PRACE NAUKOWE
Uniwersytetu Ekonomicznego we Wrocławiu
RESEARCH PAPERS
of Wrocław University of Economics and Business

Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu
Research Papers of Wroclaw University of Economics and Business
Year 2026, Vol. 70, No. 2
ISSN 2392-0041
journals.uew.pl/pn

Quality-Driven Web Service Management in Modern Organizations

Maksymilian Iwanow

Wroclaw University of Economics and Business

e-mail: maksymilian.iwanow@ue.wroc.pl

ORCID: 0000-0003-4795-0671

Quote as: Iwanow, M. (2026). Quality-Driven Web Service Management in Modern Organizations. *Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu*, *70*(2), 1-13.

DOI: 10.15611/pn.2026.2.01

JEL: L86

Abstract

Aim: To explore how organizations manage web services across diverse architectural models, including monolithic applications, internally developed microservices, and vendor systems.

Methodology: The study adopts a methodology that integrates architectural analysis with organizational patterns derived from existing literature and informal participant observation, enabling a nuanced understanding of how service management is shaped by organizational scale and system heterogeneity.

Results: The results include a systematization of service management practices across different architectural styles, supported by real-world use cases demonstrating how quality-driven service management is applied in complex IT environments.

Implications and recommendations: The findings highlight the importance of effective service management for maintaining system reliability and consistent service quality, offering practical insights and recommendations for handling complex, heterogeneous IT landscapes.

Originality/value: This study provides original value by bridging theoretical perspectives with practical experience, delivering actionable insights for IT leaders, system architects, and service managers managing evolving service-oriented infrastructures.

Keywords: web service, QoS, SLA, microservice

1. Introduction

A web service is a self-contained software component that provides functionality over a network, typically the Internet (Baravkar et al., 2024; Kreger, 2001). It performs tasks, executes transactions, and solves problems through standardised interfaces (Bean, 2010; Papazoglou, 2007), transforming the Web from a static information space into a distributed computational platform (Fensel & Bussler, 2002). Technically, a web service operates as a black box (Zhu et al., 2017), receives requests, processes them according to internal logic, and returns results, regardless of the underlying platform or implementation. This abstraction allows software components in distributed systems to interact seamlessly across organizational and technological boundaries. Service consumers access these components through different protocols and interfaces, selecting those that best fit their technical and business needs (Hao et al., 2012).

Historically, many organizations relied on isolated legacy systems that were difficult to integrate, exchange data with, or evolve (Sneed, 2006; Zahra et al., 2023). This lack of interoperability led to inefficient business processes, high maintenance costs, limited scalability, weak real-time capabilities, and growing security risks (Rochimah & Sankoh, 2016). Web services and SOA emerged as a response to these challenges, enabling flexible and standardised integration across heterogeneous systems (Conde-González et al., 2009). One of their key advantages is the ability to connect legacy systems without extensive redevelopment, significantly improving operational efficiency (Venkatraman, 2004).

Standardisation further strengthens this approach by allowing services implemented in different technologies to communicate through common protocols and design principles (Conde-González et al., 2009; Shelton, 2003). Web services are also inherently scalable, as they can be replicated across cloud and data-center infrastructures to handle increasing demand (Allamaraju, 2010; Hashemian et al., 2014). Secure communication protocols ensure the protection of sensitive data (Venkatraman, 2004), while modular service design exposes only selected business functions, supporting reuse, composition, and controlled access (Shelton, 2003).

In management theory, management itself is defined as an activity aimed at solving problems encountered while achieving objectives (Niemczyk, 2010), which involves applying knowledge and skills to make the right decisions in a conscious and goal-oriented manner. According to (Kisielnicki, 2005) management can be viewed from multiple perspectives, depending on the researcher. Generally, this includes directing organizational resources effectively and efficiently to meet goals, while ensuring people know what is expected of them and how to deliver results at the lowest possible cost. Management can also be analysed along time and organizational dimensions. Strategic management refers to long-term (3+ years) decision-making processes at company or sector level. Tactical management involves mid-term planning (1-3 years) typically at division level, while operational management focuses on short-term execution (less than one year) at departmental level (Kałkowska et al., 2010).

The results of the study include a systematisation of service management practices tailored to different service architecture types. The article also presents real-world use cases to demonstrate how quality-focused service management is applied in complex IT environments.

The article is organized as follows.

The “Methods” section introduces the methodological approach adopted in the study.

The “Results and Discussion” section presents the architectural models, examines quality management across these architectures and provides practical use cases.

The “Conclusions” section summarises the key findings and discusses their implications.

2. Methods

This study investigated how organizations manage web services across different architectural styles, including monolithic systems, internally developed microservices, and externally provided vendor applications. To capture this diversity, the author adopted a mixed qualitative methodology that combines informal participant observation (Jackson, 1983), literature review and architectural analysis.

A central element of the empirical perspective is the author's long-term professional experience as a Java web services developer in a large Polish bank, where over approximately five years he worked across multiple projects and teams within the organization. Following the methodological perspective described in (Glinka & Czakon, 2021; Kaczmarek, 2016), the form of unstructured or informal participant observation does not rely on predefined hypotheses. Instead, it allows organizational practices, recurring patterns, and practical challenges to emerge naturally through everyday professional activity.

The studied organization delivers a broad spectrum of IT services and operates within a highly heterogeneous technological landscape. This includes both heavily legacy core banking systems and modern, cloud-native microservice architectures built using contemporary frameworks and development paradigms. Working in such an environment provided direct exposure to multiple approaches to service design, deployment, integration, composition, and load balancing, as well as to different methods of quality and reliability management. Observing how similar challenges related to quality, scalability, and reliability are addressed across varied technical and organizational contexts offers valuable insight. Such diversity makes it possible to analyse how comparable problems are handled in systems ranging from tightly coupled legacy platforms to loosely coupled, service-oriented ecosystems.

The observation process consisted of four stages.

The first stage involved entering the organizational environment of a company that develops and maintains heterogeneous services implemented in various technologies and designed for different business goals.

The second stage focused on observing and identifying recurring patterns and practices related to web service management. Attention was given to how services were designed, deployed, and maintained within the organizational environment. This stage also included analysing how teams handled issues related to reliability, integration, and operational stability.

The third stage involved active participation in projects that implemented, maintained, and managed such solutions. This included contributing to the development and integration of web services within existing systems and supporting their deployment and maintenance in production environments. Through this involvement, practical insights were gained into how service management practices operate in real organizational settings.

The final stage consisted of analysing the collected insights and formulating conclusions based on the accumulated professional experience.

In the analytical phase, the observations collected during the projects were examined through a qualitative pattern identification process. Recurring architectural solutions, management practices, and quality-related challenges were compared across different systems and organizational contexts. Based on this comparative analysis, the observed approaches to service management were grouped into three conceptual architectural models (A1, A2, A3), representing monolithic, internally distributed, and hybrid service environments. This analytical process made it possible to identify common characteristics, differences in governance mechanisms, and typical quality management practices associated with each architectural model.

These empirical insights were complemented by a narrative literature review (Jahan et al., 2016; Kowalczyk-Anioł & Kay Smith, 2024) drawing on key academic and technical publications related to

web services, service quality, and system architecture. Rather than applying a rigid systematic structure, this review was used to contextualise the observations from practice, allowing recurring themes and gaps between theory and practice to be identified. In this way, practical experiences and scholarly contributions inform and refine each other.

Additionally, two illustrative use cases (Armour & Miller, 2001) are presented to demonstrate how the identified principles and quality concerns manifest in concrete service scenarios. These examples translate abstract concepts into operational settings, making the findings more accessible than theoretical models.

Taken together, this approach, combining literature-based analysis, field observation, and illustrative use cases, provides a nuanced understanding of how web service management and quality practices vary across organizations with different levels of scale, technical maturity, and system complexity.

3. Results and Discussion

3.1. Quality Management across Web Service Architectures

Rapid technological expansion has increased the need to use web services distributed across different locations. To satisfy user expectations, numerous non-functional requirements must be considered. Factors such as response time, availability, reliability, security, and cost play an important role in the overall quality of a service. Since many web services often provide similar functionality, non-functional attributes frequently become the main criteria for service selection. Various attempts have been made to structure the concept of web service quality. For example, Kim and Lee (2005) distinguished elements such as quality factors, quality associates, and quality activities, which together describe non--functional properties, stakeholders involved in service usage, and actions taken to maintain service quality. Similarly, ISO/IEC (2011) standards define system quality as the degree to which a system satisfies the stated and implied needs of stakeholders, including aspects such as performance, reliability, security, and maintainability.

To understand how web service management works in practice, it is necessary to consider both the organizational context in which services are delivered and the way service quality is managed within that context. Thus this chapter introduces three representative organizational and architectural models labelled A1, A2, and A3. Each model reflects a different way of building and operating web services, ranging from a monolithic service (A1), through an internally distributed multi-unit architecture (A2), to a hybrid model integrating external vendors (A3).

3.1.1. A1 Architecture: Monolithic Architecture – Centralised Simplicity – Small Organization

Organization A1 provides its services through a single, monolithic web service (Mehta et al., 2024). All business logic is encapsulated within one application, developed, maintained, and operated internally (Danowski, 2022). All roles, i.e. developers, DevOps engineers (Giamattei et al., 2024), analysts, and administrators, are centred around this single product. Owing to this tightly coupled setup, A1 avoids common communication challenges that typically arise between vendors, departments, or teams.

Strategically, all responsibility lies within one division. This centralised focus enables quick issue resolution, since the company controls all aspects of the service lifecycle. From a tactical perspective, service monitoring and optimisation are streamlined due to the unified infrastructure.

This model is common in smaller companies, offering limited functionalities and operating with modest resources. The main advantage is operational agility – any issue can be quickly diagnosed and resolved. However, as the organization grows and functional requirements increase, the limitations of

monolithic architecture become more apparent. Without structural changes, it becomes difficult to scale or support evolving user needs.

In organizations following the A1 model, service quality management is tightly integrated with the application itself. The approach primarily focuses on real-time monitoring of system usage, performance, and failures. Administrators and analysts examine metrics such as:

- number of service requests,
- share of correct vs. incorrect responses,
- frequency of method usage,
- cpu and memory utilisation.

These indicators allow teams to detect root causes of failures or inefficiencies (Farrell & Kreger, 2002; Papazoglou & van den Heuvel, 2005). For example, high response times (Iwanow & Wrzuszczak-Noga, 2021) may indicate suboptimal internal method design or outdated data transfer protocols. Availability issues can arise if the number of active service instances is too low to handle demand. Problems with reliability may stem from unhandled errors or unstable code blocks, requiring targeted refactoring, whereas poor documentation or firewall restrictions can hinder integration. In this tightly controlled environment, every aspect of the service lifecycle is handled in-house, allowing for swift resolution, yet scalability remains a challenge as the organization grows.

3.1.2. A2 Architecture: Internal Microservices – Decentralised Collaboration – Medium Organization

In contrast, organization A2 adopts a microservices-based approach (Larrucea et al., 2018; Ma et al., 2020) (Figure 1), where multiple independent services are developed and maintained by different teams within the same company (Apel et al., 2018). Each service fulfills a specific role within the broader ecosystem, and together they deliver complex functionality to the end user.

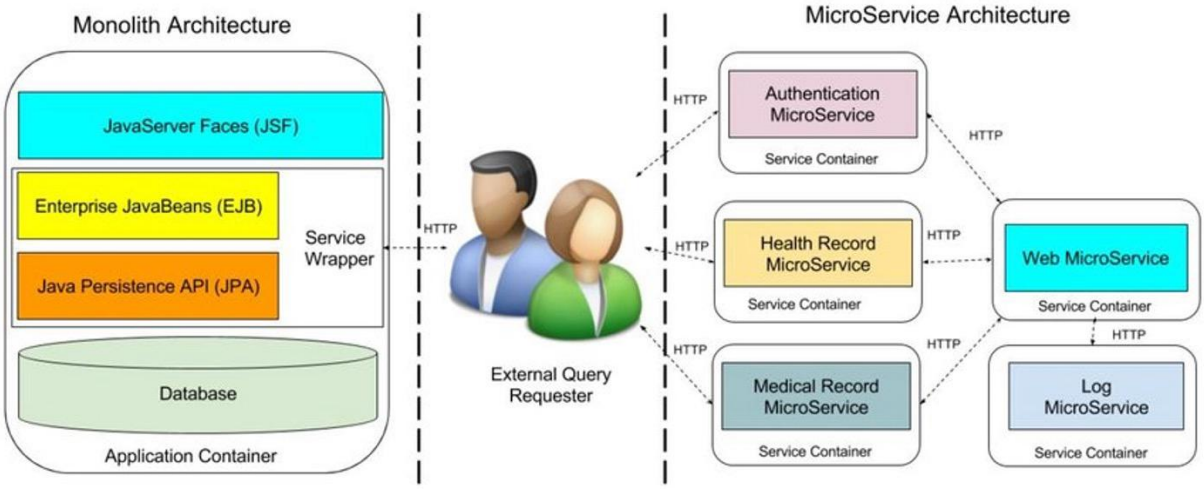

Fig. 1. Monolithic architecture vs microservice architecture

Source: (Yang et al., 2018).

While this architecture supports modularity and scalability, it also introduces greater complexity in management. When a process spans several internal services, coordination becomes critical. Each unit must ensure its component performs reliably and integrates effectively with others. If one team underperforms or a critical developer resigns, knowledge silos can emerge, making maintenance and continuity difficult.

A2-like organizations are often structured to outsource services to larger business units. The distributed development model supports agile workflows and rapid feature deployment. However, knowledge fragmentation and high personnel turnover present significant risks.

In the A2 model, where multiple services are developed by different internal teams, service level agreements (SLA) (Chippagiri & Ramesh, 2025; Swain & Garza, 2023) serve as a coordination tool to ensure consistent quality across units. The service guarantees are around what transactions need to be executed and how well they need to be executed (Jin et al., 2002). SLAs are not especially standardised or uniform in breadth, but rather they address aspects that are pertinent regarding a particular network component for that customer and provider (Iwanow, 2025; Machiraju et al., 2003):

- The agreement may have a guaranteed service level response time, throughput, period to resolve on incidents i.e. events that are associated with service failures (Ranabahu et al., 2009).
- The agreement may guarantee some release timeframe, or application installations on the servers, or creating a bandwidth pipe between subnets for specific applications.
- The agreement should determine as many specific events within a company environment as possible.

Although internal (in-house) SLAs are typically simpler than external contracts, they still define expectations related to:

- response time and reliability,
- availability (through service replication and failover mechanisms),
- documentation completeness and accessibility.

SLAs in this model rely on shared standards across the organization, such as a unified tech stack, internal security practices, and integration protocols (Kotsokalis, 2011). Each team is responsible for making its service:

- properly registered and discoverable,
- functionally stable and logically correct,
- fully documented, including method interfaces, request structure, and endpoints.

3.1.3. A3 Architecture: Hybrid Architecture – Integration of Internal and Vendor Services – Large Organization

Organization A3 represents the most complex model. It delivers functionality through a hybrid architecture, combining in-house developed services with external, vendor-provided (Figure 2) components (Milić & Makajić-Nikolić, 2022). These services may be hosted in the company's own infrastructure (Cherkasova, 2000), or provided as managed services in the cloud (Borra, 2024; Rajagopal et al., 2021).

This model introduces additional challenges – particularly when the company does not have access to the external service's source code. Diagnosing issues becomes difficult, especially when problems are unrelated to infrastructure or connectivity. Managing services that handle sensitive data adds another layer of complexity, due to regulatory obligations (Martinez Suñé & Lopez Pombo, 2020; Xiao, 2008; Yimam & Fernandez, 2016) and divided responsibilities between organizations (Simon et al., 2021).

A3-type organizations are typically large enterprises, offering numerous web-based services at scale. One advantage of this model is the clear separation of responsibilities, where each team (internal or vendor) is accountable for its own domain. Additionally, the company can focus its resources on developing only the services that deliver unique business value, outsourcing the rest.

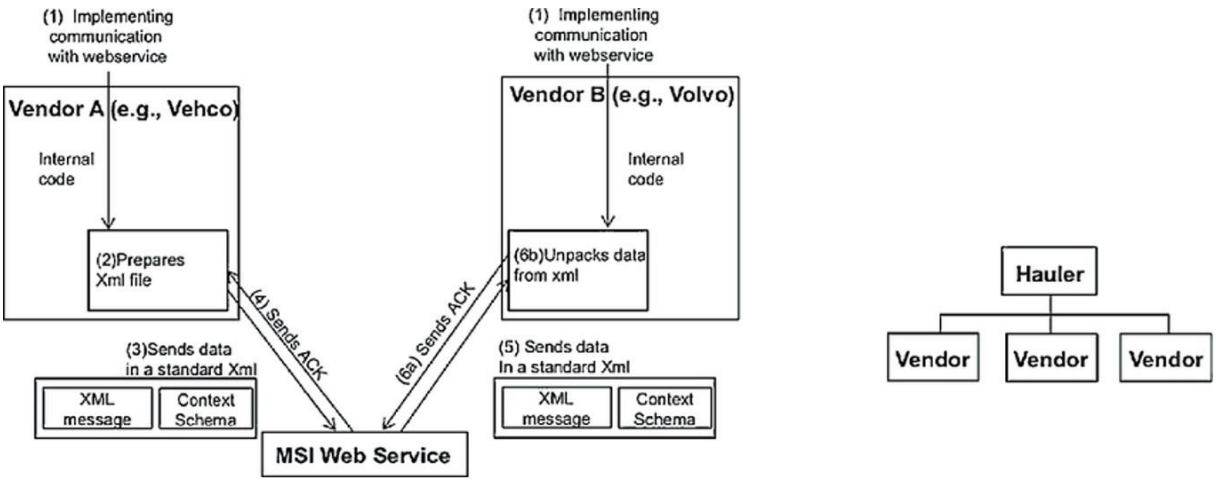

Fig. 2. Example of the vendor services adaptation

Source: (Saadatmand et al., 2019).

However, this also creates knowledge fragmentation as no single person fully understands the complete system. Strategic decisions must often be made with incomplete information, and managing service quality across loosely connected components – especially under SLA constraints – can be extremely challenging.

In the A3 architectural model where organizations integrate both internal and external services, managing service quality requires a comprehensive and precise approach to SLAs. These agreements must go far beyond the scope typical of internal arrangements as they define the foundation of cooperation between the organization and external service providers.

One of the first critical components of such an SLA is the specification of service delivery and deployment conditions. This includes defining the types of environments in which services are hosted (e.g. cloud vs. on premises (Albuquerque et al., 2023; Borra, 2024; Rajagopal et al., 2021)), the number and structure of test subnets, and the deployment pipelines involved. Clear documentation of these elements ensures proper planning for integration and operational resilience.

Equally important is the definition of communication and security standards. The SLA should explicitly state how services are allowed to interact, which protocols and architectural styles are used (e.g. REST (Golmohammadi et al., 2022; Juneau & Telang, 2022), SOAP (Hammoudeh & Al-Ajlan, 2020)), and what encryption or authentication mechanisms are in place to safeguard transferred data. These agreements must ensure that all payloads are securely transmitted and processed in accordance with both technical best practices and legal compliance requirements.

A well-structured SLA must also include a support and incident management framework (Swain & Garza, 2023) detailing:

- contact channels for reporting bugs or performance issues,
- required format and content of incident reports,
- categorisation of incidents (e.g. critical, high, low priority),
- guaranteed resolution times for each category.

Such granularity is essential for rapid response and accountability. Categorising issues based on severity, and assigning strict response timeframes, helps ensure that mission-critical processes are not delayed due to unresolved external service malfunctions.

Another indispensable element is the policy for handling sensitive information (e.g. Chippagiri & Ramesh, 2025). Organizations operating with legally protected or sensitive data must define how logs are stored, which types of information can appear in application logs, and how databases are

secured. The SLA should specify not only the preventive mechanisms for avoiding data leaks but also corrective and contingency measures in case of service failure affecting data integrity or compliance.

Table 1. Comparison of the A1, A2, and A3 architectures

Criterion	A1: Monolithic architecture	A2: Internal microservices	A3: Hybrid architecture
Organizational context	Small organization with one central development team	Medium organization with multiple internal teams	Large organization integrating internal and vendor services
Architecture structure	Single, tightly coupled application	Multiple independent services developed internally	Combination of internal services and external vendor services
Management complexity	Low: centralised control and infrastructure	Medium: requires coordination between teams	High: coordination between internal units and external providers
Scalability	Low: difficult to scale with growing demand	Very high: services can be scaled independently	High: dependent on both internal and vendor capabilities
Quality management approach	Centralised monitoring of system metrics (usage, performance, failures)	Internal SLAs between teams to ensure service reliability and performance	Formal SLAs with vendors defining performance, security and support conditions
Main advantage	Simplicity and quick problem resolution	Flexibility and modular development	Ability to combine internal capabilities with external solutions
Main limitation	Poor scalability as system grows	Knowledge fragmentation and coordination challenges	High governance complexity and limited control over external services

Source: own work.

In summary, external SLA management is a complex but vital layer of quality assurance in hybrid architectures. It standardises cooperation rules with vendors, aligns mutual expectations, and mitigates risk associated with service failures or non-compliance. However, because each external service may differ in functionality, architecture, and legal context, no universal SLA model applies. Crafting effective agreements requires collaboration between legal teams and technically experienced service managers to ensure both enforceability and operational practicality.

3.2. Real-World Scenarios in Web Service Management

3.2.1. Use Case 1: Responding to Production Failures in a Monolithic Application

Shortly after deploying a new version of a monolithic application to production, everything initially seemed fine. The service was running, users were receiving expected functionalities, and logs did not show any critical issues. Communication between the frontend and backend worked as expected.

However, a few hours later – after standard working hours – an issue emerged. One of the service instances started returning HTTP 500 (Popa et al., 2010) errors on roughly 50% of requests. The logs

only showed timeout exceptions, and no other clear indicators. Moreover, the log file began growing rapidly due to repeated error stack traces, causing the disk partition to fill up. The only person on call was a DevOps engineer, as the development team had already left for the day. What should be done in this situation?

Step 1: Roll back the deployment

Since the instability appeared a few hours after the release, it is likely linked to the new version. The DevOps engineer should immediately roll back to the previous, stable version across all production instances – even if the issue only appears on one instance. This action helps restore key Quality of Service (QoS) (Balakrishnan & Sangaiah, 2017; Duboc et al., 2022; Iwanow et al., 2022) parameters: reliability (successful request handling), availability, and accessibility.

Step 2: Manage log growth

If the log file is growing uncontrollably and threatening to fill the partition, increasing the partition size is a safer approach than deleting logs. These records contain crucial data that will help developers debug the issue. Preserving logs supports better documentation and long-term service improvement.

Step 3: Scale the service

Due to the service issues, many user requests failed or were delayed. Scaling out the application by deploying additional instances helps handle the increased traffic, reduce response times, and avoid cascading failures.

Step 4: Analyse and resolve the root cause

The next day the development team investigated and discovered a memory leak caused by improper use of a static collection in code written by a junior developer. If the rollback had not happened, all instances would have eventually crashed. The bug was fixed, and a new, stable version was released.

3.2.2. Use Case 2: Handling Vendor Service Failures through SLA Procedures

A customer of a large insurance company purchased travel insurance online. Although the payment was successful, the client did not receive the usual push notification confirming the transaction. Concerned that the payment had not gone through, the user contacted the company's support centre.

Upon investigation, the internal systems confirmed that the payment had been processed correctly. The issue lay with a third-party vendor service responsible for sending push notifications. Since this service is hosted on cloud and managed externally, the insurance company's developers had no access to the source code or internal mechanics – only a log entry showing a vendor-side error.

Thankfully the SLA between the insurer and the vendor outlined a clear process for handling such incidents, which included a standardised form with required fields:

- a detailed description of the issue and process flow,
- the incident's priority level: critical, high, medium, or low.

Choosing the correct priority level is crucial. Marking every issue as 'critical' can strain vendor relations and drive-up support costs, yet downplaying real problems risks leaving them unresolved for too long. In this case the team assigned a medium priority as the payment was successful, and the missing push notification did not affect the core transaction.

The form, complete with logs and description, was submitted through the agreed communication channel, ensuring both sides could track the incident's status.

Five days later – well within the 10-day SLA for medium-priority issues – the vendor deployed a new version of the service. The problem had been caused by the system's inability to send push messages to foreign phone numbers. This was resolved, and the incident was officially closed. The insurance

company was then able to respond to the customer with a clear explanation and reassurance that future notifications should work as expected.

4. Conclusions

This study explored the management of web services across different architectural models commonly found in modern organizations. Three primary types of architectures were distinguished: monolithic systems, internally distributed services, and hybrid models involving external vendors. For each model, specific management approaches were described, focusing on key quality parameters such as reliability, availability, documentation, and security.

Additionally, real-world use cases were presented to illustrate how these management strategies are applied in practice. These examples were based on actual business experiences and supported by academic literature, highlighting both technical and organizational challenges in ensuring service quality across complex IT ecosystems.

From a managerial perspective, the findings indicate that service quality management must be aligned with the underlying architectural model of the organization. While monolithic environments allow centralised control and rapid issue resolution, microservice and hybrid architectures require more formalised coordination mechanisms such as internal or external SLAs and clearly defined governance structures. Understanding these differences can help organizations design more effective service monitoring and quality assurance strategies as their systems evolve.

This study also had several limitations that should be acknowledged. The analysis was primarily based on qualitative observation of selected organizational cases and therefore did not aim to provide statistically generalisable results. Instead, the presented architectural models and management approaches should be interpreted as conceptual patterns derived from practical experience and supported by existing literature.

As web technologies continue to evolve rapidly, new architectural patterns and management scenarios are constantly emerging. Therefore, as part of future research, it is recommended to expand the set of analysed use cases. Ongoing observation and adaptation are crucial to maintaining effective service management in the face of continuous technological change.

References

Albuquerque, C., Relvas, K., Correia, F. F., & Brown, K. (2023). Proactive Monitoring Design Patterns for Cloud-Native Applications. In *EuroPLop '22: Proceedings of the 27th European Conference on Pattern Languages of Programs*. Association for Computing Machinery. https://doi.org/10.1145/3551902.3551961

Allamaraju, S. (2010). *RESTful Web Services Cookbook. Solutions for Improving Scalability and Simplicity*. Yahoo Press.

Apel, S., Hertrampf, F., & Späthe, S. (2018). Microservice Architecture Within In-House Infrastructures for Enterprise Integration and Measurement: An Experience Report. In M. Hodoň, G. Eichler, C. Erfurth, & G. Fahrnberger (Eds.), *Innovations for Community Services. I4CS 2018. Communications in Computer and Information Science* (vol. 863). Springer. https://doi.org/10.1007/978-3-319-93408-2_1

Armour, F., & Miller, G. (2001). *Advanced Use Case Modeling: Software Systems*. Addison-Wesley.

Balakrishnan, S. M., & Sangaiah, A. K. (2017). Integrated QoUE and QoS Approach for Optimal Service Composition Selection in Internet of Services (IoS). *Multimedia Tools and Applications*, *76*, 22889-22916. https://doi.org/10.1007/s11042-016-3837-9

Baravkar, S., Pellegrini, O., Gaikwad, P., Tilevich, E., & Song, Z. (2024). "How Can I Be of Service?" — A Comprehensive Analysis of Web Service Integration Practices. In *2024 IEEE International Conference on Web Services (ICWS)* (pp. 1206-1216). https://doi.org/10.1109/ICWS62655.2024.00144

Bean, J. (2010). *SOA and Web Services Interface Design* (pp. 43-54). Morgan Kaufmann. https://doi.org/10.1016/C2009-0-19988-9

Borra, P. (2024). A Survey of Google Cloud Platform (GCP): Features, Services, and Applications. *International Journal of Advanced Research in Science Communication and Technology*, *4*(3), 191-199. https://doi.org/10.48175/IJARSCT-18922

Cherkasova, L. (2000). FLEX: Load Balancing and Management Strategy for Scalable Web Hosting Service. In *Proceedings ISCC 2000. Fifth IEEE Symposium on Computers and Communications* (pp. 8-13). https://doi.org/10.1109/ISCC.2000.860535

Chippagiri, S., & Ramesh, A. (2025). PCI DSS: A Critical Analysis of Implementation, Effectiveness, and Legislative Impact in Payment Card Security. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *11*(1), 1258-1266. https://doi.org/10.32628/CSEIT251112115

Conde-González, M., García-Peñalvo, F. J., Guerrero, M. J., & Forment, M. A. (2009). Adapting LMS Architecture to the SOA: An Architectural Approach. In *2009 Fourth International Conference on Internet and Web Applications and Services* (pp. 322-327). https://doi.org/10.1109/ICIW.2009.54

Danowski, R. (2022). *Jak zdusić monolit i przejść na mikroserwisy? Część 1*. Techblog.ing.pl. https://techblog.ing.pl/blog/jak-zdusic-monolit-i-przejsc-na-mikroserwisy-czesc-1

Duboc, L., Bahsoon, R., Alrebeish, F., Mera-Gómez, C., Nallur, V., Kazman, R., Bianco, P., Babar, A., & Buyya, R. (2022). Systematic Scalability Modeling of QoS-Aware Dynamic Service Composition. *ACM Transactions on Autonomous and Adaptive Systems*, *16*(3-4), 1-39. https://doi.org/10.1145/3529162

Farrell, J., & Kreger, H. (2002). Web Services Management Approaches. *IBM Systems Journal*, *41*(2), 212-227. https://doi.org/10.1147/sj.412.0212

Fensel, D., & Bussler, C. (2002). The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, *1*(2), 113-137. https://doi.org/10.1016/S1567-4223(02)00015-7

Giamattei, L., Guerriero, A., Pietrantuono, R., Russo, S., Malavolta, I., Islam, T., Dînga, M., Koziolek, A., Singh, S., Armbruster, M., Gutierrez-Martinez, J. M., Caro-Alvaro, S., Rodriguez, D., Weber, S., Henss, J., Vogelin, E. F., & Panojo, F. S. (2024). Monitoring Tools for DevOps and Microservices: A Systematic Grey Literature Review. *Journal of Systems and Software*, *208*. https://doi.org/10.5445/IR/1000166110

Glinka, B., & Czakon, W. (2021). *Podstawy badań jakościowych*. Polskie Wydawnictwo Ekonomiczne.

Golmohammadi, A., Zhang, M., & Arcuri, A. (2022). *Testing RESTful APIs: A Survey*. arXiv:2212.14604. https://doi.org/10.48550/arXiv.2212.14604

Hammoudeh, M. A. A., & Al-Ajlan, A. (2020). Implementing Web Services Using PHP Soap Approach. *International Journal of Interactive Mobile Technologies (iJIM)*, *14*(10), 35-45. https://doi.org/10.3991/ijim.v14i10.14391

Hao, Y., Zhang, Y., & Cao, J. (2012). A Novel Qos Model and Computation Framework in Web Service Selection. *World Wide Web*, *15*, 663-684. https://doi.org/10.1007/s11280-012-0157-5

Hashemian, R., Krishnamurthy, D., Arlitt, M., & Carlsson, N. (2014). Characterizing the Scalability of a Web Application on a Multi-Core Server. *Special Issue on International Conference on Performance Engineering 2013 (ICPE 2013)*, *26*(12), 2027-2052. https://doi.org/10.1002/cpe.3239

ISO/IEC. (2011). *ISO/IEC 25010:2011, Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (Square)—System and Software Quality Models*.

Iwanow, M. (2025). Web Service Business Models: Between Theory and Practice. *Business & IT*, *XV*(2), 8-15. https://doi.org/10.14311/bit.2025.02.02

Iwanow, M., Dudycz, H., & Michalak, K. (2022). Required Quality of Service Attributes in the Context of Various Types of Web Services. In M. Ganzha, L. Maciaszek, M. Paprzycki, & D. Ślęzak (Eds.), *Communication Papers of the 17th Conference on Computer Science and Intelligence Systems* (vol. 32, pp. 261-267). https://doi.org/10.15439/2022F104

Iwanow, M., & Wrzuszczak-Noga, J. (2021). Prediction of the Response Time in Web Services Based on Matrix Factorization. In L. Barolli, I. Woungang, & T. Enokido (Eds.), *Advanced Information Networking and Applications* (vol. 225, pp. 345-354). Springer. https://doi.org/10.1007/978-3-030-75100-5_30

Jackson, P. (1983). Principles and Problems of Participant Observation. *Geografiska Annaler: Series B, Human Geography*, *65*(1), 39-46. https://doi.org/10.1080/04353684.1983.11879487

Jahan, N., Naveed, S., Zeshan, M., & Tahir, M. A. (2016). How to Conduct a Systematic Review: A Narrative Literature Review. *Cureus, 8*(11), e864. https://doi.org/10.7759/cureus.864

Jin, L.-J., Machiraju, V., & Sahai, A. (2002). *Analysis on Service Level Agreement of WEB SERVICES*. Hewlett-Packard Company Technical Reports.

Juneau, J., & Telang, T. (2022). *Java EE to Jakarta EE 10 Recipes* (pp. 511-530). Apress Berkeley. https://doi.org/10.1007/978-1-4842-8079-9

Kaczmarek, Ł. (2016). Między *survey research* a obserwacją uczestniczącą: Rozdarcia metodologiczno-tożsamościowe w polskiej etnologii/antropologii kulturowej w XXI wieku. *Etnografia. Praktyki, Teorie, Doświadczenia*, (2). https://doi.org/10.4467/254395379EPT.16.006.6485

Kałkowska, J., Pawłowski, E., Trzcielińska, J., Trzcieliński, S., & Włodarkiewicz-Klimek, H. (2010). *Zarządzanie strategiczne. Metody analizy strategicznej z przykładami*. Wydawnictwo Politechniki Poznańskiej.

Kim, E., & Lee, Y. (2005). *Quality Model for Web Services*. Oasis Group.

Kisielnicki, J. (2005). *Zarządzanie organizacją*. Wyższa Szkoła Handlu i Prawa w Warszawie.

Kotsokalis, K. (2011). *Automated Hierarchical Service Level Agreements*. Dissertation. Retrieved June 18, 2026, from https://eldorado.tu-dortmund.de/items/fdfc7bf6-3c68-416f-a0a8-98fed3cb3f79

Kowalczyk-Anioł, J., & Kay Smith, M. (2024). (Re)definiowanie wartości parku miejskiego – narracyjny przegląd literatury. *Konwersatorium Wiedzy o Mieście*, *37*(9), 25-38. https://doi.org/10.18778/2543-9421.09.08

Kreger, H. (2001). *Web Services Conceptual Architecture (WSCA 1.0)*. IBM.

Larrucea, X., Santamaria, I., Colomo-Palacios, R., & Ebert, C. (2018). Microservices. *IEEE Software*, *35*(3), 96-100. https://doi.org/10.1109/MS.2018.2141030

Ma, W., Wang, R., Gu, Y., Meng, Q., Huang, H., Deng, S., & Wu, Y. (2020). Multi-Objective Microservice Deployment Optimization via a Knowledge-Driven Evolutionary Algorithm. *Complex & Intelligent Systems*, *7*, 1153-1171. https://doi.org/10.1007/s40747-020-00180-1

Machiraju, V., Sahai, A., & van Moorsel, A. (2003). Web Services Management Network: An Overlay Network for Federated Service Management. In *IFIP/IEEE Eighth International Symposium on Integrated Network Management* (pp. 351-364). https://doi.org/10.1109/INM.2003.1194191

Martinez Suñé, A. E., & Lopez Pombo, C. G. (2020). Quality of Service Ranking by Quantifying Partial Compliance of Requirements. In S. Bliudze, & L. Bocchi (Eds.), *Coordination Models and Languages* (vol. 12134, pp. 181-189). Springer. https://doi.org/10.1007/978-3-030-50029-0_12

Mehta, G., Pothineni, B., Parthi, A. G., Maruthavanan, D., Veerapaneni, P. K., Jayabalan, D., & Sankiti, S. R. (2024). Revisiting Monoliths: A Pragmatic Case for Transitioning from Microservices Back to Monolithic Architectures. *IJARCCE*, *13*(12), 328-337. https://doi.org/10.17148/IJARCCE.2024.131251

Milić, M., & Makajić-Nikolić, D. (2022). Development of a Quality-Based Model for Software Architecture Optimization: A Case Study of Monolith and Microservice Architectures. *Symmetry*, *14*(9), 1824. https://doi.org/10.3390/sym14091824

Niemczyk, J. (2010). Zarządzanie i menedżerowie – Dokąd zmierzamy? *Prace Naukowe Uniwersytetu Ekonomicznego we Wrocławiu*, (115), 550-555.

Papazoglou, M. P. (2007). *Web Services: Principles and Technology*. Pearson Prentice Hall.

Papazoglou, M. P., & van den Heuvel, W.-J. (2005). Web Services Management: A Survey. *IEEE Internet Computing*, *9*(6), 58-64. https://doi.org/10.1109/MIC.2005.137

Popa, L., Ghodsi, A., & Stoica, I. (2010). HTTP as the Narrow Waist of the Future Internet. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)* (pp. 1-6). Association for Computing Machinery. https://doi.org/10.1145/1868447.1868453

Rajagopal, S., Kundapur, P., & Hareesha, K. S. (2021). Towards Effective Network Intrusion Detection: From Concept to Creation on Azure Cloud. *IEEE Access*, *9*, 19723-19742. https://doi.org/10.1109/ACCESS.2021.3054688

Ranabahu, A., Patel, P., & Sheth, A. (2009). *Service Level Agreement in Cloud Computing*.

Rochimah, S., & Sankoh, A. S. (2016). Migration of Existing or Legacy Software Systems into Web Service-Based Architectures (Reengineering Process): A Systematic Literature Review. *International Journal of Computer Applications*, *133*(3), 43-54. https://doi.org/10.5120/ijca2016907801

Saadatmand, F., Lindgren, R., & Schultze, U. (2019). Configurations of Platform Organizations: Implications for Complementor Engagement. *Research Policy*, *48*(8), 103770. https://doi.org/10.1016/j.respol.2019.03.015

Shelton, W. T. (2003). *Web Services: A Strategic Analysis*. Massachusetts Institute of Technology.

Simon, P., Graham, S., Talbot, C., & Hayden, M. (2021). Model for Quantifying the Quality of Secure Service. *Journal of Cybersecurity and Privacy*, *1*(2), 289-301. https://doi.org/10.3390/jcp1020016

Sneed, H. (2006). Integrating Legacy Software into a Service Oriented Architecture. In *Conference on Software Maintenance and Reengineering (CSMR'06)* (pp. 11-14). https://doi.org/10.1109/CSMR.2006.28

Swain, A. K., & Garza, V. R. (2023). Key Factors in Achieving Service Level Agreements (SLA) for Information Technology (IT) Incident Resolution. *Information Systems Frontiers*, *25*, 819-834. https://doi.org/10.1007/s10796-022-10266-5

Venkatraman, S. S. (2004). Web-Services – The Next Evolutionary Stage of E-Business. *Journal of International Technology and Information Management*, *13*(1). https://doi.org/10.58729/1941-6679.1246

Xiao, X. (2008). *Technical, Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective*. Morgan Kaufmann.

Yang, Y., Zu, Q., Liu, P., Ouyang, D., & Li, X. (2018). MicroShare: Privacy-Preserved Medical Resource Sharing through MicroService Architecture. *International Journal of Biological Sciences*, *14*(8), 907-919. https://doi.org/10.7150/ijbs.24617

Yimam, D., & Fernandez, E. B. (2016). A Survey of Compliance Issues in Cloud Computing. *Journal of Internet Services and Applications*, *7*(1), 5. https://doi.org/10.1186/s13174-016-0046-8

Zahra, S. W., Nadeem, M., Abbasi, M. N., Arshad, A., Riaz, S., & Ahmed, W. (2023). Systematic Literature Review on Web Services in E-Commerce and Business Communication. *E-Commerce for Future & Trends*, *10*(3), 7-25.

Zhu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2017). CARP: Context-Aware Reliability Prediction of Black-Box Web Services. In *2017 IEEE International Conference on Web Services (ICWS)* (pp. 17-24). https://doi.org/10.1109/ICWS.2017.10

Zarządzanie usługami sieciowymi zorientowane na jakość we współczesnych organizacjach

Streszczenie

Cel: Celem artykułu jest analiza sposobów zarządzania usługami sieciowymi w różnych modelach architektonicznych, obejmujących aplikacje monolityczne, mikroserwisy rozwijane wewnętrznie oraz rozwiązania dostarczane przez zewnętrznych dostawców.

Metodyka: Zastosowana metodologia opiera się na jakościowej analizie architektury oraz wzorców organizacyjnych, zaczerpniętych z literatury przedmiotu i nieformalnej obserwacji uczestniczącej. Podejście to umożliwiło uchwycenie wpływu takich czynników, jak skala organizacji, dojrzałość systemów oraz ich złożoność, na sposób zarządzania usługami.

Wyniki: Rezultatem badania jest uporządkowany przegląd praktyk zarządczych w zależności od typu architektury, uzupełniony o przykłady praktyczne (*use case*) ilustrujące zastosowanie podejścia zorientowanego na jakość w złożonych środowiskach IT.

Implikacje i rekomendacje: Uzyskane wyniki podkreślają znaczenie skutecznego zarządzania usługami dla zapewnienia stabilności systemów, spójności działania oraz wysokiej jakości usług. Wskazują również praktyczne kierunki postępowania w zarządzaniu złożonymi i heterogenicznymi środowiskami IT.

Oryginalność/wartość: Artykuł łączy podejście teoretyczne z doświadczeniem praktycznym, dostarczając użytecznych wniosków dla liderów technologicznych, architektów systemów oraz menedżerów odpowiedzialnych za jakość usług w dynamicznie rozwijających się infrastrukturach IT.

Słowa kluczowe: usługa sieciowa, QoS, SLA, mikroserwis