International Journal of Advanced Engineering, Management and Science (IJAEMS)
Peer-Reviewed Journal
ISSN: 2454-1311 | Vol-12, Issue-1; Jan-Feb, 2026
Journal Home Page: https://ijaems.com/
DOI: https://dx.doi.org/10.22161/ijaems.121.11

Strategies for Migrating Monolithic Software Systems to a Microservice Architecture

Artem Agaev

Senior Technology Expert, Sberbank, Moscow, Russia

Received: 09 Jan 2026; Received in revised form: 11 Feb 2026; Accepted: 14 Feb 2026; Available online: 18 Feb 2026

Abstract *— The article examines strategies for migrating monolithic software systems to a microservice architecture amid increasing demands for modularity, scalability, and integration in corporate and industrial environments. The study integrates findings on legacy system modernization, the use of modular monoliths as an intermediate form, stepwise component replacement patterns, ontology-based service identification methods, automated decomposition models, practices of polyglot persistence, and operational characteristics of microservice platforms. The analysis shows that migration strategies emerge within a multi-component structure in which the formal basis of methods, the quality of input artifacts, the degree of automation, and the operational context jointly determine service boundaries, the distribution of data responsibilities, and the risk profile. Based on a comparison of approaches, a conceptual migration framework is proposed that links method selection to the requirements of governmental, cloud, and industrial environments, demonstrating that the success of the transition is shaped not by any single technique but by a managed combination of methods and the phased restructuring of data, interactions, and operational processes. The article argues that disregarding architectural trade-offs associated with network latency, distributed consistency, and increased security complexity leads to biased risk assessment and reduced controllability of system evolution. The findings may be useful for architects, engineers, and decision-makers planning the transition from monolithic solutions to microservice architectures in corporate and industrial contexts.*

Keywords *— **monolith migration, microservice architecture, modular monolith, service identification, polyglot persistence, automated decomposition, architectural trade-offs.***

I. INTRODUCTION

Accelerating digitalization, the rise of distributed computing, and the increasing complexity of IT landscapes are fundamentally reshaping system design approaches. Monoliths, which previously ensured stability, no longer meet current requirements for flexibility, frequent updates, and high integration loads [3]. The strengthening of ties between business processes, cloud environments, and cyber-physical systems exacerbates their vulnerability and increases the demand for modularity and scalability. Under these conditions, microservices are becoming the primary vector of development, necessitating coordinated methodological and organizational solutions.

Rising requirements for adaptability and independent module development increase the sensitivity of monoliths to changes in infrastructure and domain logic. Any intervention raises regression risks and complicates the release of updates [8]. Migration to microservices involves reassembling dependencies, transforming development processes, and creating an environment for the managed evolution of services.

The research focus is shifting from local refactoring to comprehensive migration models that include service identification methods, domain boundary analysis, decomposition patterns, and infrastructure platforms. This approach allows for the consideration of application context, the explanation

of differences in architectural scenarios, and the identification of sustainable transition trajectories. The combination of analytical methods, ontological models, automated algorithms, and engineering platforms ensures a holistic understanding of architectural transformation and its dependence on monolith properties and operating conditions.

The aim of this study is to evaluate strategies for migrating monolithic software complexes to a microservice architecture, based on a comparison of theoretical approaches, practical migration patterns, and engineering solutions that demonstrate the applicability of microservices in high-load and distributed environments.

To achieve this goal, the study addresses tasks related to identifying key factors determining migration design; classifying service extraction methods; analyzing stepwise decomposition patterns; assessing architectural requirements for communication, security, and service management; and investigating the applicability of microservice platforms in industrial and corporate contexts.

The scientific novelty lies in the formation of an integrated conceptual migration framework that unites the methodological, algorithmic, and architectural determinants of the transition from a monolith to microservices.

The research hypothesis posits that successful migration is determined by the cumulative impact of architectural, domain, and technological factors. It is assumed that the combination of domain-driven analysis, staged decomposition patterns, automated service identification, and infrastructural support forms a stable transition trajectory and ensures the controlled evolution of the system.

The scope of the study covers corporate and industrial software complexes where issues of modularity, fault tolerance, and integration are decisive. Particular attention is paid to systems functioning under conditions of high requirement dynamics and the need for rapid scaling, where the architectural inconsistency of the monolith most acutely limits development potential.

II. MATERIALS AND METHODS

The methodological basis of the study is formed by the systematization of theoretical and applied approaches describing the transformation of monolithic software complexes into a microservice architecture and the modernization of legacy systems. The source corpus includes publications from 2021–2025 devoted to decomposition strategies, service identification methods, automated migration models, architectural patterns, and industrial execution platforms for microservices. The analysis includes studies examining organizational, technological, and infrastructural factors influencing system readiness for modularity, distribution, and scalability.

The study by Abu Bakar et al. [1] analyzes practices for modernizing legacy systems and the institutional barriers to architectural transformations. Al-Qora'n et al. [2] provide a systematic assessment of the modular monolith, noting the vagueness of modularity criteria and limited scalability. Fávero et al. [3] map monolith-to-microservice modernization methods and the input artifacts used. Hassan et al. [4] propose a pattern-based scheme for automated migration and describe limitations when working with tightly coupled systems. Kazanavičius et al. [5] investigate the transition of a monolithic database to a polyglot model, recording the architectural consequences of data separation. Maharjan et al. [6] demonstrate the capabilities of AI models in monolith decomposition. Melo et al. [7] analyze human-AI collaboration in the modernization of COBOL systems. Narváez et al. [8] summarize AI methods for microservice design and their current limitations. The most detailed classification of service identification methods is presented in Oumoussa et al. [9], where an ontology of formal decomposition approaches is developed. The practical architectural basis for the industrial application of microservices is revealed in Pontarolli et al. [10], demonstrating the impact of network mechanisms and communication infrastructure on the performance of distributed applications.

The research methodology relies on three analytical directions: (1) a systematic review of theoretical, algorithmic, and engineering approaches determining the logic of migration from monolith to microservices; (2) a comparative analysis of digital and infrastructural tools influencing decomposition, integration, request routing, and the transparency of inter-component interactions; (3) the structuring of organizational and technological factors determining

the sustainability of the architectural transition, including the maturity of development processes, dependency manageability, data quality, and execution platform characteristics. This combination allows migration to be viewed as the result of the interaction of methodological, algorithmic, and architectural elements that form a stable transition trajectory and determine the viability of the microservice model in various operating environments.

III. RESULTS

Analysis of the presented corpus of work revealed significant diversity in migration strategies, differing in target architectural model, types of input artifacts, and degree of decomposition procedure automation. The study by Abu Bakar et al. [1] demonstrates that the modernization of legacy systems in the public sector relies primarily on expert interpretation of architectural dependencies, while contextual constraints of an organizational nature act as a key factor in strategy selection. In the systematic review by Al-Qora'n et al. [2], the modular monolith is interpreted as an intermediate evolutionary form allowing for the elimination of some structural defects of monoliths without proceeding to full service disaggregation. The work of Fávero et al. [3] confirms the fragmentation of the field. Decomposition methods differ in input data, ranging from user stories and code to dependency diagrams, creating a significant gap in the reproducibility of results.

Pattern-based schemes, presented by Hassan et al. [4], form a software engineering migration contour oriented toward minimizing risks when working with tightly coupled monoliths. A distinctive characteristic of this direction is the rigid structural fixation of replacement steps and the sequential isolation of functional segments. In the works of Narváez et al. [8] and Maharjan et al. [6], a shift toward AI algorithms is traceable. In particular, the use of graph neural networks and variational autoencoders allows for the extraction of latent dependencies between components, which increases the precision of service boundary delineation. However, the study by Narváez et al. [8] records significant limitations: the lack of unified validation metrics and the high dependence of result quality on the correctness of initial requirements. The ontological approach proposed by Oumoussa et al. [9] systematizes service extraction methods through the formal description of artifacts and dependencies, creating a base layer of conceptual consistency unavailable in heuristic methods.

The aggregate analysis revealed methodological contradictions. The presence of different architectural representation models (code, requirements, dependency graphs, ontologies) leads to incompatibility of decomposition criteria. Automation methods show varying sensitivity to the quality of input artifacts. Machine learning algorithms amplify errors in source data, whereas ontological schemes require significant costs for formalization. The modular monolith demonstrates utility as an intermediate state, but its boundaries are not standardized. This complicates the inclusion of this form in formal migration strategies. The formulated taxonomy of migration strategies allows for the structuring of contradictions and the comparison of methods by level of formality, automation, and artifact types. Table 1 presents an aggregated view of these differences.

Table 1 – Taxonomy of migration strategies and their characteristics (Compiled by the author based on sources: [1, 4, 8, 9])

Strategy Class	Formal Basis	Input Artifacts	Automation Level	Strengths	Limitations
Modular Monolith	Conceptual modularity; architectural segmentation	Domain model, codebase, module boundaries	Low	Reduces coupling before full migration; simpler operational model	No unified definition; limited scalability
Pattern-based Migration	Migration patterns; staged replacement	Code, dependency graphs, refactoring patterns	Medium	Controlled risk; predictable sequencing	Ineffective with tightly coupled monoliths
Ontology-	Formal	Requirements,	Medium–	High conceptual	High cost of

based Decomposition	ontology; dependency semantics	models, interaction diagrams	High	clarity; reproducible service boundaries	ontology construction
ML/GNN-based Identification	VAE, GNN; latent structural extraction	Code, traces, dependency graphs	High	Improved precision; ability to reveal hidden relations	Sensitive to data quality; lacks unified validation metrics
Knowledge-driven Modernization	Expert-driven contextual analysis	Legacy documentation, organizational constraints	Low	Captures institutional specifics; supports socio-technical constraints	Low replicability; high dependence on experts

Analysis of the taxonomy shows that no single class of strategies is universal. The architectural context, the degree of requirement maturity, and the structure of the original monolith require a combination of methods. The practical applicability of AI approaches is limited by data quality, while ontological models ensure interpretability but require significant formalization resources. The modular monolith and pattern-based schemes remain in demand as managed intermediate stages; however, source data confirm their limitations when working with heterogeneous domains. The formation of a comprehensive migration strategy requires considering these differences and adapting tools to specific system properties.

The transition from a monolith to a service architecture leads to a deep restructuring of the data model, network interaction, and infrastructural organization. The transformation of a monolithic database into a polyglot structure observed in the study by Kazanavičius et al. [5] demonstrates that data decomposition becomes the primary architectural driver of migration and determines the nature of interactions between services. The results of Pontarolli et al. [10] show that distribution and network mechanisms radically change the performance and reliability profile, forming new requirements for orchestration, load balancing, and fault tolerance.

Diagram 1 presents the MOAI service interaction model, reflecting key infrastructural dependencies and structural changes arising during the transition to a microservice architecture.

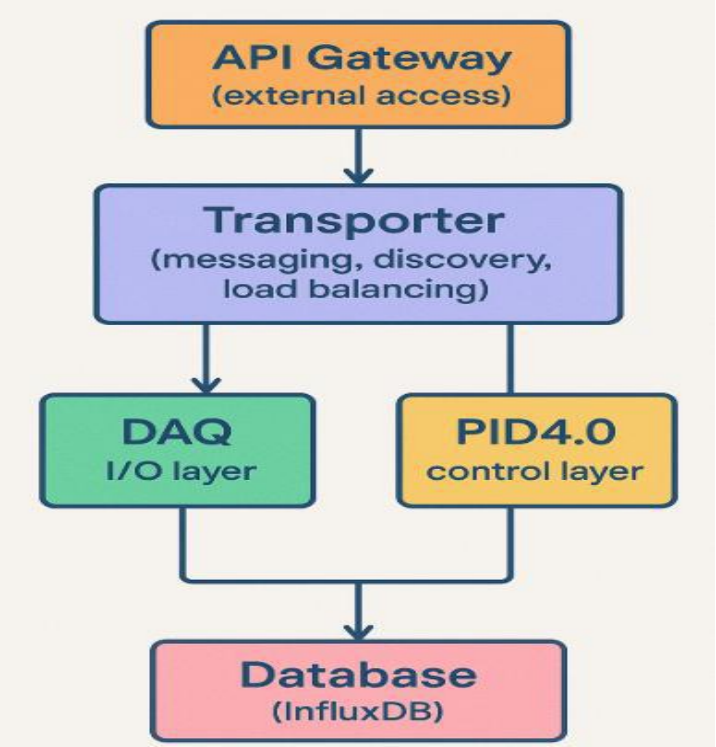

Fig.1 – MOAI Service Interaction Model
(Compiled by the author based on source [10])

The model presented in the diagram demonstrates a typical service interaction configuration in an industrial MOAI environment, where the Transporter performs the functions of a message broker, automating routing, service discovery, and fault tolerance assurance [10]. Such an organization differs sharply from the classic monolith, as data, control loops, and network mechanisms are separated and interact through standardized protocols.

The architectural consequences of migration are most clearly manifested in the data layer. In the study by Kazanavičius et al. [5], it is established that

the transition to a multi-model store increases the consistency of service boundaries but exposes previously hidden dependencies related to domain logic and transactional loops. This restructuring requires a revision of the data schema and a change in coordination mechanisms, as the single transactional model of the monolith gives way to distributed consistency patterns.

Distribution and network mechanisms investigated by Hassan et al. [4] demonstrate a dual nature of influence on architecture. Transparent routing and transport layer balancing improve scalability but introduce latency and interaction variability, which is particularly significant for real-time systems. Experimental results indicate that even with increased network configuration complexity, the microservice model remains applicable if control cycle and throughput requirements are met.

Automation plays a special role in architectural restructuring. The study by Melo et al. [7] shows that human-machine interaction accelerates migration and reduces the probability of errors arising during the manual reconfiguration of service boundaries. Automated tools, in conjunction with expert correction, form a hybrid work model in which AI reduces the load during code analysis, while the human ensures the validity of decisions.

Thus, a comparison of contexts—the public sector, cloud environments, and industrial IAS—demonstrates that the architectural consequences of migration are shaped not so much by technology choice as by the action of institutional and operational constraints. Government systems require increased transparency and change manageability, cloud platforms require elasticity and automated orchestration, and the industrial contour requires predictable network dynamics and resilience to failure.

IV. DISCUSSION

The identified patterns indicate that the character of the initial monolith determines the applicability of specific decomposition strategies. In systems with a high density of informal dependencies, architectural transformation is hindered not by a lack of tools, but by the impossibility of unambiguously defining domain boundaries, which is characteristic of government IT landscapes described by Abu Bakar et al. [1]. In such situations, methods relying on a formal domain structure yield biased results, as the structure itself is incomplete or contradictory. The analysis confirms that automation in such contexts acts not as a mechanism for replacing expert decisions, but as a means of reducing labor intensity while maintaining the leading role of human interpretation. Table 2 examines the comparison of key migration barriers and corresponding architectural solutions recorded in the literature.

The comparison of approaches demonstrates that methods based on formal models depend on the degree of domain structure distinctness. Ontological schemes, presented by Oumoussa et al. [9], ensure maximum transparency and reproducibility of the service identification process but require the presence of a consistent subject domain model. In systems where domain boundaries have evolved historically and are represented by a multitude of exceptions, such methods capture only the top level of architectural relations without covering operational dependencies.

AI models described by Narváez et al. [8] exhibit resilience in fragmented monolith landscapes due to their ability to detect latent dependencies between entities and modules. However, their effectiveness depends on the quality of source artifacts: in domains with a high degree of regulatory ambiguity, models correctly identify technical links but do not distinguish normative and organizational constraints. This aligns with the observations of Melo et al. [7], where automated methods require constant verification by experts to avoid distortions of the architectural picture.

Industrial IAS, analyzed by Pontarolli et al. [10], reveal a different class of limitations. Distributed architecture imposes heightened requirements for network predictability, response time, and fault tolerance. Here, the influence of decomposition methods is determined by module logic and permissible latency in communication between services. The authors demonstrate that changes in the data model are inextricably linked with changes in network infrastructure, creating an additional layer of architectural interdependencies absent in "softer" domains such as cloud applications.

Table 2 – Key Migration Barriers and Architectural Solutions (Compiled by the author based on sources: [2, 3, 7])

Barrier	Description	Proposed Architectural Solution
High module coupling	Dense intra-module dependencies	Modular monolith restructuring; staged refactoring
Legacy constraints	Legacy platforms limit automation	Human–AI collaboration loops
Centralized data	Single shared database	Polyglot persistence; domain-aligned partitioning
Ambiguous boundaries	Lack of clear domain semantics	Ontology-based mapping; AI-based identification
Runtime fragility	Distributed instability in IAS	Network-aware orchestration
Semantic gaps in automation	Algorithms misinterpret business rules	Expert supervision in hybrid models

In the context of cloud environments, approaches demonstrate different dynamics. Pattern-based strategies described by Hassan et al. [4] prove most applicable due to the standardized execution environment, where orchestration, scaling, and networking mechanisms are defined by the platform. In contrast, government information systems described by Abu Bakar et al. [1] are characterized by institutional constraints that limit the freedom to reorganize data and processes. A comparison of these contexts shows the varying sensitivity of architectures to decomposition methods.

The investigation of algorithmic method limitations shows that automated decomposition possesses a pronounced dependence on the nature of source artifacts. In the models described by Narváez et al. [8], grouping precision is determined by the completeness of structural dependencies: with scattered or partially lost data, algorithms form a fragmented service structure. Variational autoencoders and graph neural networks, presented in the work of Maharjan et al. [6], demonstrate a different form of limitation: they capture stable correlations between components but do not reflect the semantic layer related to business rules or regulatory restrictions. As a result, automated approaches reproduce only the technical configuration of the monolith, leaving significant domain dependencies outside the analysis.

Distributed architecture amplifies operational risks associated with network dynamics. The nature of network latency, demonstrated in Pontarolli et al. [10], illustrates the sensitivity of service interactions to the choice of protocols, routing mechanisms, and load balancing tools. increased maintenance complexity manifests in the need for version control, change coordination, and management of a growing number of failure points. Furthermore, the transition to distributed consistency requires new coordination mechanisms. Abandoning centralized transactional procedures leads to the need to redefine rules for sequence, idempotency, and operation compensation.

Data issues form a separate layer of limitations. The transition to polyglot persistence, examined by Kazanavičius et al. [5], changes the nature of transactions. Previously monolithic operations become composite and require coordination between heterogeneous stores. Complexities intensify when migrating large historical schemas where data structure was optimized for a centralized model. A need arises to define domain aggregates, redistribute responsibility between services, implement eventual consistency strategies, and introduce new data governance regulations.

Organizational and process risks manifest in various forms. In government systems described by Abu Bakar et al. [1], architectural transformation is limited by regulatory procedures affecting change speed, degree of automation, and the admissibility of architectural restructuring. In industrial IAS analyzed by Al-Qora'n et al. [2], a key role is played by requirements for the determinism and stability of network paths, which influences the choice of

orchestration mechanisms and the nature of routing. In contexts with obsolete technologies, the load on integration processes increases. Modernization requires the installation of additional bridge or adapter modules and the restructuring of distributed security models. Under conditions of high infrastructure fragmentation, the complexity of dependency management, monitoring, and failure diagnosis increases.

Security constitutes an intersecting class of risks. A distributed environment increases the number of authentication points, expands the attack surface, and requires a coordinated distributed access control policy. These changes necessitate simultaneous technological and organizational restructuring, as security policy must account for the heterogeneity of services, the difference in their execution contours, and the specifics of data exchange.

Thus, the observed structure of limitations shows that the microservice transition is accompanied by the transformation of technical, organizational, and operational mechanisms, wherein architectural trade-offs manifest in the redistribution of complexity between system levels. As the architectural form changes, interaction mechanisms and the nature of engineering responsibility change, determining the specificity of risks in each examined context.

V. CONCLUSION

Migration from a monolith to a microservice architecture should be viewed as a managed change of architectural form rather than simple code disaggregation. The consistency of the domain model, data, and operational constraints is of key importance, as it determines the balance between system flexibility, complexity, and controllability.

This requires a shift from a set of disparate tools to integrated architectural frameworks where formal methods, AI approaches, patterns, and expert decisions work jointly. It is advisable to evaluate migration success by the transparency of service boundaries, the distribution of data responsibility, and the behavioral predictability of the distributed system, rather than by the fact of using microservices per se.

Differences between governmental, cloud, and industrial contexts show the limitations of universal strategies. Architectural solutions must be tuned to regulatory requirements, permissible latencies, infrastructure specifics, and development process maturity; otherwise, uniform approaches increase institutional and operational risks. The practical migration trajectory is logically built as a staged process, combining a modular monolith, pattern-oriented component replacement, managed data decomposition, and hybrid human-machine decision-making loops.

Future research should be oriented toward applied models directly linking the choice of decomposition methods, input artifact quality, and infrastructure parameters with indicators of fault tolerance, scalability, and maintenance complexity. The priority becomes the development of metrics for evaluating the results of automated service identification, comparing the architectural trade-offs of different consistency and orchestration strategies, and forming customizable migration scenarios for typical system classes.

REFERENCES

[1] Abu Bakar, H., Razali, R., & Jambari, D. I. (2022). A qualitative study of legacy systems modernisation for citizen-centric digital government. Sustainability, 14(17), 10951. https://doi.org/10.3390/su141710951

[2] Al-Qora'n, L. F., & Al-Said Ahmad, A. (2025). Modular monolith architecture in cloud environments: A systematic literature review. Future Internet, 17(11), 496. https://doi.org/10.3390/fi17110496

[3] Fávero, L. F., Almeida, N. R. d., & Affonso, F. J. (2025). A systematic mapping study on the modernization of legacy systems to microservice architecture. Applied System Innovation, 8(4), 86. https://doi.org/10.3390/asi8040086

[4] Hassan, H., Abdel-Fattah, M. A., & Mohamed, W. (2025). A pattern-based framework for automated migration of monolithic applications to microservices. Big Data and Cognitive Computing, 9(10), 253. https://doi.org/10.3390/bdcc9100253

[5] Kazanavičius, J., Mažeika, D., & Kalibatienė, D. (2022). An approach to migrate a monolith database into multi-model polyglot persistence based on microservice architecture: A case study for mainframe database. Applied Sciences, 12(12), 6189. https://doi.org/10.3390/app12126189

[6] Maharjan, R., Sooksatra, K., Cerny, T., Rajbhandari, Y., & Shrestha, S. (2025). A case study on monolith to

microservices decomposition with variational autoencoder-based graph neural network. Future Internet, 17(7), 303. https://doi.org/10.3390/fi17070303

[7] Melo, I., Polónia, D., & Teixeira, L. (2025). Human–AI collaboration in the modernization of COBOL-based legacy systems: The case of the Department of Government Efficiency (DOGE). Computers, 14(7), 244. https://doi.org/10.3390/computers14070244

[8] Narváez, D., Battaglia, N., Fernández, A., & Rossi, G. (2025). Designing microservices using AI: A systematic literature review. Software, 4(1), 6. https://doi.org/10.3390/software4010006

[9] Oumoussa, I., & Saidi, R. (2025). The ontology-based mapping of microservice identification approaches: A systematic study of migration strategies from monolithic to microservice architectures. Computers, 14(4), 133. https://doi.org/10.3390/computers14040133

[10] Pontarolli, R. P., Bigheti, J. A., de Sá, L. B. R., & Godoy, E. P. (2023). Microservice-oriented architecture for Industry 4.0. Eng, 4(2), 1179–1197. https://doi.org/10.3390/eng4020069