ISCTE IUL
REPOSITÓRIO
INSTITUTO UNIVERSITÁRIO DE LISBOA

Repositório ISCTE-IUL

Deposited in *Repositório ISCTE-IUL*:
2026-06-24

Deposited version:
Accepted Version

Peer-review status of attached file:
Peer-reviewed

Citation for published item:
Noetzold, D., Leithardt, V. R. Q. & Barbosa, J. L. V. (2026). Self-adaptive systems in smart environments: Strategies, metrics, and challenges. Computer. 59 (5), 22-32

Further information on publisher's website:
10.1109/MC.2025.3637543

Publisher's copyright statement:
This is the peer reviewed version of the following article: Noetzold, D., Leithardt, V. R. Q. & Barbosa, J. L. V. (2026). Self-adaptive systems in smart environments: Strategies, metrics, and challenges. Computer. 59 (5), 22-32, which has been published in final form at https://dx.doi.org/10.1109/MC.2025.3637543. This article may be used for non-commercial purposes in accordance with the Publisher's Terms and Conditions for self-archiving.

Use policy

Creative Commons CC BY 4.0
The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a link is made to the metadata record in the Repository
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Serviços de Informação e Documentação, Instituto Universitário de Lisboa (ISCTE-IUL)
Av. das Forças Armadas, Edifício II, 1649-026 Lisboa Portugal
Phone: +(351) 217 903 024 | e-mail: administrador.repositorio@iscte-iul.pt
https://repositorio.iscte-iul.pt

Self-Adaptive Systems in Smart Environments: Strategies, Metrics, and Challenges

Darlan Noetzold Valderi Reis Quietinho Leithardt
Jorge Luis Victória Barbosa

Abstract

Self-adaptive systems are increasingly used for effective operation of smart environments, enabling dynamic responses to changing conditions, user needs, and system requirements. This article provides a structured overview of the main strategies for self-adaptation, the most relevant performance metrics, and the key implementation challenges in smart environments. Theoretical foundations such as the MAPE-K loop are presented, together with the role of machine learning and hybrid approaches in adaptation and monitoring. The discussion of performance metrics emphasizes the growing importance of energy efficiency, anomaly detection, and user satisfaction. Practical challenges—including scalability, privacy, interoperability, and real-time processing—are examined through real-world case studies in smart buildings, healthcare, and industrial automation. By synthesizing these perspectives, the article explains the concepts and trends that define the field, offering a foundation for future research and practical deployment of self-adaptive systems in next-generation smart environments.

Smart environments are no longer a futuristic vision—they are becoming part of our everyday reality. Intelligent buildings now optimize energy and comfort, healthcare systems adapt in real time to patient needs, and factories and cities are powered by the IoT. These environments are transforming how we live, work, and interact [1, 2].

But behind this seamless intelligence lies a major challenge: how can these complex, ever-changing systems remain efficient, reliable, and responsive

without constant human oversight? The answer is self-adaptive systems—software and hardware architectures designed to sense context, analyze data, and autonomously adjust behavior to meet evolving goals and conditions [3].

Unlike traditional systems, which depend on static rules and manual reconfiguration, self-adaptive systems operate within a continuous feedback loop. They monitor their own performance, learn from the environment, and adjust decisions dynamically at runtime. This adaptability enables a wide range of applications, from improving energy efficiency in smart buildings to supporting adaptive decision-making in healthcare and other critical domains [4, 5].

This article explores the foundations and frontiers of self-adaptive systems in smart environments. It highlights the main adaptation strategies and the most relevant performance metrics, discusses the latest trends in machine learning and automation, and examines the practical challenges and opportunities that define this field.

1 Theoretical Foundations and Adaptation Strategies

Self-adaptive systems in smart environments are grounded in autonomic computing, managing themselves according to high-level objectives defined by administrators [6]. The MAPE-K (*Monitor*, *Analyze*, *Plan*, *Execute*, *Knowledge*) control loop forms the theoretical foundation for implementing self-adaptation in these systems [7].

Figure 1 presents the conceptual architecture and adaptation flow, highlighting the core MAPE-K phases—*Monitor*, *Analyze*, *Plan*, and *Execute*—all interconnected through a shared *Knowledge* base. In this structure, *Monitor* collects data from the environment and sends it to *Analyze*, which processes the information to detect anomalies or predict future states. Insights from *Analyze* are passed to *Plan*, which generates adaptation strategies based on the analysis and system objectives. These strategies are then executed by *Execute*, which applies the necessary changes to the system or environment. The diagram also depicts a feedback loop from *Analyze* to *Plan*, supporting iterative refinement of adaptation strategies. Colors in the figure indicate system health: red signals an unhealthy environment requiring adaptation, while green represents a healthy, stable state.

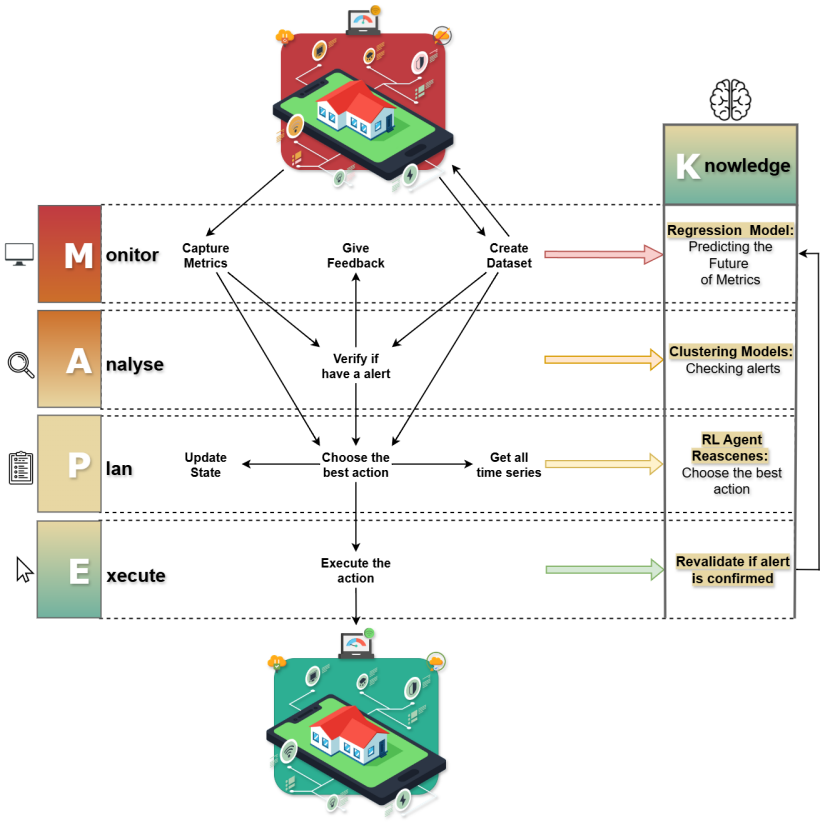


Figure 1: Conceptual architecture and adaptation flow of a self-adaptive system in smart environments, based on the MAPE-K loop.

The *Monitor* phase continuously collects data from sensors, system components, and user interactions, covering environmental conditions, performance metrics, user behavior, and contextual information [8]. The large volume and heterogeneity of this data in smart environments require advanced monitoring infrastructures capable of handling real-time streams from diverse sources [9]. *Analyze* converts raw data into actionable insights using computational techniques. Rule-based analysis applies predefined logic to detect patterns and anomalies, while machine learning uncovers complex relationships and predicts future states [10]. Hybrid approaches combine the strengths of both, offering robust analysis for varied contexts [5].

Plan generates adaptation strategies based on analysis results and sys-

tem objectives. This includes context-aware planning (considering environment, user preferences, and constraints), predictive planning (using historical data and forecasts), and reactive planning (responding to immediate changes) [11]. *Execute* applies planned adaptations through parameter adjustments, reconfiguration, service migration, or resource reallocation, aiming for smooth transitions and minimal disruption. *Knowledge* management captures experiences from adaptation cycles, supporting continuous learning and performance improvement [12].

The *Knowledge* component, central to the MAPE-K loop, aggregates historical data, system models, adaptation policies, and context information. All phases—*Monitor*, *Analyze*, *Plan*, and *Execute*—rely on this shared repository for informed decision-making. For example, *Monitor* uses stored thresholds and context models, *Analyze* leverages historical patterns, *Plan* consults policies and constraints, and *Execute* updates the knowledge base with adaptation outcomes. Continuous updates to this knowledge enable self-adaptive systems to refine behavior, improve accuracy, and respond more effectively to dynamic changes [6].

The *Knowledge* component encompasses a broad spectrum of resources, not limited to machine learning models. It includes static rules, adaptation policies, system and context models, historical logs, and semantic ontologies, in addition to trained ML models and data-driven predictors [5]. Each MAPE-K phase leverages these resources differently: *Monitor* and *Analyze* may use statistical thresholds, context models, and ML-based anomaly detectors; *Plan* can consult both rule-based policies and optimization models, as well as reinforcement learning agents; *Execute* may reference safety constraints and operational procedures [6]. The figure emphasizes machine learning due to its growing importance in modern adaptive systems, but in practice, the *Knowledge* base is a heterogeneous repository that balances data-driven, rule-based, and semantic knowledge to support robust and explainable adaptation [7].

Each MAPE-K loop phase can be implemented using specific technologies. For example, *Monitor* can use distributed event-streaming platforms like Apache Kafka and Flink, or sensor fusion middleware. The *Analyze* phase may employ methods from statistical anomaly detection (Z-score, PCA) to deep learning architectures (LSTM, autoencoders). The *Plan* phase often uses optimization algorithms, genetic algorithms, or reinforcement learning agents. The *Execute* phase can be implemented with container orchestration (e.g., Kubernetes), microservice runtime controllers, or Software-Defined

Networking (SDN) controllers.

To make this mapping actionable, we enumerate common challenges and phase-specific pitfalls. In *Monitor*, issues include sensor drift, calibration errors, missing events in distributed streams, limited edge bandwidth, and privacy constraints; mitigation involves online calibration, robust timestamping, edge buffering, and privacy-preserving collection.

In *Analyze*, challenges include class imbalance, model drift, and latency spikes from complex models. Countermeasures involve incremental/online learning, concept-drift detectors, lightweight feature extraction, and tiered analytics. In *Plan*, practitioners face strategy oscillations, conflicting objectives (e.g., comfort vs. energy), and combinatorial explosion. Solutions use multi-objective optimization, constraint programming, hierarchical policies, and reinforcement learning with safety shields.

In *Execute*, common pitfalls include actuation race conditions, partial failures, and unsafe transient states; these are addressed with transactional actuations, canary or staged rollouts, model predictive control, and runtime guards. The *Knowledge* base must cope with schema evolution, versioning of models and policies, and provenance; practical approaches include typed schemas, policy/model registries, and immutable audit logs.

To expand beyond MAPE-K, we contrasted it with other adaptation frameworks. Table 1 summarizes five representative frameworks: MAPE-K, AWARE, OODA, FOCALE, and Rainbow.

Although MAPE-K is a standardized and widely adopted control-loop architecture, its limitations become clear when compared to alternatives. AWARE offers reflective reasoning and richer contextual awareness, OODA provides lightweight adaptability for constrained IoT devices, FOCALE enhances reasoning with semantic models, and Rainbow delivers practical middleware support for architecture-based adaptation. MAPE-K tends to be more reactive and faces scalability challenges in large-scale, real-time environments. In response, new hybrid and AI-driven frameworks are emerging [9], combining predictive analytics, reinforcement learning, and decentralized consensus to address these shortcomings.

A rank-based quantitative synthesis complements the qualitative comparison in Table 1, following the key performance indicators (KPIs) defined in Table 2. Rainbow and FOCALE achieve the highest overall ranks, reflecting strong reliability, throughput, and reasoning capabilities. MAPE-K follows closely, ranking high in precision and stability. AWARE reaches comparable levels of user satisfaction and adaptability in context-rich domains. OODA

Table 1: Comparison of major adaptation frameworks in smart environments

Framework	Main Phases	Focus	Strengths	Weaknesses/Limitations
MAPE-K[9]	*Monitor*, *Analyze*, *Plan*, *Execute*, *Knowledge*	Iterative control loop for self-management	Structured, widely adopted, good for autonomic computing	Often reactive; scalability issues in large real-time distributed systems
AWARE[13]	Assess, Weigh, Act, Reflect, Enrich	Decision-making with enriched context	Richer context-awareness, reflective reasoning	Less standardized, limited empirical adoption in IoT
OODA[14]	Observe, Decide, Act	Minimalistic decision loop	Lightweight, suitable for resource-constrained devices	Too simple for complex dynamic environments
FOCALE[15]	Observe, Decide, Act + Semantic reasoning	Semantic-based self-configuration in heterogeneous networks	Incorporates ontologies and semantic models; better reasoning	Higher complexity; less practical adoption in large scales
Rainbow[16]	Monitoring, Analysis, Planning, Execution	Runtime adaptation of software architectures	Middleware-oriented; provides tool support and testbeds; widely applied in practice	Complexity; requires architectural modeling

outperforms other frameworks in energy efficiency and response time due to its lightweight decision loop but ranks lower in scalability.

Performance evaluation in self-adaptive systems requires metrics that capture both functional and non-functional aspects [1]. Common metrics include response time (system promptness), energy efficiency (sustainability) [17], and accuracy (correctness of decisions, especially in ML-enabled adaptations) [5]. Throughput and resource utilization measure the system's capacity and efficiency, while user satisfaction and reliability ensure the system meets expectations and maintains dependable operation [9].

Table 2 shows that a comprehensive evaluation framework should consider a balanced set of metrics, ensuring that both system-level and user-level objectives are addressed. The information presented in this table was collected from the systematic review by Aguilar et al. [18]. Figure 2 summarizes the distribution of the most monitored metric groups in self-adaptive smart environments, based on a systematic literature review [3, 6, 19].

The chart emphasizes that energy efficiency and anomalies are the most frequently monitored groups, reflecting the growing importance of sustainability and system robustness. Latency and response time also receive significant attention, especially in applications where timely adaptation is critical. The distribution in Figure 2 is based on a synthesis of systematic literature

Table 2: Key Performance Metrics for Self-Adaptive Systems in Smart Environments

Metric Group	What It Measures	Examples / Indicators
Response Time	Speed of system reaction to events or changes	Latency, time to adapt, time to recover
Energy Efficiency	Power consumption and sustainability	Energy usage, battery life, power savings
Accuracy	Correctness of decisions and adaptations	Prediction/classification accuracy, error rate
Throughput	Capacity to process data or tasks concurrently	Requests per second, data rate, task completion rate
Resource Utilization	Efficiency in using computational, storage, and network resources	CPU/memory usage, bandwidth utilization
User Satisfaction	Perceived quality and usability by end-users	User ratings, feedback, task success rate
Reliability	System availability and fault tolerance	Uptime, mean time between failures, recovery time

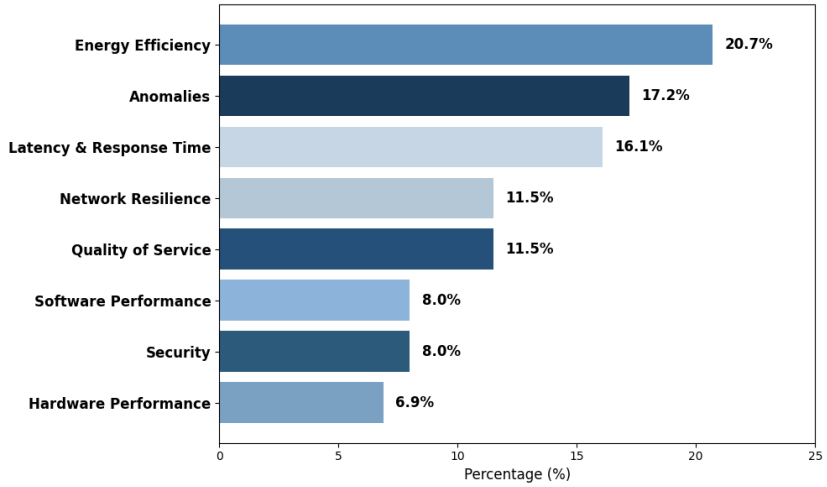


Figure 2: Distribution of performance metrics and evaluation frameworks in self-adaptive systems, synthesized from systematic literature reviews and key studies.

reviews [1, 2].

Recent advances in artificial intelligence have expanded the adaptability of self-adaptive systems in smart environments. Deep reinforcement learn-

ing (DRL) enables systems to learn optimal adaptation strategies through trial and error in complex, high-dimensional spaces [19]. DRL agents can autonomously discover adaptation policies that generalize across varying contexts, as demonstrated in energy management for smart buildings and adaptive network routing [20].

Transfer learning further improves adaptability by allowing models trained in one domain or environment to be efficiently adapted to new related contexts with minimal retraining [5]. Multi-agent systems (MAS) introduce distributed intelligence, where multiple autonomous agents collaborate or compete to achieve system-wide adaptation goals [21]. MAS approaches are especially effective in large-scale, heterogeneous environments, such as smart cities or industrial IoT, where centralized control is impractical [22].

These advanced AI techniques have been validated in case studies. DRL-based controllers have achieved up to 20% energy savings in smart buildings while maintaining occupant comfort [18]. Transfer learning has enabled rapid deployment of predictive maintenance models across different industrial sites [1]. Multi-agent coordination has demonstrated improved throughput and resilience in smart grid and logistics applications [22].

2 Adaptability Challenges, Metrics, and Benchmarks

Achieving true adaptability in smart environments remains a complex challenge. In practice, adaptation must be fast to prevent losses or service degradation, accurate to resolve issues, lightweight to avoid excessive overhead, and stable to prevent oscillations or new failures.

These requirements highlight the need for adaptation-specific metrics that go beyond traditional performance indicators. Mean Adaptation Time (MAT) measures how quickly the system responds to anomalies or environmental changes. Adaptation Accuracy (AA) reflects the proportion of successful adaptations. Adaptation Overhead (AO) quantifies the additional resource consumption caused by adaptation. Finally, Adaptation Stability (AS) assesses whether the system remains stable after adapting [19].

Table 3 summarizes these key adaptation metrics, presenting their formulas and practical interpretations. The information was synthesized from Parunak and Brueckner's work [21].

Table 3: Adaptation Metrics for Self-Adaptive Systems

Metric	**Formula**	**Interpretation**
Mean Adaptation Time (MAT)	$MAT = \frac{1}{n}\sum_{i=1}^{n}(t_{\text{end},i} - t_{\text{detect},i})$	Measures the average time between anomaly detection and completion of adaptation. Lower MAT means the system reacts quickly, minimizing losses and service disruption.
Adaptation Accuracy (AA)	$AA = \frac{\text{Successful adaptations}}{\text{Total adaptations}} \times 100$	Indicates the proportion of adaptation actions that successfully resolve the detected issue. High AA reflects the system's ability to make correct and effective decisions.
Adaptation Overhead (AO)	$AO = \frac{R_{\text{after}} - R_{\text{before}}}{R_{\text{before}}} \times 100$	Quantifies the additional resource consumption (CPU, memory, bandwidth) caused by adaptation. Lower AO is desirable to avoid degrading overall system performance.
Adaptation Stability (AS)	$AS = \frac{1}{\sigma^2_{\text{post-adaptation}}}$	Assesses how stable the system remains after adaptation. High AS means the system avoids oscillations or performance drops, maintaining reliable operation.

A variety of benchmarks and testbeds have been developed to make adaptation metrics actionable and comparable across solutions. For example, the Rainbow framework [16] is widely used for architecture-based self-adaptation, offering a modular testbed for fault injection and adaptation measurement. DYNAMICO supports controlled testing of dynamic adaptation in distributed systems [12], while IoT platforms like FIWARE and Eclipse Hono provide realistic data streams and device emulation [11].

Recent studies using platforms such as Rainbow and FIWARE have reported quantitative results. In a Rainbow-based smart building testbed, deep reinforcement learning (DRL) achieved a mean adaptation time (MAT) of 2.3 seconds, adaptation accuracy (AA) of 94.5%, adaptation overhead (AO) of

7.2% CPU, and adaptation stability (AS) with post-adaptation KPI variance below 3% [23]. Comparative experiments on FIWARE show that hierarchical MAPE-K strategies consistently outperform flat architectures, especially as system scale increases [2].

3 Implementations and Solutions

This section addresses key technical challenges and practical solutions for implementing self-adaptive systems in smart environments, covering scalability, privacy/security, interoperability, real-time processing, and validation.

Scalability is a major concern in large smart environments due to growing device numbers and data volumes [23]. Solutions include distributed monitoring, hierarchical adaptation, and edge/cloud partitioning [9]. Privacy and security are critical, requiring techniques like differential privacy, federated learning, and secure multi-party computation [19]. Interoperability challenges arise from heterogeneous components [2]. Frameworks such as FIWARE, OpenHAB, and Eclipse Hono use semantic middleware and standardized APIs to integrate diverse ecosystems.

Real-time adaptation is important for many smart environment applications, demanding rapid responses under strict latency constraints [10]. This leads to adopting lightweight anomaly detection (e.g., moving averages, fast clustering) for safety-critical loops, complemented by more complex asynchronous analysis. Trade-offs between accuracy and latency are managed with approximation algorithms, event prioritization, and model compression techniques [5].

Validation and verification of self-adaptive systems are challenging due to their dynamic nature [19]. Emerging approaches include model checking, simulation-based testing, and runtime monitors. Figure 3 summarizes the main groups of techniques identified in recent literature.

As shown in Figure 3, traditional machine learning and deep learning techniques are the most widely adopted. Neural networks and reinforcement learning are also prominent, especially in scenarios requiring advanced pattern recognition and autonomous decision-making. The distribution is synthesized from systematic literature reviews [5, 10].

Most implementations operate at scales of hundreds to thousands of nodes, facing challenges such as communication overhead, synchronization of distributed monitoring, and prevention of negative feedback loops. Hier-

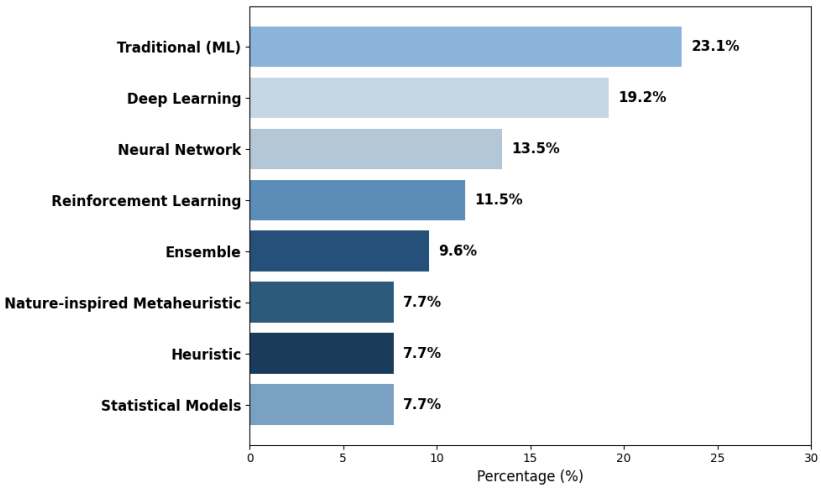


Figure 3: Distribution of monitoring and self-adaptation techniques in smart environments, synthesized from systematic literature reviews and key studies.

archical MAPE-K layers and edge/cloud partitioning are commonly used to address these problems. Techniques to prevent negative feedback and oscillations include hysteresis bands, minimum dwell times, rate limiting, backoff strategies, priority queues, conflict resolution, and leader–follower coordination among distributed planners.

4 Case Studies and Practical Applications

Smart building systems are a prime example of successful self-adaptive technology deployment [18]. By integrating HVAC, lighting, security, and energy management, these systems use adaptive algorithms to respond to occupancy, weather, and energy costs, achieving significant energy savings and improved comfort through continuous adaptation based on sensor data and user feedback.

Healthcare monitoring systems further illustrate the value of self-adaptation in life-critical contexts. These solutions adapt to changing patient conditions, equipment availability, and clinical protocols, maintaining high reliability and accuracy. Notable implementations include adaptive alarms that reduce false positives, personalized monitoring protocols, and resource allocation systems that optimize staff and equipment use [20].

In industrial automation, self-adaptive systems optimize production, maintain equipment, and ensure safety by adjusting to production demands, equipment performance, and supply chain disruptions through intelligent scheduling, predictive maintenance, and adaptive quality control [23]. Transportation systems apply self-adaptation to traffic management, route optimization, and autonomous vehicle coordination [3].

A smart building case study demonstrates an explicit MAPE-K design. The *Monitor* phase collects environmental and occupancy data through edge gateways. The *Analyze* phase combines fast anomaly detection with regression-based energy forecasts. The *Plan* phase solves constrained optimization problems to balance energy consumption and comfort. Finally, the *Execute* phase applies setpoint changes and ventilation cycles using staged rollouts and minimum dwell times to prevent oscillations [17, 18].

Beyond this application, heterogeneous AI techniques are integrated into the MAPE-K workflow. Predictive models for time-series forecasting and anomaly anticipation provide early warnings about potential inefficiencies or failures, granting reinforcement learning agents in the *Plan* phase additional lead time to evaluate context metrics across temporal streams. This predictive–reactive synergy lowers the computational pressure of time-critical decisions and increases resilience [6].

In the healthcare mini-case, *Monitor* streams vital signs and room context to an edge collector with privacy-aware ingestion; *Analyze* runs change-point detection and an early-warning score updated each minute; *Plan* uses a two-tier strategy with a rule-based safety layer for immediate alerts and a predictive layer for scheduling non-urgent interventions; *Execute* coordinates alarms and device configuration with minimum dwell times and priority queues to prevent alert fatigue and control races [10].

This methodology is transferable to industrial automation by mapping *Monitor* to production sensors and quality metrics, *Analyze* to anomaly detection and yield forecasts, *Plan* to scheduling and setpoint optimization under throughput and safety constraints, and *Execute* to robotic or PLC actuation with staged rollouts and safety interlocks [21].

5 Conclusion

Self-adaptive systems represent a fundamental advancement in smart environment technology, enabling dynamic optimization and autonomous man-

agement of complex, interconnected systems. The evolution from rule-based to machine learning-enabled adaptation strategies reflects the increasing sophistication of smart environments and the growing availability of data and computational resources. Hybrid approaches that combine multiple adaptation techniques offer the most promising path forward, providing the flexibility and robustness required for real-world deployments.

Performance evaluation must encompass multiple dimensions, from traditional metrics such as response time and throughput to emerging concerns such as energy efficiency and user satisfaction. Implementation challenges are being addressed through advancing technologies and methodologies. Scalability, privacy, interoperability, and real-time processing requirements drive continued innovation in system architectures, algorithms, and deployment strategies.

The future of self-adaptive systems is closely tied to advances in artificial intelligence, edge computing, and new computational paradigms. As AI integration progresses, sophisticated learning algorithms—including deep reinforcement learning, transfer learning, and multi-agent systems—are enabling increasingly intelligent and autonomous adaptation behaviors. The adoption of edge computing is making these systems more responsive and privacy-preserving by processing data closer to its source [24]. Looking further ahead, quantum computing may address complex optimization and learning challenges that are currently beyond the reach of classical systems. At the same time, the growing deployment of self-adaptive systems highlights the need for explainable AI, ensuring that adaptation decisions are transparent and understandable to users and administrators.

Acknowledgments

The authors wish to acknowledge that this work was supported by CNPq (National Council for Scientific and Technological Development - grant number 307137/2022-8), and CAPES (Coordination for the Improvement of Higher Education Personnel - Brazil - Finance Code 001). We are also grateful to Unisinos (University of Vale do Rio dos Sinos) and ISCTE-IUL (University Institute of Lisbon) for undertaking this research. This research was partially financed by national funds through FCT - Foundation for Science and Technology, I.P. under projects UIDB/04466/2025 and UIDP/04466/2025, and project 16881, LISBOA2030-FEDER-00816400.

References

[1] K. Patel, C. Mistry, R. Gupta, S. Tanwar, and N. Kumar, "A systematic review on performance evaluation metric selection method for IoT-based applications," *Microprocessors and Microsystems*, vol. 101, p. 104894, 2023, doi: 10.1016/j.micpro.2023.104894.

[2] A. Souri, A. Hussien, M. Hoseyninezhad, and M. Norouzi, "A systematic review of IoT communication strategies for an efficient smart environment," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, p. e3736, 2022, doi: 10.1002/ett.3736.

[3] S. E. Sorour, M. Aljaafari, A. M. Shaker, and A. E. Amin, "LSTM-JSO framework for privacy preserving adaptive intrusion detection in federated IoT networks," *Scientific Reports*, vol. 15, no. 1, p. 11321, 2025, doi: 10.1038/s41598-025-95966-z.

[4] R. Calinescu, R. Mirandola, D. Perez-Palacin, and D. Weyns, "Understanding Uncertainty in Self-adaptive Systems," in *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pp. 242–251, 2020, doi: 10.1145/3487921.

[5] O. Gheibi, D. Weyns, and F. Quin, "Applying machine learning in self-adaptive systems: A systematic literature review," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 15, no. 3, pp. 1–37, 2021, doi: 10.1145/3469440.

[6] S. Jamshidi, A. Amirnia, A. Nikanjam, K. W. Nafi, F. Khomh, and S. Keivanpour, "Self-adaptive cyber defense for sustainable IoT: A DRL-based IDS optimizing security and energy efficiency," *Journal of Network and Computer Applications*, vol. 239, p. 104176, 2025, doi: 10.1016/j.jnca.2025.104176.

[7] M. Elgendi, "MAPE-K: A reference model for self-adaptive systems," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 1–6, 2019, isbn: 978-91-87427-85-5.

[8] G. Calderon, G. del Campo, E. Saavedra, and A. Santamaría, "Monitoring framework for the performance evaluation of an IoT platform with Elasticsearch and Apache Kafka," *Information Systems Frontiers*, pp. 1–17, 2023, doi: 10.1007/s10796-023-10409-2.

[9] A. Khan et al., "Fault-tolerance and unique identification of vertices and edges in a graph: The fault-tolerant mixed metric dimension," *Journal of Parallel and Distributed Computing*, vol. 197, p. 105024, 2025, doi: 10.1016/j.jpdc.2024.105024.

[10] T. R. D. Saputri and S.-W. Lee, "The application of machine learning in self-adaptive systems: A systematic literature review," *IEEE Access*, vol. 8, pp. 205948–205967, 2020, doi: 10.1109/ACCESS.2020.3036037.

[11] S. S. Selvarajan et al., "IoT driven healthcare monitoring with evolutionary optimization and game theory," *Scientific Reports*, vol. 15, no. 1, p. 15224, 2025, doi: 10.1038/s41598-025-99129-y.

[12] A. Bombarda, G. Ruscica, and P. Scandurra, "A self-managing IoT-Edge-Cloud architecture for improved robustness in environmental monitoring," in *Proc. 40th ACM/SIGAPP Symp. on Applied Computing (SAC '25)*, Catania, Italy, pp. 1738–1745, 2025, doi: 10.1145/3672608.3707801.

[13] G. M. Machado, V. Maran, G. M. Lunardi, and others, "AwARE: A framework for adaptive recommendation of educational resources," *Computing*, vol. 103, pp. 675–705, 2021, doi: 10.1007/s00607-021-00903-3.

[14] R. Shawe and I. McAndrew, "OODA and CECA: Analysis of decision-making frameworks," *American Journal of Industrial and Business Management*, vol. 13, pp. 457–464, 2023, doi: 10.4236/ajibm.2023.136029.

[15] J. Strassner, S. van der Meer, and J. Hong, "The applicability of self-awareness for network management operations," *Lecture Notes in Computer Science*, vol. 5844, pp. 15–28, 2009, doi: 10.1007/978-3-642-05006-0_2.

[16] S.-W. Cheng, D. Garlan, and B. Schmerl, "Evaluating the effectiveness of the Rainbow self-adaptive system," in *Proc. 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, Vancouver, BC, Canada, 2009, pp. 132–141, doi: 10.1109/SEAMS.2009.5069082.

[17] W. B. Qaim et al., "Towards energy efficiency in the internet of wearable things: A systematic review," *IEEE Access*, vol. 8, pp. 175412–175435, 2020, doi: 10.1109/ACCESS.2020.3025270.

[18] J. Aguilar, A. Garces-Jimenez, M. D. R-moreno, and R. García, "A systematic literature review on the use of artificial intelligence in energy self-management in smart buildings," *Renewable and Sustainable Energy Reviews*, vol. 151, p. 111530, 2021, doi: 10.1016/j.rser.2021.111530.

[19] S. Weerasinghe et al., "Reinforcement Learning Based Approaches to Adaptive Context Caching in Distributed Context Management Systems," *ACM Transactions on Internet of Things*, vol. 5, no. 2, 2024, doi: 10.1145/3648571.

[20] Imran, N. Iqbal, and D. H. Kim, "IoT Task Management Mechanism Based on Predictive Optimization for Efficient Energy Consumption in Smart Residential Buildings," *Energy and Buildings*, vol. 257, p. 111762, 2022, doi: 10.1016/j.enbuild.2021.111762.

[21] L. Garcia, H. Samin, and N. Bencomo, "Decision Making for Self-Adaptation Based on Partially Observable Satisfaction of Non-Functional Requirements," *ACM Trans. Auton. Adapt. Syst.*, vol. 19, no. 2, Art. no. 11, pp. 1–44, Apr. 2024, doi: 10.1145/3643889.

[22] X. Li et al., "Research on key technologies of high energy efficiency and low power consumption of new data acquisition equipment of power Internet of Things based on artificial intelligence," *International Journal of Thermofluids*, vol. 21, p. 100575, 2024, doi: 10.1016/j.ijft.2024.100575.

[23] A. Matathammal et al., "EdgeMLBalancer: A Self-Adaptive Approach for Dynamic Model Switching on Resource-Constrained Edge Devices," in *Proc. 2025 IEEE 22nd Int. Conf. on Software Architecture Companion (ICSA-C)*, pp. 543–552, 2025, doi: 10.1109/ICSA-C65153.2025.00081.

[24] K. Vaidhyanathan, M. Caporuscio, S. Florio, and H. Muccini, "ML-enabled Service Discovery for Microservice Architecture: a QoS Approach," in *Proc. 39th ACM/SIGAPP Symp. on Applied Computing (SAC '24)*, Avila, Spain, pp. 1193–1200, 2024, doi: 10.1145/3605098.3635942.