# Architecture Conformance Checking for Microservice-based Systems

Ednilson G. Rossi
Federal Institute of São Paulo and Federal University of São Carlos
São Carlos, São Paulo, Brazil
ednilsonrossi@ifsp.edu.br

## Abstract

Microservice-based architectures (MSA) offer scalability, flexibility, and modularity, but their distributed and heterogeneous nature increases the risk of architectural deviation and erosion. Ensuring consistency between the planned and implemented architecture, known as Architectural Conformance Checking (ACC), is particularly challenging in MSA due to their distributed nature, heterogeneous technologies, and dynamic communication mechanisms. This research proposes an AI-assisted approach for ACC in MSA, addressing both structural and behavioral dimensions. This work aims to advance the state of the art by integrating AI into the ACC process, improving automation, accuracy, and scalability in architectural conformance verification.

## Keywords

Software Architecture, Architectural Consistency, Erosion, Drift

**ACM Reference Format:**
Ednilson G. Rossi. 2026. Architecture Conformance Checking for Microservice-based Systems. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-Companion '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3774748.3787642

## 1 Context and Motivation

Microservice-based Architecture (MSA) is an architectural pattern that evolved from approaches used in distributed systems, particularly Service-Oriented Architecture (SOA). Its core principle derives from component orientation, in which predefined software units are integrated to operate cohesively. In MSA, these components are implemented as small, independent services, each exposing a well-defined communication interface. Consequently, the system is organized as a set of autonomous services aligned with the business domain and designed to collaboratively solve problems [11, 12, 27].

Newman [12] describes microservices as characterized by technological autonomy, independent deployment, and self-managed development teams. The author also highlights the flexibility to choose persistence mechanisms and the natural alignment with continuous integration and continuous delivery (CI/CD) practices. These properties are grounded in the principles of low coupling and high cohesion among services.

A key aspect of MSA concerns service communication, which can be synchronous and blocking or asynchronous and non-blocking. Collaboration styles follow this distinction: the request–response pattern may operate synchronously or asynchronously, whereas event-driven and shared data persistence approaches rely exclusively on asynchronous communication [12].

An architecture offering such flexibility and freedom becomes a fertile environment for architectural deviations and erosion. Software evolution—driven by new requirements or maintenance activities—can introduce deviations from the originally planned architecture [6, 21]. The uncontrolled accumulation of these deviations leads to architectural erosion, marked by violations of predefined architectural rules or constraints [5, 9], which, in turn, increase structural fragility and compromise system quality.

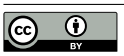
When implementation significantly diverges from the planned architecture, the benefits of a well-defined design are undermined [5]. Detecting and correcting architectural deviations thus become essential, especially in MSA, whose nature is inherently complex, dynamic, and continuously evolving.

Architectural Conformance Checking (ACC) emerges as a mechanism to identify deviations between implementation and planned architecture [1, 3, 9]. The planned architecture corresponds to the conceptual view that describes the intended software structure, representing a static abstraction of architectural elements and how these elements are expected to interact [20]. This verification process enables the detection of architectural violations, helping to preserve integrity and prevent architectural erosion over time.

We conducted a systematic literature review (SLR) [23] in 2025 as part of this research, which identified only 17 studies explicitly addressing ACC in microservice-based architectures. Among them, only two [3, 28] directly tackle the verification between planned and implemented architectures. Araújo et al. [3] propose a Domain-Specific Language (DSL)-based approach, whereas Zhong et al. [28] examine conformance between the implementation and the domain model defined by Domain-Driven Design (DDD).

Most studies instead assess architectural conformance indirectly, through best practices or architectural design patterns applied to MSA. Krieger et al. [10] explore behavioral patterns, while multiple studies [7, 15–18, 27] analyze quality attributes such as dependencies, cohesion, granularity, and coupling. Genfer and Zdun [8] addresses cyclic dependencies, whereas others assess quality requirements via Infrastructure as Code [13, 14, 19, 22]. Additional studies evaluate communication quality using REST APIs [24, 25], and Brogi et al. [4] focuses on detecting architectural smells. After analyzing the retrieved studies, we found that none of the approaches employ AI-based techniques or tools in the ACC process.

Beyond the reviewed studies, further research employing dynamic approaches to detect non-conformance in microservices




*ICSE-Companion '26, Rio de Janeiro, Brazil*

ACM ISBN 979-8-4007-2296-7/2026/04
https://doi.org/10.1145/3774748.3787642

was identified [2, 26]. However, these works remain limited to behavioral analyzes in specific scenarios, without encompassing the overall architectural structure.

Despite the growing interest in architectural consistency and conformance in Microservice-Based Systems (MBS), few studies explicitly address architectural conformance checking, as noted by Ali et al. [1] and Andrade [2]. Approaches typically focus on specific quality attributes [7, 13, 15, 27], best practices [16–19], or behavioral verification [2, 10, 26]. This scarcity of systematic and automated ACC solutions highlights a promising and underexplored research area—one that is essential for ensuring architectural integrity and sustainability throughout the continuous evolution of MBS.

## 2 Problem Statement

While ACC for monolithic systems has been extensively studied — through approaches such as Reflexion Models, Dependency Structure Matrices, and Source Code Query Languages [20, 27] — there is still no systematic approach specifically tailored for MSA. Traditional techniques operate at a low level of abstraction, focusing on classes and packages, whereas ACC in MSA demands higher-level analyzes that account for distributed components, inter-service communication, and technological heterogeneity.

In MSA, architectural properties introduce significant challenges for ACC. The inherently distributed nature of these systems complicates both the specification of the planned architecture and the reconstruction of the implemented one, whose source code may be spread across multiple repositories, implemented in heterogeneous technologies, and deployed on diverse platforms.

A further challenge arises from the heterogeneity of communication mechanisms in MSA. Interactions may be synchronous or asynchronous and supported by a variety of technologies and protocols. Moreover, systems often combine multiple communication strategies — for example, synchronous REST calls between some services and asynchronous messaging among others — resulting in a highly heterogeneous and dynamic communication landscape.

Beyond structural conformance, a second critical dimension is behavioral conformance, which requires dynamic analysis and full observability of runtime interactions. Even when structural dependencies align with the planned architecture, architectural constraints may still be violated at runtime. For example, microservices may be invoked in an order that differs from the architectural model, leading to behavioral inconsistencies.

Given these challenges and the lack of systematic approaches for ACC in MSA, this research is guided by the following questions:

**RQ1**. How can the process of architectural conformance checking be defined for microservice-based architecture?

**RQ2**. How can ACC be systematically performed in MSA, considering both structural and behavioral dimensions?

**RQ3**. Which stages of the ACC process can most benefit from the application of AI-based techniques, and how can these techniques support or enhance architectural deviation detection?

## 3 Research Proposal

This work proposes a systematic approach to architectural conformance checking in microservice-based systems, addressing two complementary dimensions. Structural conformance verifies whether architectural components—such as microservices, persistence mechanisms, and communication middleware—and their relationships are implemented according to the planned architecture. Behavioral conformance assesses whether the expected behaviors, expressed through execution scenarios or interaction flows, are reflected in the system's runtime behavior [2, 26].

The proposal faces challenges across all stages of the ACC process. Modeling the planned architecture requires representing large numbers of microservices, heterogeneous technologies, and multiple communication styles—synchronous (e.g., REST, gRPC) and asynchronous (e.g., queues, publish/subscribe, event-driven patterns) [12]. It also requires capturing architectural rules and decisions, often documented in natural-language artifacts.

Reconstructing the implemented architecture is equally challenging due to the distributed nature of MSA. Dependencies between services manifest as remote interactions observable only through tracing, logging, or other observability mechanisms [2, 10, 26]. Large Language Models (LLMs) are expected to assist in abstracting these low-level signals into higher-level architectural representations, both structural and behavioral.

Finally, the verification stage requires comparing planned and implemented architectures across different abstraction levels and correlating textual architectural rules with observed system behavior. AI-based techniques will therefore support the extraction, interpretation, and automated verification of architectural constraints, enabling more effective detection of architectural deviations.

## 4 Expected Contributions

Given the challenges and gaps identified with respect to ACC in MSA, we expected to provide the following contributions: **C1**. A systematic approach for identifying architectural deviations in MSA, addressing both structural and behavioral conformance dimensions. **C2**. A metamodel for representing the planned architecture of microservice-based systems, capable of modeling not only structural aspects but also associated architectural rules and constraints. **C3**. A technique for reconstructing and representing the implemented architecture of microservices, enabling comparison against the planned model. **C4**. A technique for capturing and analyzing runtime behavior through microservice observability mechanisms, supporting the verification of behavioral conformance. **C5**. The formalization and application of AI-based techniques within the ACC process, aimed at automating and improving the accuracy of architectural deviation detection.

Collectively, these contributions are expected to advance the state of the art in architectural consistency analysis by integrating conceptual modeling, automated reconstruction, and intelligent ACC for MSA.

## 5 Conclusion

This research addresses the challenge of ACC in MBS by considering both structural and behavioral dimensions. We propose an AI-supported approach that integrates the representation of the planned architecture, the reconstruction of the implemented system, the identification of runtime behavior, and the detection of inconsistencies between design and implementation.

## Acknowledgments

Prof. Dr. Valter Vieira de Camargo, advisor of this research, leads the Advanced Research Group on Software Engineering (AdvanSE).

## References

[1] Nour Ali, Sean Baker, Ross O'Crowley, Sebastian Herold, and Jim Buckley. 2018. Erratum to: Architecture Consistency: State of the Practice, Challenges and Requirements. *Empirical Software Engineering* 23, 3 (2018), 1868–1869. doi:10.1007/s10664-017-9542-0

[2] Ricardo Andrade. 2024. *Real-Time Conformance Checking for Microservice Applications*. Master's thesis. Eindhoven University of Technology, Eindhoven, The Netherlands. https://research.tue.nl/en/studentTheses/real-time-conformance-checking-for-microservice-applications/ Master's Thesis.

[3] Elena A Araujo, Álvaro M Espíndola, Vinicius Cardoso Garcia, and Ricardo Terra. 2020. Applying a Multi-platform Architectural Conformance Solution in a Real-world Microservice-based System. In *Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse*. 41–50.

[4] Antonio Brogi, Davide Neri, and Jacopo Soldani. 2019. Freshening the air in microservices: Resolving architectural smells via refactoring. In *International Conference on Service-Oriented Computing*. Springer, 17–29.

[5] Alessio Bucaioni, Amleto Di Salle, Ludovico Iovino, Leonardo Mariani, and Patrizio Pelliccione. 2024. Continuous Conformance of Software Architectures. In *2024 IEEE 21st International Conference on Software Architecture (ICSA)*. 112–122. doi:10.1109/ICSA59870.2024.00019

[6] Andrea Caracciolo, Mircea Filip Lungu, and Oscar Nierstrasz. 2015. A Unified Approach to Architecture Conformance Checking. In *2015 12th Working IEEE/IFIP Conference on Software Architecture*. 41–50. doi:10.1109/WICSA.2015.11

[7] Thomas Engel, Melanie Langermeier, Bernhard Bauer, and Alexander Hofmann. 2018. Evaluation of Microservice Architectures: A Metric and Tool-Based Approach. In *Information Systems in the Big Data Era*, Jan Mendling and Haralambos Mouratidis (Eds.). Springer International Publishing, Cham, 74–89.

[8] Patric Genfer and Uwe Zdun. 2021. Identifying domain-based cyclic dependencies in microservice apis using source code detectors. In *European Conference on Software Architecture*. Springer, 207–222.

[9] Lorin Hochstein and Mikael Lindvall. 2005. Combating architectural degeneration: a survey. *Information and Software Technology* 47, 10 (2005), 643–656. doi:10.1016/j.infsof.2004.11.005

[10] Christoph Krieger, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann, Vladimir Yussupov, and Uwe Zdun. 2020. Monitoring behavioral compliance with architectural patterns based on complex event processing. In *European Conference on Service-Oriented and Cloud Computing*. Springer, 125–140.

[11] James Lewis and Martin Fowler. 2014. Microservices: A Definition of a New Architectural Term. https://martinfowler.com/articles/microservices.html. Accessed: 2025-11-11.

[12] Sam Newman. 2021. *Building microservices: designing fine-grained systems.* " O'Reilly Media, Inc.".

[13] Evangelos Ntentos, Uwe Zdun, Ghareeb Falazi, Uwe Breitenbücher, and Frank Leymann. 2022. Assessing architecture conformance to security-related practices in infrastructure as code based deployments. In *2022 IEEE International Conference on Services Computing (SCC)*. IEEE, 123–133.

[14] Evangelos Ntentos, Uwe Zdun, Ghareeb Falazi, Uwe Breitenbücher, and Frank Leymann. 2023. Detecting and resolving coupling-related infrastructure as code based architecture smells in microservice deployments. In *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*. IEEE, 201–211.

[15] Evangelos Ntentos, Uwe Zdun, Konstantinos Plakidas, and Sebastian Geiger. 2021. Evaluating and Improving Microservice Architecture Conformance to Architectural Design Decisions. In *Service-Oriented Computing: 19th International Conference, ICSOC 2021, Virtual Event, November 22–25, 2021, Proceedings* (Dubai, United Arab Emirates). Springer-Verlag, Berlin, Heidelberg, 188–203. doi:10.1007/978-3-030-91431-8_12

[16] Evangelos Ntentos, Uwe Zdun, Konstantinos Plakidas, and Sebastian Geiger. 2021. Semi-automatic feedback for improving architecture conformance to microservice patterns and practices. In *2021 IEEE 18th International Conference on Software Architecture (ICSA)*. IEEE, 36–46.

[17] Evangelos Ntentos, Uwe Zdun, Konstantinos Plakidas, Sebastian Meixner, and Sebastian Geiger. 2020. Assessing architecture conformance to coupling-related patterns and practices in microservices. In *European Conference on Software Architecture*. Springer, 3–20.

[18] Evangelos Ntentos, Uwe Zdun, Konstantinos Plakidas, Sebastian Meixner, and Sebastian Geiger. 2020. Metrics for Assessing Architecture Conformance to Microservice Architecture Patterns and Practices. In *International Conference on Service-Oriented Computing*. 580–596.

[19] Evangelos Ntentos, Uwe Zdun, Jacopo Soldani, and Antonio Brogi. 2022. Assessing architecture conformance to coupling-related infrastructure-as-code best practices: Metrics and case studies. In *European Conference on Software Architecture*. Springer, 101–116.

[20] Leonardo Passos, Ricardo Terra, Marco Tulio Valente, Renato Diniz, and Nabor Mendonça. 2009. Static architecture-conformance checking: An illustrative overview. *IEEE software* 27, 5 (2009), 82–89.

[21] Dewayne E. Perry and Alexander L. Wolf. 1992. Foundations for the study of software architecture. *SIGSOFT Softw. Eng. Notes* 17, 4 (Oct. 1992), 40–52. doi:10.1145/141874.141884

[22] Pierre-Jean Quéval and Uwe Zdun. 2023. Extracting the architecture of microservices: An approach for explainability and traceability. In *European Conference on Software Architecture*. Springer, 346–353.

[23] Ednilson Geraldo Rossi and Valter Vieira de Camargo. 2025. Artifacts of the Research Process. https://github.com/ednilsonrossi/acc-in-distribuitedsistems-srl. GitHub.

[24] Apitchaka Singjai and Uwe Zdun. 2022. API description-based conformance assessment of architectural design decision. In *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 59–68.

[25] Apitchaka Singjai and Uwe Zdun. 2022. Conformance assessment of Architectural Design Decisions on API endpoint designs derived from domain models. *J. Syst. Softw.* 193, C (Nov. 2022), 19 pages. doi:10.1016/j.jss.2022.111433

[26] Chang-Ai Sun, Yufei Gong, Meng Li, Luo Xu, Jun Han, and Yanbo Han. 2024. Detecting Inconsistencies in Microservice-Based Systems: An Annotation-Assisted Scenario-Oriented Approach. *IEEE Transactions on Services Computing* 17, 5 (2024), 2194–2209. doi:10.1109/TSC.2024.3399652

[27] Uwe Zdun, Elena Navarro, and Frank Leymann. 2017. Ensuring and assessing architecture conformance to microservice decomposition patterns. In *International Conference on Service-Oriented Computing*. Springer, 411–429.

[28] Chenxing Zhong, He Zhang, Huang Huang, Zhikun Chen, Chao Li, Xiaodong Liu, and Shanshan Li. 2024. DOMICO: Checking conformance between domain models and implementations. *Software: Practice and Experience* 54, 4 (2024), 595–616.