold-driven controllers, and reactive fault-handling strategies—primarily focus on maintaining system performance but provide little insight into *why* adaptation decisions are made. This lack of explainability limits operator trust, auditability, and effective decision-making, particularly in high-stakes and regulated domains such as financial technology. This paper presents a conceptual analysis of runtime uncertainty in microservice systems and critically examines the limitations of existing adaptation approaches with respect to reasoning transparency. We argue that explainability must be treated as a first-class requirement in runtime self-adaptation and position large language models (LLMs) as reasoning and explanation agents capable of synthesizing contextual information and articulating adaptation rationales. Rather than proposing a concrete architecture, this paper establishes the theoretical and conceptual foundations necessary for explainable, intelligence-driven self-adaptive systems.

Keywords

Microservices, Runtime Adaptation, Explainable Systems, Autonomic Computing, Large Language Models, Self-Adaptive Systems

1. Introduction

Microservice architectures have emerged as a foundational design paradigm for modern distributed systems, enabling independent deployment, scalability, and technological heterogeneity at scale [1]. Cloud platforms and container orchestration frameworks such as Kubernetes have further accelerated their adoption by providing automated resource management and fault-tolerance mechanisms [2]. Despite these advantages, microservices introduce significant runtime complexity due to their highly dynamic behavior, decentralized control, and intricate service interactions.

To address this complexity, self-adaptation mechanisms have been widely adopted. These mechanisms aim to adjust system behavior at runtime in response to changing environmental conditions, workload fluctuations, and failures. Common approaches include metric-driven autoscaling, rule-based controllers, and reactive self-healing strategies [3]. While effective at maintaining availability and performance, such mechanisms are largely *opaque*: they execute adaptation actions without providing intelligible explanations of the underlying reasoning.

This opacity poses serious challenges. System operators are often unable to understand why a scaling decision was triggered, why a particular mitigation strategy was chosen, or what alternative actions were considered. In high-stakes domains such as fintech, healthcare, and critical infrastructure, this lack of transparency undermines trust, auditability, and regulatory compliance [4].

This paper argues that explainability is not merely a usability feature but a fundamental architectural requirement for runtime self-adaptation in microservice systems. We further contend that recent advances in large language models (LLMs) enable a new class of reasoning-driven, explainable adaptation capabilities. Rather than directly controlling system behavior, LLMs can act as reasoning and explanation layers that interpret system context and articulate adaptation rationales in human-understandable terms.

The contributions of this paper are threefold:

1. A structured taxonomy of runtime uncertainty in microservice systems, highlighting how load variability, partial failures, emergent behavior, and cloud environmental factors challenge existing self-adaptation mechanisms.
2. A critical analysis of contemporary runtime adaptation approaches, demonstrating their limitations in reasoning transparency, explainability, and operator trust.
3. A conceptual positioning of large language models as reasoning and explanation agents in self-adaptive systems, clearly delineating their advisory role from deterministic control execution.
4. An articulation of explainability requirements for runtime adaptation, framing explainability as a first-class software quality attribute relevant to regulated and high-stakes systems.

2. Background and Related Work

2.1 Microservice Architectures

Microservices decompose applications into independently deployable services that communicate over lightweight protocols [1]. This decomposition enhances scalability and development agility but increases operational complexity due to network latency, partial failures, and emergent behavior [5]. Observability frameworks have been introduced to monitor such systems, yet monitoring alone does not equate to intelligent adaptation [6].

2.2 Runtime Self-Adaptation

Self-adaptive systems are commonly structured around feedback-control loops, such as the MAPE-K model (Monitor, Analyze, Plan, Execute, Knowledge) [7]. These models have been applied extensively in autonomic computing and cloud systems. However, most practical implementations rely on predefined rules, thresholds, or optimization heuristics, limiting their ability to reason under uncertainty [8].

2.3 Explainable AI in Software Systems

Explainable AI (XAI) seeks to make machine-learning systems transparent and interpretable to humans [9]. While XAI has been widely studied in decision-support and classification contexts, its application to runtime software adaptation remains limited. Existing work often treats explainability as a post-hoc analysis rather than a runtime requirement.

3. Runtime Uncertainty in Microservice Systems

Microservice systems operate under multiple forms of runtime uncertainty that challenge static or reactive adaptation strategies.

3.1 Load Uncertainty

Traffic patterns in real-world systems are rarely predictable. Bursty workloads, flash events, and seasonal variations can invalidate static scaling assumptions. Threshold-based autoscalers may oscillate or over-provision resources due to delayed or noisy metrics [10].

3.2 Failure Uncertainty

Partial failures are inherent in distributed systems [11]. Microservices may experience cascading failures, transient network partitions, or degraded dependencies. Reactive self-healing mechanisms often lack sufficient context to distinguish between isolated faults and systemic issues.

3.3 Behavioral Uncertainty

Emergent behavior arises from complex service interactions. Small configuration changes or deployment updates can have disproportionate runtime effects, which are difficult to anticipate using predefined rules [12].

3.4 Environmental Uncertainty

Cloud environments introduce variability due to multi-tenancy, hardware heterogeneity, and resource contention. These factors can affect performance metrics in ways unrelated to application behavior, complicating adaptation decisions [13].

4. Limitations of Existing Adaptation Approaches

Current adaptation mechanisms primarily focus on *what* action to take rather than *why* it is appropriate.

Rule-based systems encode expert knowledge but are brittle in the face of unforeseen scenarios. Metric-driven autoscalers respond to quantitative signals without understanding qualitative context. Optimization-based approaches may improve efficiency but provide little insight into decision trade-offs.

Crucially, these approaches do not generate reasoning traces or explanations that are intelligible to human operators. As a result, operators struggle to diagnose unexpected behavior, assess system risk, or justify decisions during audits [14].

5. Explainability as a First-Class Runtime Requirement

Explainability in runtime adaptation refers to the system's ability to articulate the rationale behind adaptation decisions in a form understandable by humans.

We identify four explainability requirements for self-adaptive microservice systems:

1. **Traceability** – the ability to link decisions to observed signals and contextual factors.
2. **Interpretability** – explanations must be comprehensible to system operators.
3. **Auditability** – decisions must be reviewable for compliance and accountability.
4. **Confidence Communication** – the system should communicate uncertainty and risk associated with decisions.

Without these capabilities, adaptation remains technically effective but operationally fragile.

6. Conceptual Role of LLMs in Runtime Adaptation

Large language models have demonstrated remarkable capabilities in reasoning over heterogeneous information and generating coherent natural-language explanations [15]. In the context of runtime adaptation, LLMs can be positioned as *reasoning and explanation agents* rather than control executors.

Conceptually, LLMs can:

- Synthesize metrics, logs, traces, and policy constraints into a unified context.
- Reason about possible adaptation strategies and their trade-offs.
- Generate human-readable explanations describing why a particular adaptation is recommended.

Importantly, LLMs should operate under strict guardrails, providing advisory output while deterministic control mechanisms execute final actions. This separation preserves system stability while enhancing transparency.

7. Research Implications and Future Directions

The conceptual framing presented in this paper has several implications. Architecturally, it motivates the integration of reasoning layers alongside traditional control loops. From a human-factors perspective, it enables improved trust and situational awareness. For research, it opens avenues for evaluating explainability and reasoning quality as first-class metrics in self-adaptive systems.

8. Conclusion

This paper examined the challenges of runtime uncertainty in microservice systems and argued that existing self-adaptation mechanisms lack explainability and reasoning transparency. By treating explainability as a first-class requirement and positioning LLMs as reasoning and explanation agents, we establish a conceptual foundation for explainable, intelligence-driven runtime adaptation. This work provides the theoretical grounding necessary for future architectural and empirical investigations into self-adaptive microservice systems.

References

[1] Newman, S. *Building Microservices*. O'Reilly Media, 2015.

[2] Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. "Borg, Omega, and Kubernetes." *ACM Queue*, 2016.

[3] Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. "A Review of Auto-Scaling Techniques for Elastic Applications." *Journal of Grid Computing*, 2014.

[4] ISO/IEC 27001:2013. *Information Security Management Systems*.

[5] Dragoni, N. et al. "Microservices: Yesterday, Today, and Tomorrow." *Present and Ulterior Software Engineering*, Springer, 2017.

[6] Sigelman, B. et al. "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure." *Google Research*, 2010.

[7] Kephart, J., & Chess, D. "The Vision of Autonomic Computing." *IEEE Computer*, 2003.

[8] Salehie, M., & Tahvildari, L. "Self-Adaptive Software: Landscape and Research Challenges." *ACM TAAS*, 2009.

[9] Doshi-Velez, F., & Kim, B. "Towards a Rigorous Science of Interpretable Machine Learning." *arXiv preprint*, 2017.

[10] Gandhi, A. et al. "Autoscale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers." *ACM TOCS*, 2014.

[11] Tanenbaum, A., & Van Steen, M. *Distributed Systems: Principles and Paradigms*. Pearson, 2007.

[12] Jamshidi, P. et al. "Self-Learning Cloud Controllers." *IEEE ICAC*, 2016.

[13] Leitner, P., & Cito, J. "Patterns in the Chaos." *ACM Queue*, 2016.

[14] Chen, Y. et al. "Failure Diagnosis Using Decision Trees." *ICSE*, 2004.

[15] Wei, J. et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." *NeurIPS*, 2022.