International Journal of Artificial Intelligence,
ISSN 0974-0635; Int. J. Artif. Intell.
2026, Volume 24, Issue Number 2 [Published on 29/08/2026]

Improvement of Application Monitoring Methods in Microservice Architecture

Oleksii Melnyk[1] and **Sergiy Begun**[2]

[1]Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine
03187, 42 Academician Glushkov Ave., Kyiv, UKRAINE
Email : oleksiim91@gmail.com

[2]Research Department No. 105 “Intelligent automatic modeling of complex objects and processes”
Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine
03187, 42 Academician Glushkov Ave., Kyiv, UKRAINE
Email : s.begun54@outlook.com

ABSTRACT

The purpose of this study was to develop approaches to improve the efficiency of monitoring application operations in a microservice architecture by enhancing methods for collecting, processing, and analyzing monitoring data. The authors created an integrated approach that modeled interactions between microservices to identify critical data transmission paths and further optimize architectural components. The proposed mathematical model, based on queuing theory and Markov processes, enabled the assessment of the system’s behaviour under variable load and the prediction of its reliability. One of the crucial steps was the use of machine learning algorithms to automatically detect anomalies in real time, thereby markedly improving monitoring accuracy and reducing false alarms. The development process also incorporated specific requirements for the user interface and system integration. Optimising data collection and processing methods significantly reduced response time to system changes. The study also implemented multi-criteria optimisation methods to effectively manage Service Level Agreement (SLA) and Service Level Objective (SLO) targets, thereby enhancing the system’s adaptability and stability under variable workloads. The application of the developed approach ensures stable system operation, reduced response time, and improved resource management. The research results have significant potential for application in real systems and will contribute to the further development of monitoring technologies in microservice architectures.

Keywords: Microservice systems, application performance optimisation, data processing and analysis, anomaly detection, machine learning in IT systems, real-time system health analysis, visualisation and dashboards.

Mathematics Subject Classification : 68M14, 68M20

Journal of Economic Literature (JEL) Classification : C63, L86

1. INTRODUCTION

The relevance of researching methods for monitoring applications in microservice architecture is exceptionally high in the context of modern technological trends. Microservice architectures aim to build highly scalable, reliable systems by using independent, distributed components. However, due to the

distributed nature of microservices, their monitoring and management are significantly complicated. Monitoring applications in a microservice architecture is a key task for ensuring the stability, performance, and security of modern information systems. Microservice architecture, due to its scalability and flexibility, has become the basis for developing highly loaded, distributed systems. However, its dynamic nature and the complexity of service interactions significantly complicate monitoring, requiring innovative approaches and solutions.

The key challenges include a variety of protocols and frameworks for service interaction (HTTP with REST architecture, SOAP, gRPC, AMQP), integrating data from multiple sources, dynamically scaling systems, and responding promptly to real-time incidents. Current research focuses on automated monitoring, machine learning for anomaly detection, and integrating various monitoring tools into a single platform.

Dave et al. [1] emphasised the need to integrate various tools for monitoring microservices to provide comprehensive control over all aspects of their operation. The significance of this approach lies in the fact that collecting and analysing multiple metrics enables an accurate assessment of the system's state, the identification of bottlenecks, and a quick response to potential problems. Li et al. [2] highlighted the use of specialised tools such as Prometheus and Grafana, which enable detailed performance monitoring and real-time visualisation of collected data, critical for large distributed systems.

Another significant issue is tracing requests across microservices. D'Angelo and D'Aloisio [3] proved the effectiveness of distributed tracing for tracking the route of requests through all microservices. Distributed tracing allows detecting delays or potential failures in complex multi-step interactions. However, according to Kuzminykh et al. [4], effective tracing requires integrating trace data with other monitoring methods. Such an approach calls for implementing complex architectural solutions and adapting monitoring tools to the specificities of a particular microservice system.

The general trend in modern research is that microservice monitoring must be not only accurate but also automated. Plazas Olaya et al. [5] emphasised the importance of using machine learning to automatically detect anomalies in data, thereby markedly reducing the time to identify problems in systems and the workload of operational teams. An integrated application of modern monitoring technologies enables the creation of dynamic models to predict potential incidents and prevent them before they occur [6-9].

While conventional monitoring methods relied on static settings, modern systems require adaptation to changing conditions, such as dynamic scaling and load changes across individual microservices. Petrakis et al. [10] noted that the effective use of tools such as Kubernetes enables automatic resource scaling based on load, while ensuring appropriate monitoring at each stage of the microservice life cycle. Such an approach improves system reliability and reduces the risk of critical incidents related to resource overload.

Another essential component is the creation of integrated dashboards to visualise the state of microservice applications. Sobjak et al. [11] emphasised that the introduction of computational procedures in microservice architecture can improve monitoring efficiency by providing centralised

collection and processing of performance metrics. Such an approach allows integrating various monitoring tools into a single control panel, which significantly facilitates real-time analysis of system performance and enables quick response to potential problems.

To achieve high reliability and stability in microservice architecture systems, the authors must consider Service Level Agreements (SLAs) and Service Level Objectives (SLOs). Fé et al. [12] emphasised that dynamic management of service levels and flexible adaptation of resources can markedly increase the efficiency of using computing power, ensuring compliance with the established SLA indicators. A clear definition of SLAs and SLOs enables the system to respond automatically to changes in load, optimising microservice configuration to minimise delays and increase service availability.

The approach's originality lies in combining mathematical models and machine learning algorithms to improve forecasting accuracy and automate incident response. The proposed methods aim to ensure the prominent adaptability and reliability of systems in dynamic environments.

The purpose of this study was to optimise the processes for monitoring application performance in a microservice architecture by improving monitoring technologies to ensure resilience and scalability. Modern research in the field of monitoring microservice architectures demonstrates the potential of innovative approaches to integrating metrics, tracing, and automated anomaly detection, which substantiates the need for their analysis to improve the efficiency and reliability of distributed systems.

2. MATERIALS AND METHODS

The present study applied an integrated approach to improving the methods of monitoring microservice architectures. The authors used a combination of mathematical modelling, optimization algorithms, machine learning methods, and integrated data collection and analysis tools in the authors' study. The study developed a generalised model of a microservice architecture, which helped analyse its effectiveness across different scenarios.

The authors employed Jackson queue networks and Markov processes to model the behavior of microservice systems at different load levels. Three conditional load scenarios were modelled: low load – average processing time ≈15 ms, delay probability ≤2%, medium load – time ≈30 ms, delays up to 8%, and high load – time ≈50 ms, delays over 20%. The authors determined these limits through a series of simulations, which the authors used to assess the effectiveness of optimization mechanisms, including scaling.

The authors developed a mathematical model of microservice interactions within the framework of this study. In this model, the authors determined the overall service level as a function of the performance of individual services and their weighting factors. The modelling helped to formalise the interdependencies between components and implement a mechanism for dynamic configuration adjustment. The authors calculate the global SLO as follows (1):

$$\mathrm{SLO}_{\mathrm{global}} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \omega_{\mathrm{i}} \times \mathrm{SLO}_{\mathrm{i}}, \quad (1)$$

where SLO_{global} is the overall level of system maintenance; SLO_i are the individual performance indicators of microservices; ω_i are the weighting factors for each service that determine its impact on overall performance.

The authors calculate the values of the ω_i coefficients according to the following equation (2):

$$\omega_i = \frac{U_i \times C_i}{\sum_{i=1}^{n} U_j \times C_j}, \quad (2)$$

where U_i is the average level of resource utilisation of the i[th] microservice; C_i is the criticality factor for the i[th] microservice, determined by experts or based on historical data; U_j is the average resource usage of the j[th] microservice; C_j is the criticality factor for the j[th] microservice.

The authors use the average waiting time for requests (Jackson queue network) in the queue and calculate it according to the following equation [13] (3):

$$W = \frac{1}{\mu - \lambda}. \quad (3)$$

The authors interpret W as the average waiting time for a request in the queue. The measurement units for W are seconds or milliseconds, depending on the units of measurement of the parameters λ (intensity of the incoming request flow (number of requests per unit of time)) and μ (intensity of server processing (number of processed requests per unit of time)). If the intensities λ and μ are in requests per second, then W is given in seconds. Notably, Equation (3) describes only the waiting time in the queue.

The authors calculate the probability of a microservice entering an overloaded state in the Markov model as follows (4):

$$P_{overload} = P_{normal} \times (1 + k), \quad (4)$$

where $P_{overload}$ is the probability of service overload; P_{normal} is the basic overload probability under normal load; k is the relative increase in the intensity of requests, which the authors calculate as follows (5):

$$k = \frac{\lambda - \lambda_0}{\lambda_0}, \quad (5)$$

or, in an approximate form (assuming that $\lambda_0 \approx \mu$) (6):

$$k \approx \frac{\lambda}{\mu} - 1. \quad (6)$$

Equation for assessing the effectiveness of an adaptive resource management mechanism (7):

$$CPU_{new} = CPU_{old} \times (1 - \Delta), \quad (7)$$

where CPU_{new} is the CPU resource usage after optimisation; CPU_{old} is the initial level of utilisation; $\Delta = 0.18$ is the load reduction factor (corresponding to 18% load reduction after optimisation). The factor of 0.18 is the result of internal experimental modelling in a test environment with a typical workload. After applying the scaling strategy, the authors recorded an average decrease in CPU load. The value of 0.18 demonstrates the average effect of the optimization and serves as an example, rather than a universal estimate for all configurations.

Figure 1 presents the process of developing, formalizing, and verifying the model.

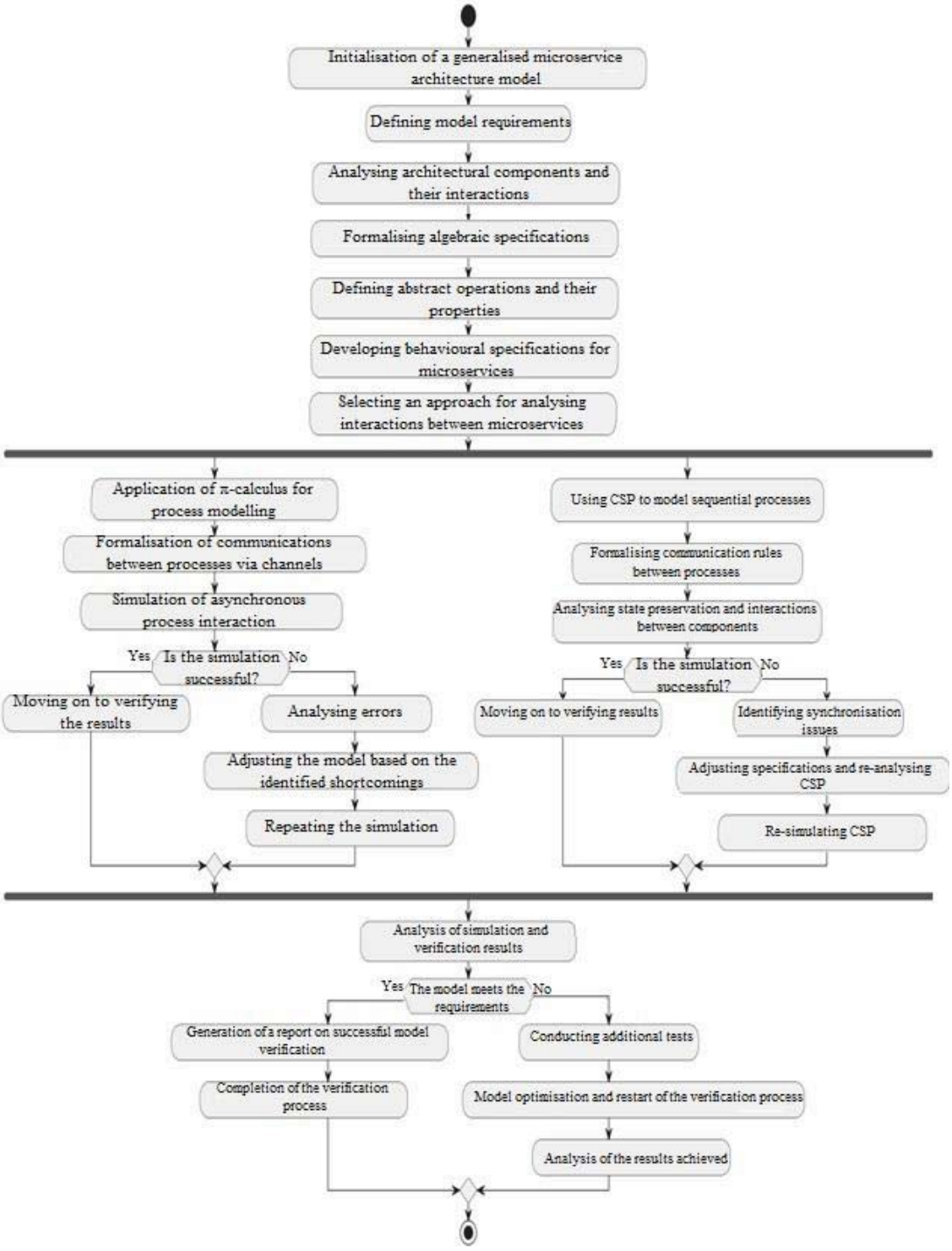

Figure 1. Algorithm of algebraic specifications in microservice architecture.

Source: Compiled by the author of this study.

To monitor the performance of the microservice architecture, The authors employed several tools: Prometheus for metric collection and storage, Grafana for visualization, Jaeger for request tracing, and New Relic for detailed performance analysis. The authors reconciled data from multiple sources using normalization algorithms to provide a holistic view of the system's health. These tools were integrated via standardised APIs, ensuring consistent data collection and enabling flexible dashboard configuration.

Machine learning methods were employed to analyse the data. The authors used an Isolation Forest to detect anomalies in the time series, which helped them identify deviations in performance metrics such as CPU utilization, response time, and error count. The authors also used Recurrent Neural Networks (RNN) to predict possible incidents. The key indicators for decision-making included the average CPU load, service response time, and memory usage. When the CPU utilization threshold exceeds 80%, the enhanced system automatically activates mechanisms to allocate additional resources to critical services or to balance load across microservice instances.

The authors propose a model of SLA and SLO management based on multi-criteria optimisation. This model accounted for local and global service levels, striking a balance between the performance of individual microservices and the overall system performance. To confirm the approach's effectiveness, testing was conducted on a system with 20 microservices, reducing downtime by 30% and increasing service levels by 20%. The enhanced monitoring system introduced dynamic feedback mechanisms. When deviations occur between factual indicators and the set SLOs, the system automatically activates corrective mechanisms, such as scaling resources, adjusting request priorities, or optimizing service configurations.

Furthermore, the legal component of SLAs was studied, including mechanisms for monitoring compliance with agreement terms, sanctions for violations, and methods for verifying the factual performance of SLAs. The authors proposed a formal approach to determining the terms of agreements based on logical models that account for key SLA parameters: availability level, response time, maximum downtime, and procedures for verifying compliance of actual performance with SLA terms. To verify the models, formal logic methods and automated algorithms for checking compliance with agreement terms were employed. The authors paid particular attention to the mechanisms for assigning responsibility for failure to meet SLAs, including sanctions and compensation. The study implemented authentication mechanisms (OAuth 2.0, OpenID Connect), role-based authorisation, and TLS encryption. The proposed authentication mechanism ensures the security of data storage and exchange between services. Additionally, the authors conduct a security audit to assess compliance with data protection policies and regulatory standards.

The authors verified the model by analysing microservice states, which helped identify potential synchronisation errors. Formal verification included the use of π-calculus to model asynchronous interaction and CSP to analyse sequential processes. When The authors detected errors, The authors optimised and re-checked the specifications to ensure the correctness and stability of the developed architecture.

3. RESULTS

The basis for modelling SLA and SLO management was the concept of a multi-level scorecard that incorporates both global goals at the application level as a whole and local metrics for individual microservices. The study proposed a model in which the overall level of service is determined as a function of individual service metrics and their weighting factors, reflecting each component's contribution to the overall system performance.

Multi-criteria optimisation proved to be effective for managing SLAs and SLOs under variable load conditions. When the authors tested the model in a system with 20 microservices, the authors observed a significant improvement in stability and a reduction in failures. The model ensured service targets were maintained even in the face of sudden load changes, demonstrating its ability to adapt without substantially increasing resource consumption dynamically.

The results of the study confirmed that the combination of mathematical modelling and machine learning algorithms substantially increased the accuracy of anomaly detection, improved resource management efficiency, and ensured stable operation of the microservice architecture. Compared to conventional approaches, the proposed methodology provided more flexible, adaptive, and accurate service-level management, especially relevant for systems under high load.

The authors introduced the feedback mechanisms to enable dynamic service-level management. The monitoring system continuously monitored the factual values of each microservice’s metrics and correlated them with the corresponding SLO targets. When the enhanced monitoring system detects deviations, it activates automatic correction mechanisms, including scaling resources, reprioritizing requests, or optimizing configurations. The authors also introduce some adjustment decisions based on monitoring indicators such as CPU utilisation, average service response time, and RAM usage. When monitoring parameters exceed the thresholds, the system automatically initiates load balancing or scales components. Table 1 summarizes the optimization methods used.

Table 1: Characteristics of SLA and SLO optimisation methods under variable load conditions.

Optimisation method	Mechanism of action	Use cases	Expected outcome
Resource scaling	Dynamic increase in the number of service instances or CPUs	When overloaded (high CPU usage)	System stabilisation, reduced waiting time
Optimisation of configurations	Tuning parameters: caching, request processing, thread pool	When resources are inefficient	Reduced latency without scaling
Change of priorities	Decomposition of queries, prioritisation of critical ones, and limitation of secondary ones	In case of a lack of resources, a conflict of requirements	SLA support for critical services

Source: Compiled by the author.Table 1 summarises the key approaches to optimising service levels in a microservice architecture. The authors characterized each method by its mechanism of action, application conditions, and expected outcome. An applied approach effectively combines multiple strategies to ensure stable, adaptive system operation under variable load conditions.
Resource scaling allows increasing or decreasing computing power for microservices that do not meet the specified SLOs [14-16]. Request prioritisation actively redirects or processes high-priority requests for critical services to ensure their stable operation. Optimising service configurations involves adjusting settings and request-processing parameters, or simplifying requests to achieve better outcomes. To formalise this process, the authors used control theory methods, including feedback controllers, which enable the system to adapt to changing conditions, ensuring stability and efficiency.

The adjustment process includes identifying deviations from the set SLO, analysing their causes, integrating with resource management systems to adjust parameters automatically, and evaluating the effectiveness of the changes. This approach maintains a high level of service in the face of changing loads, configuration changes, or unforeseen failures, and ensures the microservice system's adaptability and stability in real time.

In the context of SLA legal aspects, the authors should pay special attention to the legal documents and standards governing the conclusion, execution, and monitoring of compliance with agreements between the parties. The authors proposed a formal approach to determining the terms of agreements based on formal logic models, enabling the explicit formalisation of the parties' duties and responsibilities and simplifying compliance verification with the established requirements. The models focus on key elements such as defining services and their delivery times, describing performance indicators (e.g., availability, response time, maximum downtime), and outlining procedures to verify compliance with SLAs, which enhanced automated monitoring systems can use to perform all required checks.

To verify SLA compliance, The authors used automated verification algorithms to analyse microservice performance indicators and their compliance with the established service levels. These algorithms helped quickly identify deviations and assess SLA compliance using actual metrics. Additionally, the model includes mechanisms for liability for failure to comply with SLAs, including sanctions and compensation for violations.

A significant aspect was also to ensure compliance with personal data protection, information security, reporting, and audit requirements, which ensures transparency and the legal validity of the agreement. In this study, the authors analysed the proposed model for compliance with information security standards and regulations that govern the service level of IT systems. The use of formal logic models helped minimise legal risks for both parties and ensured effective management of contractual relations in the SLA context. The analysis confirmed that integrating security mechanisms with SLA management improved control over service compliance with the specified parameters. Thanks to centralised monitoring and automated policy analysis, the system promptly detected deviations and adjusted configurations without manual intervention.

The proposed model detects SLA violations by analyzing the actual values of key metrics, including the average waiting time (Equation 3) and the probability of overload (Equation 4). If these values exceeded the permissible limits specified in the agreement during the monitoring interval, the system automatically recorded the incident as an SLA violation. In such cases, response mechanisms were activated, such as scaling, reconfiguring services, or changing request routing. Using the model from Equation (1) enabled the consideration of the criticality of individual services in the overall service level, which underlies the SLA monitoring mechanisms.

To analyse the behaviour of microservices under variable load, the study employed the Jackson queueing network model, which enabled analysis of critical system parameters, including waiting time, request intensity, and throughput. Based on the scenarios defined in the methodology (low, medium, and high load), The authors identified the key dependencies between the load level and service stability. Specifically, when moving to high load, significant increases in delays and SLA violations were observed, confirming the feasibility of adaptive scaling and other adjustment mechanisms.

The authors calculated the average waiting time for requests in the queue using Jackson queue networks, as in Equation (3), which helped model the behaviour of the microservice system at various load levels. At low load, when the request rate is much lower than the service throughput, the average waiting time is minimal. As the load increases, the waiting time gradually increases, indicating that the system is approaching its throughput limit. In critical modes, when the flow of requests approaches the maximum performance, waiting times increase sharply, leading to overloads and potential failures [17-20].

These findings confirmed the need for dynamic scaling to maintain stable performance. Such mechanisms include load balancing, horizontal scaling (adding service instances), and configuration optimisation. The use of these approaches helps to avoid overloads and maintain SLO targets. According to formula (7), Markov processes with transitions between states – normal, overloaded, and failed – were used to estimate the probability of overloading. With a 30% increase in request intensity, the model showed a 15% increase in the probability of overload, from 10% to 25%. This increase was due to queue accumulation and delays, which reduced overall performance. This approach helped not only to predict critical situations but also to initiate prompt adjustments.

Dynamic scaling, described by Equation (6), adjusts resource allocation in response to changes in the workload. Analysis of current metrics, such as CPU utilisation, the number of active requests, and average processing time, enabled the system to redistribute computing power automatically. Optimisation mechanisms detected overloads and launched relevant actions, such as scaling or load balancing. As a result of these actions, the average CPU utilisation decreased, improving performance without the need to expand infrastructure.

The proposed service-level assessment model (Equation 1) accounted for the performance of individual microservices, considering their weighting factors that reflected their criticality to overall functioning. This approach helped to allocate resources adaptively and, if individual services were overloaded, to quickly scale or reconfigure them. Applying this approach ensured a balance between the local efficiency of individual components and the overall stability of the entire system.

Based on modelling the operation of a microservice system under variable load conditions and analysing responses to overload and SLO violations, The authors developed a generalised description of the three approaches to anomaly detection. Table 2 compares their key advantages, limitations, and feasibility for use in systems with high performance requirements.

Table 2: Comparative characteristics of anomaly detection methods in microservice systems.

Method	Advantages	Limitations	Application under load conditions
Static thresholds	Ease of implementation	Sensitivity to fluctuations	Limited
Markov forecasting	Simulates probabilities	Requires configuration of parameters	Moderate stability
Proposed model	Adaptability, accuracy, SLA-oriented	Greater complexity of implementation	Efficient even at peak loads

Source: Compiled by the author of this study.

The analysis of anomaly detection methods revealed that their effectiveness depends heavily on their ability to adapt to changes in load and consistently maintain established service levels. Some approaches demonstrated limited sensitivity to anomalous changes and a high false-alarm rate, thereby reducing overall system performance. Others, on the contrary, provided greater detection accuracy and scaled better in an unstable environment. This comparison helped to evaluate the strengths and weaknesses of each approach in terms of their practical feasibility. The obtained findings served as a basis for selecting the best method based on the specific requirements for performance, reliability, and adaptability of the microservice architecture.

The use of machine learning for anomaly detection demonstrated high efficiency in detecting non-standard system behavior. The implemented algorithms, particularly the Isolation Forest and RNN, showed excellent results in analysing metrics and predicting incidents. The authors applied the Isolation Forest to analyse time-series data for metrics such as CPU utilisation, response time, and error count. The application of the authors enhanced approach ensured an anomaly detection accuracy of 92% and reduced the number of false positives to 5%. RNNs predicted 85% of incidents 5-15 minutes before they occurred, which markedly improved response times.

The data analysis also considered memory usage, providing a more accurate picture of the system's health. The key indicators for decision-making were the average CPU utilisation, service response time, and memory usage. If the enhanced monitoring system detected that the CPU utilisation parameter exceeded the threshold (80%), mechanisms to increase resources for critical services or to load-balance across microservice instances were automatically activated. The authors integrated the algorithms into a monitoring system that automatically analyzes collected data and generates alerts to enable prompt incident response, ensuring faster problem detection and resolution.

As a result of the modelling and verification steps outlined in the structured process representation, The authors achieved substantial improvements in system performance. The inclusion of processes such as formalising specifications, modelling asynchronous interactions, and optimising configurations

helped to identify and address critical performance-limiting issues. Verification confirmed that the proposed model meets the requirements for system performance and stability. Implementing the model adjustment stages based on the results of the previous simulation reduced downtime and improved overall service levels. The analysis of the final results showed that the model maintained stable compliance with the SLA/SLO parameters even under increased load. However, the study identified several barriers and limitations that complicate the integration of security and SLA management mechanisms within a microservice architecture. Despite the successful implementation of the proposed model, some aspects still require further improvement and optimisation.

One key issue is the varying user requirements for dashboards. System operators need quick access to generalised data on service state, while developers are interested in detailed performance information for individual microservices. This variability creates challenges in building a universal interface that satisfies both user groups. Another limitation lies in the cognitive load on users caused by excessive screen information. The study found that a profusion of graphical elements can complicate the quick perception of critical indicators, especially under high workload conditions.

The study also revealed a limited space for visualising a considerable number of services. In microservice architectures, there are often hundreds of interconnected services, making it challenging to display all the necessary information on a single dashboard [21-23]. The use of data aggregation methods partially solves this problem. Still, during testing, the authors noted that in some cases, generalised metrics do not enable prompt detection of the issues in the operation of individual microservices. Another challenge is integrating dashboards with notification and automatic response systems. When the enhanced monitoring system detected deviations in metrics, some configuration changes (e.g., updating access rules or adjusting authorisation policies) required manual intervention, slowing the system's ability to adapt to changes.

A separate issue is the protection of confidential data in dashboards, as they may contain critical information on service performance, request logs, and potential system vulnerabilities. Implementing Role-Based Access Control (RBAC) authentication and authorisation enables restricting access to sensitive data. Still, testing showed that dynamically updating access rights in real time is technically challenging, especially when the system reconfigures the microservice environment.

Another challenge is scalability and adaptation of dashboards to changes in the system. The analysis showed that some automatic configuration update methods did not work effectively as the number of services increased significantly. The challenges mentioned above can complicate the management of large microservice architectures and delay the display of up-to-date data. Therefore, further improvements are needed to optimise mechanisms for updating dashboard configurations and to manage information access flexibly.

The integration of multiple monitoring tools, including Prometheus, Grafana, Jaeger, and the ELK Stack, revealed several technical challenges related to data unification. These systems operate with diverse metric formats and have specific collection and processing mechanisms, which complicate indicator synchronisation and require additional processing to create a holistic picture of the system. Another limitation is the performance of centralised data storage for monitoring. The use of scalable databases

(e.g., Cassandra or Elasticsearch) enables efficient handling of large amounts of data; however, if the load is excessive, delays may occur when accessing historical data and running analytical queries. Furthermore, the increase in the number of microservices significantly increases the amount of data to be analysed, placing additional burden on metrics collection and storage systems. The use of aggregation helps partially solve this problem, but it can lead to a loss of detail in individual performance indicators.

Another problem is protecting confidential data during transmission between services. The use of TLS and encryption mechanisms provides a basic level of protection, but testing revealed that some services do not support modern encryption standards, creating potential vulnerabilities. Additionally, monitoring security events has its limitations. Centralised logging enables analysing potential threats, but high volumes of events in a distributed system can overload log storage and increase the time required to process analytical queries.

The study revealed that integrating security and SLA management mechanisms into a microservice architecture entails a range of technical and organisational challenges. These challenges require further improvements in data processing, storage, and visualisation methods, as well as the development of more flexible approaches to access control and to system adaptation to changes.

4. DISCUSSION

The study developed a comprehensive approach to improving the methods for monitoring microservice architectures, encompassing various aspects of the system, including mathematical modelling, optimisation of monitoring processes, management of service levels (SLA and SLO), security, and compliance with regulatory requirements.

Automated performance monitoring of microservice architectures is a key aspect to ensure stable system operation, dynamic SLA and SLO management, and efficient resource allocation under changing load conditions. The data obtained were consistent with the findings of Kosinska et al. [24], who emphasised the need for an integrated approach to monitoring through a combination of logs, metrics, and tracing. The present study also applied an analogous concept. Still, unlike the cited research, which mainly emphasized aggregating observed data, the authors approach implemented an automated response to performance deviations, enabling a transition from passive monitoring to active management.

Sharma and Nadig [25] identified the possibilities of using eBPF for monitoring microservices, which allowed collecting low-level metrics without a heavy load on the system. The present study also used low-level metrics (CPU and memory), but the key difference is that the authors integrated them into an automated SLA management system. Thus, not only were the metrics analysed, but they also directly influenced decision-making on scaling and resource redistribution, which provided an effective mechanism for real-time adaptation.

Borges et al. [26] investigated architectural solutions for observability and emphasised the importance of correctly interpreting metrics for effective management of microservice systems. In contrast to the researchers' approach, which was mainly concerned with architecture and data visualisation, the present study proposed a model that not only interpreted metrics but also automated anomaly analysis using machine learning algorithms. The authors' approach not only enabled recording violations but also predicted critical incidents, thereby increasing the overall level of system autonomy. Thus, the proposed approach is not limited to centralised monitoring but implements a complete cycle – from collecting metrics to automated service configuration adjustments in response to SLA deviations. Compared to earlier approaches, this ensures greater flexibility, stability, and compliance with service agreements in a dynamic environment.

The use of distributed tracing for microservice performance analysis and anomaly detection is a crucial aspect for maintaining system stability and efficiency, especially under dynamic load conditions. Zhang et al. [27] employed tracing to analyse request latency, an approach analogous to the one described in the present study, in which the authors identified critical delays by combining trace data with machine learning algorithms, such as Isolation Forest and RNN. Unlike the study mentioned above, which limited its analysis to identifying delays, the proposed model also included an adaptive response to detected anomalies, adjusting service parameters accordingly.

Ashok et al. [28] implemented tracing without modifying the code, which correlated with the automatic collection of performance data implemented in the present study without any additional load on services. However, the authors' approach supplemented with context-sensitive processing of trace events allowed us to reduce the number of false positives by accounting for typical load patterns for each service. Panahandeh et al. [29] combined tracing with performance metrics to analyse deviations in CPU, memory, and service response time, which was analogous to the approach implemented in the present study. However, the current study additionally implemented mechanisms for predicting overloads based on Markov processes, which helped to identify problems before they critically affected the SLA.

Additionally, the findings obtained correlated with those of Ding et al. [30], who used graph models to detect anomalies in microservice systems. Unlike their modelling, which uses topological connections between services, the present study employed Jackson queue networks to model load, thereby enabling more accurate assessment of resource consumption dynamics and the prediction of overload risk. The authors' approach had an analogous target orientation but provided better interpretability of the findings in the context of performance and SLAs. The findings confirmed the effectiveness of distributed tracing in detecting anomalies. Unlike most existing approaches that use tracing only as an observation tool, a distinctive feature of the proposed approach was the integration of tracing with an automatic performance management system, which allowed not only detecting problems but also adaptively eliminating them in real time by automatically scaling and reconfiguring configurations.

Efficient resource management and dynamic scaling are also critical to ensure the performance of microservice architectures, especially under variable load conditions [31-33]. Rodriguez et al. [34] studied the optimal placement of microservices within containers, which aligned with the approach

adopted in the present study, which used automatic load balancing to ensure efficient resource allocation between services. However, in contrast to the cited research, the present study closely linked the balancing mechanism to SLA-oriented control, enabling the allocation to be adjusted not only in terms of infrastructure but also in line with service commitments.

The current study also aligned with the research by da Silva Pinheiro et al. [35], who proposed modelling the performance of cloud infrastructure. However, the authors proposed approach differed in that it used mathematical models (queueing networks and Markov processes) not only to assess load but also to predict SLA deviations and trigger automated responses. Pacella et al. [36] developed a scalable framework for real-time data processing. The present study applied an analogous approach, implementing automatic microservice scaling to maintain a stable service level during load changes. At the same time, unlike in the study by Pacella et al., the system was not limited to scaling alone – it also performed SLA-oriented parameter optimisation at the level of individual services, accounting for their criticality within the overall architecture.

The findings confirmed that automated resource management and dynamic scaling are key factors in the stability of microservice architectures. Unlike earlier solutions, the authors' approach combined tracing, predictive modelling, SLA monitoring, and automated response into a single integrated system, enabling adaptive real-time resource allocation and reducing the risk of service disruption.

Formal SLA definitions, automated monitoring, and mechanisms for liability for breach of contract are essential to the effective management of microservice systems [37, 38]. Liu and Qin [39] analysed monitoring in environments with multiple service providers. Analogously to the present study, the researchers emphasised the significance of automated SLA management and monitoring in distributed systems, using formal models to verify compliance with agreements. However, in contrast to the cited approach, which focused mainly on post-facto compliance checks, the present study implemented a proactive response mechanism: automatic scaling and reconfiguration of services upon SLA violations.

This study was also consistent with the study by Wang et al. [40], who reviewed the operation management of microservice architectures, particularly in the context of SLAs. Their approach to SLA compliance monitoring, sanctioning mechanisms, and performance evaluation methods aligned with the current study, but the authors proposed model also accounted for load changes and supported adaptive resource scaling based on service weights (Equation 1). The authors' approach ensures not only the detection of violations but also their prevention.

The analysis confirmed that automated monitoring and sanctioning mechanisms are adequate for SLA monitoring. In contrast to predominantly static approaches, the authors proposed model integrated the legal aspects of the SLA with a dynamic performance management system, enabling continuous, real-time adjustment of the service level and minimising the risk of non-performance of contractual obligations.

However, there were some differences between this study's findings and those reported by other researchers. Zhang et al. [41] focused on the localisation of anomalies in microservices through detailed performance profiling and analysis of request delays within the Microservice Anomaly Localisation Tool.

Their approach enabled the identification of problem nodes but did not include load forecasting, adaptive resource management, or automated responses to SLA deviations, as in the present study. Sreemathy et al. [42] studied telemetry analytics and application monitoring, but did not use machine learning to adapt to performance changes. In study, the authors implemented automated anomaly prediction, enabling prompt overload detection and triggering appropriate scaling mechanisms to prevent SLA loss. Thus, the proposed approach ensures proactive management rather than merely post-facto analysis of situations.

Kumar et al. [43] proposed an architecture for security that does not include SLA modelling and adaptive resource management. In the present study, formal models were employed for load forecasting and performance management, helping not only maintain SLA compliance but also make adjustments before violations occur. Belkhiri et al. [44] focused on tracing for performance diagnostics but did not apply theoretical queueing models and Markov processes. In the present study, the authors employed these models for dynamic load management and accurate prediction of critical states, thereby reducing the number of SLA deviations and improving system stability.

Kratzke [45] focused on unified logs to improve observability but did not include automatic scaling based on SLAs. The authors proposed model complements monitoring with real-time automatic control mechanisms that minimise the impact of infrastructure load on performance. Sedghpour and Townend [46] studied the use of an extended Berkeley Packet Filter for monitoring microservices, reducing overhead, but did not consider SLA management or dynamic resource scaling. Amaral et al. [47] used an extended Berkeley Packet Filter to collect kernel-level metrics, but their approach did not cover SLA compliance analysis or adaptive resource reallocation. The authors proposed model combined these techniques with service control, enabling a transition from monitoring to active management. Albuquerque et al. [48] analysed monitoring patterns for cloud applications, but did not include legal aspects of SLAs, such as liability for non-performance of the contract. The present study integrated the technical and legal components of SLAs, complementing monitoring with formal compliance, verification, and compensation mechanisms. The authors' approach not only helped detect incidents but also ensured legal liability in the event of SLA violations.

To further improve and expand the capabilities of the proposed SLA and SLO management model, it is necessary to focus on several key aspects. One of the key areas for improvement is the automatic adaptation of dashboards to user needs. For operators, the system must automatically aggregate metrics and display the overall service status, and for developers, it must provide in-depth analysis of the performance of individual components. These improvements will minimise cognitive load and increase the efficiency of working with monitoring tools. Another promising area is the integration of predictive analysis algorithms to detect potential system failures and overloads. The combination of conventional monitoring with machine learning will not only analyse the current state of the system but also predict critical situations, thereby markedly improving the efficiency of resource management.

Another significant aspect of the development is the automation of incident response mechanisms. Further integration of dashboards with orchestration systems will enable automatic corrective actions, such as restarting services or changing load-balancing configuration, in response to anomalies. Another

promising area is expanding support for mobile platforms, allowing administrators to receive critical notifications and use dashboards even when they are remote. The remote option is especially relevant for large companies, where administrators may be distributed across different data centres or work in hybrid environments, as noted by Roh et al. [49] recommendations to optimise the use of mobile platforms in distributed systems.

The introduction of multi-factor authentication and advanced access control is a significant prospect. The multi-factor authentication will protect dashboards from unauthorised access and minimise the risk of confidential information leakage. Another vital area is the dynamic adjustment of data display depending on the current system load. Solving this problem will enable the monitoring system to display only critical metrics during peak times, minimizing information noise for operators.

Expanded support for standardised data exchange formats, such as OpenMetrics for metrics and OpenTracing for traces, significantly increases compatibility among monitoring tools. Using standardised data exchange formats ensures unified data processing, eliminating the need for complex conversion procedures and facilitating the integration of monitoring systems [50-53].

A promising area for improvement is automating the configuration management of monitoring infrastructure. The Infrastructure as Code approach enables automated deployment and updates to monitoring tool settings, reducing the risk of errors when modifying configurations and increasing system controllability. Additionally, the scalability of the monitoring infrastructure is key. Implementing hybrid solutions with edge components for local metric collection and processing reduces load on the central data storage. It also increases analysis efficiency in distributed systems, as confirmed by Chen et al. [54].

Optimising the storage of historical data was a prominent aspect of the study. Multi-level archiving strategies, where the authors store operational data in fast-access databases (e.g., Elasticsearch, Prometheus) and less-relevant historical data is transferred to more cost-effective storage (e.g., S3, HDFS) for analytical queries, contribute to the rational use of resources. Integration of monitoring systems with predictive analysis mechanisms also has considerable potential. The use of machine learning algorithms to analyze trends in metrics enables predicting overloads or service failures based on historical data, allowing operators to proactively address risks before they occur, which aligns with the recommendations of Tzanettis et al. [55].

To enhance control and minimize security risks, organizations should optimize centralized access control by applying the principle of least privilege and dynamically updating access policies. Improved logging and adaptive log filtering algorithms enable faster anomaly detection. To protect APIs, it is essential to implement mechanisms for request validation and rate limiting to prevent DoS attacks and data injection. Automating security policy updates with real-time vulnerability scanning helps to detect threats and reduce their impact on the system quickly [56-58].

Integration with predictive threat analysis mechanisms based on machine learning methods to assess user and service behavioural patterns enables the identification of potential risks before they occur, increasing system resilience. The development of scalable hybrid solutions with edge components

further optimises local metric processing, reducing load on the central infrastructure and increasing analysis accuracy in distributed systems [59-62].

The study highlighted the significance of an integrated approach to monitoring and managing microservices. The proposed methods cover various aspects, including performance, security, resource management, and regulatory compliance, helping effectively solve complex problems and ensure high service quality. The proposed approaches can serve as a basis for implementing new optimisation and monitoring methods in real-world applications, providing better scalability, adaptability, and security – critical for modern cloud infrastructures.

5. CONCLUSIONS

The study helped develop a holistic model of service quality management for microservice systems, encompassing technical, analytical, and regulatory components. The proposed approach combines mathematical modelling, machine learning, monitoring automation, and SLA control mechanisms into a single adaptive system focused on performance stability even under variable load conditions. The authors confirmed the effectiveness of the developed model under dynamic load-balancing conditions, resulting in reduced downtime and stabilized key service indicators.

One of the key results was the introduction of a multi-level service-level assessment system that accounted for local service performance metrics and their relative weights within the overall architecture. This approach ensured the system's adaptive response to overload of individual components through dynamic scaling and reconfiguration. The optimisation algorithms helped maintain the set SLO values without a substantial increase in resources, reflecting the effectiveness of the applied management methods. The study also demonstrated the feasibility of using machine learning algorithms (Isolation Forest and RNN) to predict failures and anomalies. Using machine learning helped to achieve high accuracy in detecting critical situations and reduce the number of false positives. As a result, the authors facilitated a proactive response to threats, thereby increasing the system’s overall reliability

The authors paid considerable attention to the legal aspects of SLAs, including verifying compliance with contractual terms, defining sanctions, and ensuring transparency of the parties’ responsibilities. The formalisation of SLAs through logic models ensured automated verification of compliance with factual performance and minimised the risk of legal violations.

However, the study also identified several limitations. The key problems were the complexity of integrating numerous monitoring tools, dashboard overload, challenges in visualising a vast number of services, and ensuring data confidentiality. The study found that centralised repositories sometimes struggle to handle a high volume of metrics, complicating real-world performance analysis.

Further research should focus on improving the scalability of monitoring systems, implementing edge components, expanding support for standards (OpenMetrics, OpenTracing), and automating tool configuration management. Another promising area is integrating predictive analytics into strategic resource planning to prevent performance degradation. Overall, the results confirmed the feasibility of

using the proposed model for SLA/SLO management in complex IT environments, particularly in cloud and hybrid architectures, where high flexibility, autonomy, and compliance with service obligations are crucial.

6. REFERENCES

[1] Dave, S.A., Nadukuru, S., Singiri, S., Goel, O., Tharan, O., Jain, A., 2024, Scalable microservices for cloud-based distributed systems. *Darpan Int. Res. Analys.* **12**(3), 776-809.

[2] Li, L., Li, S.-X., Hong, Y., Ma, J., Shi, W.-J., Li, J.-Q., 2024, Design and implementation of a business middle platform microservice operation monitoring platform. *Proc. Int. Conf. on Elect. Comm. Netw.* 116-123.

[3] D'Angelo, A., D'Aloisio, G., 2024, Grammar-based anomaly detection of microservice systems execution traces. *Companion of the 15th ACM/SPEC Int. Conf. Perf. Engin.*, 77-81.

[4] Kuzminykh, V., Koval, O., Havrylko, Y., Xu, B., Yepifanova, I., Zhu, S., Bieliaieva, N., Yeraliyeva, B., 2024, Synchronization of event-driven management during data collection. *IT, Autom. Meas. Econ. Env. Prot.* **14**(4), 121-129.

[5] Plazas Olaya, M.K., Vergara Tejada, J.A., Aedo Cobo, J.E., 2024, Securing microservices-based IoT networks: Real-time anomaly detection using machine learning. *J. Comp. Net. Comm.* **2024**, 281529.

[6] Havrylenko, Y., Kholodniak, Y., Halko, S., Vershkov, O., Miroshnyk, O., Suprun, O., Dereza, O., Shchur, T., Śrutek, M., 2021, Representation of a monotone curve by a contour with regular change in curvature. *Entropy* **23**(7), 923.

[7] Ginters, E., Revathy, J.C., 2021, Hidden and latent factors' influence on digital technology sustainability development. *Mathematics* **9**(21), 2801.

[8] Kerimkhulle, S., Aitkozha, Z., 2017, A criterion for correct solvability of a first order difference equation. *AIP Conf. Proceed.* **1880**, 040016.

[9] Havrylenko, Y., Kholodniak, Y., Halko, S., Vershkov, O., Bondarenko, L., Suprun, O., Miroshnyk, O., Shchur, T., Śrutek, M., Gackowska, M., 2021, Interpolation with specified error of a point series belonging to a monotone curve. *Entropy* **23**(5), 493.

[10] Petrakis, E.G., Skevakis, V., Eliades, P., Aznavouridis, A., Tsakos, K., 2023, ModSoft-HP: Fuzzy microservices placement in Kubernetes. *Electronics* **13**(1), 65.

[11] Sobjak, R., de Souza, E.G., Bazzi, C.L., Schenatto, K., Betzek, N.M., Gavioli, A., 2024, Incorporation of computational routines in a microservice architecture in AgDataBox platform. *Sust. Comp. Inf. Sys.* **44**, 101038.

[12] Fé, I., Nguyen, T.A., Di Mauro, M., Postiglione, F., Ramos, A., Soares, A., Choi, E., Min, D., Lee, J.W., Silva, F.A., 2024, Energy-aware dynamic response and efficient consolidation strategies for disaster survivability of cloud microservices architecture. *Computing* **106**(8), 2737-2783.

[13] Jackson, J.R., 1957, Networks of waiting lines. *Oper. Res.* **5**(4), 518-521.

[14] Fedoryshyn, B., 2025, High availability in a microservice architecture. *Bull. Cherkasy State Tech. Univ.* **30**(4), 155-165.

[15] Kozhakhmetova, D., Kaliyeva, S., Sugurova, L., Sugur, Z., Wójtowicz, R., Zhylkybayev, T., 2024, Analysis of the Distillation Column of a Catalytic Cracking Unit Using Fuzzy Input Information. *Energ.* **17**(17), 4446.

[16] Brescia, E., Costantino, D., Marzo, F., Massenio, P.R., Cascella, G.L., Naso, D., 2021, Automated multistep parameter identification of spmsms in large-scale applications using cloud computing resources. *Sensors* **21**(14), 4699.

[17] Kyurchev, V., Kiurchev, S., Rezvaya, K., Fatyeyev, A., Głowacki, S., 2024, Assessing the Reliability of a Mathematical Model of Working Processes Occurring in a Hydraulic Drive. *Lect. Not. Mech. Eng.,* 281-292.

[18] Bezshyyko, O., Dolinskii, A., Bezshyyko, K., Kadenko, I., Yermolenko, R., Ziemann, V., 2008, PETAG01: A program for the direct simulation of a pellet target. *Comp. Phys. Commun.* **178**(2), 144-155.

[19] Alekseenko, S.V., Prikhod'ko, A.A., 2014, Mathematical modeling of ice body formation on the wing airfoil surface. *Fluid Dyn.* **49**(6), 715-732.

[20] Kerimkhulle, S., Turtkarayeva, G., Mussaibekov, R., Ospanova, N., Kuttykozhayeva, S., Adalbek, A., 2025, Using Markov Chain Model to Forecasting of the Agricultural Industry Development. *Lect. Not. Networks Syst.* **1489**, 148-158.

[21] Gozhyj, A., Kalinina, I., Gozhyj, V., Vysotska, V., 2019, Web service interaction modeling with colored petri nets. *Proceed. 2019 10th IEEE Int. Conf. Intell. Data Acquis. Advan. Comp. Syst. Techn. Appl. IDAACS 2019* **1**, 319-323.

[22] Slivka, S., 2024, Microservices architecture for ERP systems. *Bull. Cherkasy State Tech. Univ.* **29**(4), 32-42.

[23] Kopytsia, V., Kvyetnyy, R., 2025, Module for integrating parking hubs with the parking lot occupancy forecasting system. *Inf. Tech. Comp. Eng.* **22**(1), 93-102.

[24] Kosinska, J., Balis, B., Konieczny, M., Malawski, M., Zielinski, S., 2023, Toward the observability of cloud-native applications: The overview of the state-of-the-art. *IEEE Access* **11**, 73036-73052.

[25] Sharma, B., Nadig, D., 2024, eBPF-enhanced complete observability solution for cloud-native microservices. *Proc. 59th IEEE Int. Conf. Comm.*, 1980-1985.

[26] Borges, M.C., Bauer, J., Werner, S., Gebauer, M., Tai, S., 2024, Informed and assessable observability design decisions in cloud-native microservice applications. *Proc. 21st IEEE Int. Conf. Soft. Arch.*, 69-78.

[27] Zhang, S., Che, Z., Pan, Z., Nie, X., Sun, Y., Pan, L., Pei, D., 2024, LabelEase: A semi-automatic tool for efficient and accurate trace labeling in microservices. *Proc. 35th IEEE Int. Symp. on Soft. Rel. Eng.*, 238-247.

[28] Ashok, S., Harsh, V., Godfrey, B., Mittal, R., Parthasarathy, S., Shwartz, L., 2024, TraceWeaver: Distributed request tracing for microservices without application modification. *Proc. ACM SIGCOMM Conf.*, 828-842.

[29] Panahandeh, M., Hamou-Lhadj, A., Hamdaqa, M., Miller, J., 2024, ServiceAnomaly: An anomaly detection approach in microservices using distributed traces and profiling metrics. *J. Sys. Soft.* **209,** 111917.

[30] Ding, S., Yuepeng, E., Zhang, J., Li, L., Zhang, L., Ge, J., 2024, Trace anomaly detection for microservice systems via graph-based semi-supervised learning. *Proc. 27th Int. Conf. Comp. Supp. Coop.Work Des.*, 2375-2380.

[31] Panchenko, A., Voloshina, A., Titova, O., Panchenko, I., Zasiadko, A., 2021, The Study of Dynamic Processes of Mechatronic Systems with Planetary Hydraulic Motors. *Lect. Not. Mech. Eng.,* 704-713.

[32] Smailov, N., Tolemanova, A., Aziskhan, A., Sekenov, B., Sabibolda, A., 2025, Implementation of fiber-optic sensing systems in structural health monitoring of concrete. *Inf. Automat. Pom. Gosp. Ochron. Srodow.* **15**(3), 73-76.

[33] Kerimkhulle, S., Kuttykozhayeva, S., Turtkarayeva, G., Seitova, T., Ospanova, N., Toleubay, D., 2025, Using the Fuzzy Logic Models for Analysis of Time Phases of Crude Oil Prices: Brent-Europe. *Lect. Not. Networks Syst.* **1489**, 9-17.

[34] Rodriguez, G., Yannibelli, V., Rocha, F.G., Barbara, D., Azevedo, I.M., Menezes, P.M., 2024, Understanding and addressing the allocation of microservices into containers: A review. *IETE J. Res.* **70**(4), 3887-3900.

[35] da Silva Pinheiro, T.F., Pereira, P., Silva, B., Maciel, P., 2023, A performance modeling framework for microservices-based cloud infrastructures. *J.Supercomp.* **79**(7), 7762-7803.

[36] Pacella, M., Papa, A., Papadia, G., Fedeli, E., 2025, A scalable framework for sensor data ingestion and real-time processing in cloud manufacturing. *Algorithms* **18**(1), 22.

[37] Brescia, E., Vergallo, P., Serafino, P., Tipaldi, M., Cascella, D., Cascella, G.L., Romano, F., Polichetti, A., 2023, Online Condition Monitoring of Industrial Loads Using AutoGMM and Decision Trees. *Mach.* **11**(12), 1082.

[38] Sekenov, B., Smailov, N., Tashtay, Y., Amir, A., Kuttybayeva, A., Tolemanova, A., 2024, Fiber-Optic Temperature Sensors for Monitoring the Influence of the Space Environment on Nanosatellites: A Review. *Mech. Mach. Sci.* **167**, 371-380.

[39] Liu, J., Qin, Y., 2024, CSPUMS: Pioneering integrated monitoring in multi-service provider ecosystems. *Proc. 9th Int. Conf. Cloud Comp. Big Data Analyt.*, 211-215.

[40] Wang, L., Jiang, Y.X., Wang, Z., Huo, Q.E., Dai, J., Xie, S.L., Li, R., Feng, M.T., Xu, Y.S., Jiang, Z.P., 2023, The operation and maintenance governance of microservices architecture systems: A systematic literature review. *J. Soft. Evol. Proc.* **35**(10), e2433.

[41] Zhang, D., Wu, Q., Zhao, Y., Li, Z., He, P., Zhang, Y., Duan, X., Zhang, Y., Miao, G., Tong, J., Xie, G., 2023, MALT: Fine-grained microservice profiling for request latency anomaly localization. *Proc. Int. Conf. on High Perf. Comp. & Comm. Data Sci. Sys. Smart City Depend. Sens. Cloud Big Data Sys. Appl.*, 114-121.

[42] Sreemathy, S.P., Kumar, V.C., Priyadharshini, S., 2024, Application monitoring and telemetry analytics. *Proc. 7th Int. Conf. Circuit Pow. Comp. Tech.*, 1559-1565.

[43] Kumar, A., Ahmed, T., Saini, K., Kumar, J., 2023, NEOS: Non-intrusive edge observability stack based on zero trust security model for ubiquitous computing. *Proc. 7th IEEE Int. Conf. on Edge Comp. Comm.*, 79-84.

[44] Belkhiri, A., Shahnejat Bushehri, A., Gohring De Magalhaes, F., Nicolescu, G., 2023, Transparent trace annotation for performance debugging in microservice-oriented systems (Work in progress paper). *Proc. ACM/SPEC Int. Conf. on Perf. Engin.*, 25-32.

[45] Kratzke, N., 2022, Cloud-native observability: The many-faceted benefits of structured and unified logging – A multi-case study. *Fut. Int.* **14**(10), 274.

[46] Sedghpour, M.R., Townend, P., 2022, Service mesh and eBPF-powered microservices: A survey and future directions. *Proc. 16th IEEE Int. Conf. Service-Oriented Sys. Engin.*, 176-184.

[47] Amaral, M., Chiba, T., Trent, S., Yoshimura, T., Choochotkaew, S., 2022, MicroLens: A performance analysis framework for microservices using hidden metrics with BPF. *Proc. 15th IEEE Int. Conf. on Cloud Comp.*, 230-240.

[48] Albuquerque, C., Relvas, K., Correia, F.F., Brown, K., 2022, Proactive monitoring design patterns for cloud-native applications. *Proc. 27th Europ. Conf. Patt. Lang. Prog. (EuroPLoP '22)*, 1-13.

[49] Roh, J.-Y., Choi, S.-H., Park, K.-W., 2024, OOSP: Opportunistic optimization scheme for pod deployment enhanced with multilayered sensing. *Sensors* **24**(19), 6244.

[50] Ginters, E., Cirulis, A., Blums, G., 2013, Markerless outdoor AR-RFID solution for logistics. *Proc. Comp. Sci.* **25**, 80-89.

[51] Mazhibayeva, G., Kenesbayev, S.M., Idrissov, S.N., Salgozha, I.T., Akhmetova, Z.A., 2026, Use of Innovative Methods in the Creation of Testing and Assessment Materials for Future Computer Science Teachers. *High. Learn. Res. Commun.* **16**(1).

[52] Smailov, N., Akmardin, S., Ayapbergenova, A., Ayapbergenova, G., Kadyrova, R., Sabibolda, A., 2025, Analysis of VLC efficiency in optical wireless communication systems for indoor applications. *Inf. Automat. Pom. Gosp. Ochron. Srodow.* **15**(2), 135-138.

[53] Sabibolda, A., Tsyporenko, V., Smailov, N., Tsyporenko, V., Abdykadyrov, A., 2024, Estimation of the Time Efficiency of a Radio Direction Finder Operating on the Basis of a Searchless Spectral Method of Dispersion-Correlation Radio Direction Finding. *Mech. Mach. Sci.* **167**, 62-70.

[54] Chen, Y., Fernandes, E., Adams, B., Hassan, A.E., 2023, On practitioners' concerns when adopting service mesh frameworks. *Emp. Soft. Engin.* **28**(5), 113.

[55] Tzanettis, I., Androna, C.-M., Zafeiropoulos, A., Fotopoulou, E., Papavassiliou, S., 2022, Data fusion of observability signals for assisting orchestration of distributed applications. *Sensors* **22**(5), 2061.

[56] Korotynskyi, A., Zhuchenko, O., 2020, A system of automated control for the baking process that minimizes the probability of defects. *East. Eur. J. Enter. Tech.* **2-103**, 58-67.

[57] Kondratenko, Y.P., Kozlov, O.V., Gerasin, O.S., Zaporozhets, Y.M., 2016, Synthesis and research of neuro-fuzzy observer of clamping force for mobile robot automatic control system. *Proceed. 2016 IEEE 1st Int. Conf. Data Stream Min. Proc. DSMP 2016*, 90-95.

[58] Denissova, O., Ismukhamedov, A., Konurbayeva, Z., Rakhmetullina, S., Samussenko, Y., Kulisz, M., 2025, Application of machine learning algorithms for forecasting labour demand in the metallurgical industry of the east Kazakhstan region. Appl. Comp. Sci. **21**(4), 136-158.

[59] Smailov, N., Nussupov, Y., Taissariyeva, K., Kuttybayev, A., Baigulbayeva, M., Turumbetov, M., Hryhoriev, Y., Lutsenko, S., 2025, Identification of dangerous situations in the road infrastructure using unmanned aerial vehicles. *Tech. Audit Prod. Reserv*. **6**(2), 97-102.

[60] Brescia, E., Massenio, P.R., Nardo, M.D., Cascella, G.L., Gerada, C., Cupertino, F., 2023, Nonintrusive Parameter Identification of IoT-Embedded Isotropic PMSM Drives. *IEEE J. Emerg. Select. Top. Pow. Elect.* **11**(5), 5195-5207.

[61] Savastio, L.P., Brescia, E., De Tuglie, E.E., Tipaldi, M., Cascella, G.L., Surico, M., Conte, G., Polichetti, A., 2025, Advances in non-intrusive type I load monitoring using R-statistic steady-state detection and subtractive clustering. IET Smart Grid **8**(1), e12205.

[62] Prikhod'ko, A.A., Alekseenko, S.V., 2014, Numerical simulation of the processes of icing on airfoils with formation of a barrier ice. *J. Eng. Phys. Thermophys.* **87**(3), 598-607.