Scientific Research Publishing

Journal of Software Engineering and Applications, 2025, 18(2), 76-86
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Microservices in Organizations

Andrew Ganje

Vuori, Enterprise Engineering & Architecture, San Diego, California
Email: Andrew.ganje@gmail.com

How to cite this paper: Ganje, A. (2025) Microservices in Organizations. *Journal of Software Engineering and Applications*, **18**, 76-86.
https://doi.org/10.4236/jsea.2025.182005

Received: April 15, 2024
Accepted: February 25, 2025
Published: February 28, 2025

Abstract

Microservices have revolutionized traditional software architecture. While monolithic designs continue to be common, particularly in legacy applications, there is a growing trend towards the modularity, independent deployability, and flexibility offered by microservices, which is further enhanced by developments in cloud technology. This shift towards microservice architecture meets the modern business need for agility, facilitating rapid adaptability in a competitive landscape. Microservices offer an agile framework and, in many cases, can simplify the development process, though the implementation can vary and sometimes introduce complexities. Unlike monolithic systems, which can be cumbersome to modify, microservices enable quicker adjustments and faster deployment times, essential in today's dynamic environment. This article delves into the essence of microservices and explores their growing prominence in the software industry.

Keywords

Microservices, Monolithic, Agile, Composability, Encapsulation, Loose Coupling, Independent Deployability, Scalability, Resilience, Cloud Computing, Digital Transformation, Competitive Advantage, Trade-Off Analysis, Data Integrity, Latency, Fault Tolerance

1. Introduction

In the modern era, we live in a rapidly moving and highly interconnected global environment. Such dynamism and connectivity demand software systems that are not only agile and adaptable allowing easy modification and swift deployments. Gone are the days when software changes could take months or even years.

In today's rapidly evolving business landscape, organizations face increasing pressure to remain agile, innovative, and competitive. The rise of digital transformation and disruptive technologies has enabled new market entrants to challenge established players at an unprecedented pace [1] [2]. Businesses that fail to adapt

swiftly to these changes risk obsolescence, particularly large enterprises burdened by rigid and outdated processes [3].

The adoption of microservices architecture has emerged as a strategic approach to fostering business agility. By breaking complex monolithic systems into modular, independently deployable services, organizations can accelerate development cycles, enhance scalability, and respond to market shifts with greater flexibility [4] [5]. Research highlights that this architecture enables faster time-to-market for new features, as development teams can work autonomously, reducing dependencies and bottlenecks in the deployment process [6] [7].

Moreover, the shift towards microservices is not merely a technical decision but a fundamental business strategy. As enterprises navigate highly competitive environments, the ability to deliver new capabilities rapidly and scale services on demand becomes a core differentiator [8] [9]. The alignment of microservices with business domains further strengthens operational efficiency, ensuring that technology infrastructures support dynamic and evolving market conditions [10] [11].

Ultimately, microservices empower organizations to innovate continuously, minimize technical debt, and remain resilient against industry disruptions. In a world where adaptability is a key determinant of success, embracing microservices can provide businesses with the agility needed to thrive in an ever-changing competitive landscape [1] [2] [12].

For organizations to thrive and demonstrate resilience, especially in the face of unprecedented challenges like pandemics, trade wars, or other geopolitical affairs as our global interconnectedness has increased, it's vital to embrace such transformative technologies. Adopting microservices may not only position businesses for success but also aid their longevity and adaptability in an ever-evolving world. Reference [2] states that the ability of your business to change quickly, innovate easily, and meet competition wherever it arises is a strategic necessity today. Organizations only using monolithic software and ideologies will hinder their ability to not only adapt but also incrementally enhance current information and communication technology at the speed required to maintain competitive advantage [1].

2. What Are Microservices?

Microservices, also known as microservice architecture, represent an architectural paradigm that structures an application as a collection of single-responsibility, autonomous services. Each service focuses on a single process or business capability and is independently deployable. This architectural style offers a departure from monolithic design, fostering flexibility, scalability, and adaptability to rapidly changing business requirements [6]. Given the increasing potential for disruptions in various industries, microservices emerge as a pivotal software consideration for organizations striving to keep pace in the rapidly evolving global landscape. Delving into this technology and contrasting it with predecessors like monolithic systems provides clarity on the essence of microservices.

2.1. Defining Microservices

Microservices are an architectural style that changes the way applications are created, tested, implemented, and maintained. By using microservices, a large application can be implemented as a set of small applications that can be developed, deployed, expanded, managed, and monitored independently [9].

The term "microservice" can be misleading, as "micro" refers not to the size of the service, but to the scope of its functionality. Each microservice is designed to handle a specific business capability, allowing for independent development, deployment, and scaling [9]. This approach enhances modularity and agility within the overall system. A microservice is designed to perform a single function or a small group of related functions usually grouped by business domain, enabling more granular and flexible scaling, easier maintenance, and better fault isolation compared to traditional monolithic applications [13].

To fully grasp the essence of microservices and the transformative impact they can have on organizations, it is essential to familiarize oneself with their fundamental characteristics. These are:

1) Independent deployability;

2) Loosely coupled;

3) Organized around specific functionalities or business domains;

4) Goal-oriented nature;

5) Control over their own data or state, a principle also referred to as encapsulation;

6) Flexibility;

7) Alignment between architecture and organizational structure [12] [14].

2.2. Interdependent Deployability

The independence of microservices allows them to be developed, deployed, and scaled independently by different teams, which enhances an organization's ability to adapt to changes quickly [7]. This independence also supports a more resilient and flexible system architecture, as issues in one service can be contained and addressed without impacting others. Therefore, while the services themselves can be quite robust and complex, the emphasis on doing a specific, limited set of tasks aligns with the principle of reducing complexity through division and specialization [15].

A microservice provides a business or platform capability through a well-defined API, data contract, and configuration. It provides this function and only this function. It does one thing, and it does it well [16].

2.3. Loosely Coupled

The loosely coupled nature of microservices means that each service operates independently of the others, yet they can work together to form a comprehensive application. This separation reduces dependencies, which in turn minimizes the risk of cascading failures across services. It enables developers to manage and up-

date services more efficiently, promoting robust system architecture [2] [8].

2.4. Organized around Specific Functionalities or Business Domains

Microservices are organized around specific functionalities or business domains, aligning the service structure directly with business needs. This organizational method ensures that each microservice is focused on a single business capability, enhancing both the functional clarity and the effectiveness of the development teams responsible for different services [2].

2.5. Control over Their Own Data or State

Each microservice manages its own data or state, a practice known as encapsulation. This autonomy prevents data conflicts and ensures that the service's performance is optimized for its specific tasks. Encapsulation supports the integrity and independence of microservices, facilitating more secure and stable operations. Additionally, this promotes backwards compatibility as limiting shared data prevents issues created from updates in different service boundaries [2].

2.6. Flexibility

Since microservices are designed to be independent, loosely coupled, encapsulated, and organized around a business function they inherently provide flexibility [2]. As mentioned, it allows each team to make their own changes without impacting the other if API contracts are not being changed. With this type of flexibility, it creates adaptability as well as composability in the organization to quickly introduce changes minimizing impact to other services if data contracts are not changing. Additionally, microservices offer technological flexibility, enabling organizations to adopt the latest technologies and gain competitive advantages [3]

Each microservice can be implemented using various programming languages and storage methods. This flexibility allows teams to select the most suitable technologies for each specific service [17]. It is important for an organization to establish a framework or set standards to effectively manage a diverse technology stack. This structured approach helps ensure compatibility, maintainability, and security across different services and technologies, facilitating smoother operations and easier integration of new components as needed.

2.7. Alignment between Architecture and Organizational Structure

The architecture of microservices is often aligned with the organizational structure, embodying the principle of Conway's Law, which posits that system designs mirror the communication structures of the organizations that create them. This alignment helps in minimizing communication overhead and enhances coordination among teams, leading to more efficient and effective development cycles [2]. These traits allow the software to be flexible, agile, easily changed, and frequent and reliable delivery for large organizations [6].

3. Monolithic vs Microservice

Monolithic architecture is a traditional model for designing software applications where software is built as in all-in-one package. Where changing one requires the entire application to be rebuilt and deployed. What this means is the code base is singular which inherently couples all business logic together. "With monoliths, applications are developed in entire blocks that communicate internally, manage their data usually in a single database, and each new feature demands the deployment of the application as a whole" [3].

This software architecture was commonly used and is still today as it can be easier to set up, manage, and deploy. Reference [3] identifies several advantages of monolithic architecture, particularly concerning the size and complexity of the software:

1) Easier Deployment: The deployment process involves a single application packaged as a unified file or directory.

2) Simpler Initial Development: Development is in one code repository, more often with only one programming language. Note, there is software that uses one code repository with more than one language. This is due to interoperability with the compiler of languages to the same Common Intermediate Language (CIL). Example, of this is Dynamics 365 Finance and Operations where both X++ and C# are in the same repository [18]. This is also known as polyglot programming and allows developers to leverage the benefits of different programming languages [19]. This approach can be used in microservices.

Despite the initial advantages of monolithic, it has been found often that overtime monolithic software grows quite large and complex. Having a large software base with all business logic coupled together in single base presents challenges when needing to rapidly innovate, as changes are slower to introduce. As mentioned, monolithic architecture has a high coupling but also, they have low cohesion [11]. Cohesion is the degree to which the elements inside a module or software application are grouped together [14]. Low cohesion means the applications encompass a wide range of functionalities within the single code base. One could argue that grouping code a specific way could prevent low cohesion in monolithic but at the end of the day all the functionality is in the same singular application resulting in down time of all functionalities for one change and shared resources. As the development timeline progresses, numerous developers contribute new features and functionalities to a single codebase. This accumulation often results in highly coupled software with low cohesion, complicating the introduction of continuous changes and making the codebase less transparent and more challenging to maintain.

With this increased complexity there also is a higher risk of introducing unintentional behavior known as regressions [20]. As more changes are introduced into the single codebase, the risk of regressions rises. Over time, this complexity slows down the pace of changes and reduces the software's adaptability and flexibility. The illustration created by Andrew Ganje below demonstrates that, in a

monolithic design, all functionalities are integrated into a single system. In contrast, the microservice approach divides these functionalities into three distinct silos, ensuring each service has a singular purpose (Figure 1).

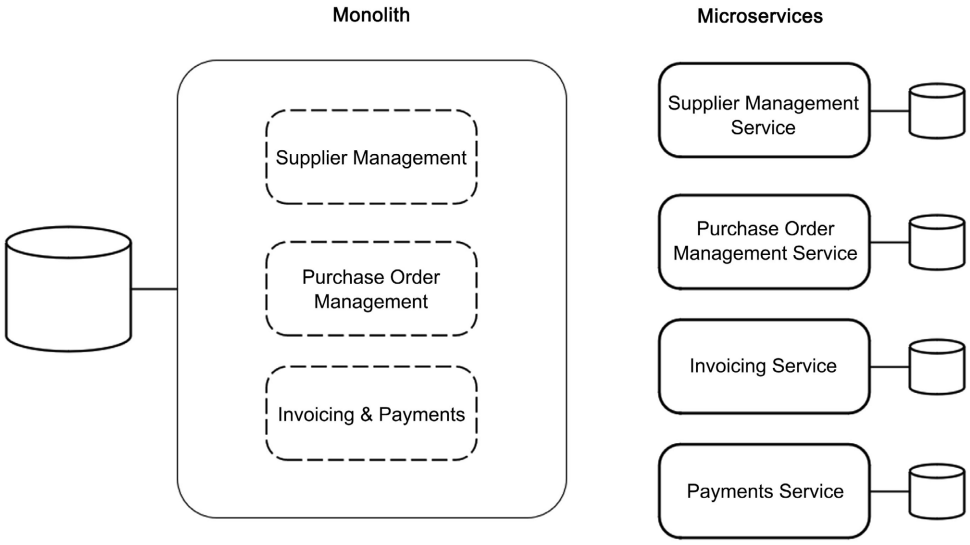

Figure 1. Monolith vs Microservices.

The diagram illustrates how a microservice architecture can simplify systems that have grown overly large and intricate by segmenting their functionality into distinct services. As systems expand in size and complexity, leading to challenges, the adoption of microservices becomes an attractive solution. This is why its popularity has surged among organizations facing issues with their expansive software systems [13]. Note, as you read these concepts reference this article to see the architecture paradigms and benefits and cons are supported visually by the architecture diagram. This diagram is similar used in other popular resources in different variations but still depicts decoupled vs coupled mono architecture.

Furthermore, microservices enhance composability, which refers to the ability to interchange components without impacting other parts of the system. This level of modularity allows for independent updates and maintenance, a feature that is not feasible with monolithic architecture. In a monolithic system, making changes often requires taking down the entire application, and dependencies between components must be carefully managed. Composable software architectures like microservices thus promote greater flexibility and agility, enabling organizations to adapt more quickly to changing requirements and technologies.

Microservices in summary are a software development approach that creates a system with isolated services that each have a goal or function that can be easily changed, deployed, and maintained. These services are widely popular with cloud providers due to the ability to easily deploy them on technologies such as Microsoft Azure or Amazon AWS [4] [5]. Monolithic on the other hand is a software architecture where all business logic is coupled together in a singular code base.

4. Strategic Considerations in Microservices

Microservices architecture continues to gain global traction as a pivotal approach in software development. Its ability to address modern challenges of agility and adaptability aligns perfectly with the interconnected nature of today's world. As global trade, supply chains, and communication networks become more intertwined, businesses face increasing vulnerabilities to disruptions. Tensions between countries, trade wars, and other geopolitical issues can create a domino effect, impacting industries far removed from the initial point of conflict. These challenges necessitate systems capable of adapting quickly, operating efficiently, and innovating at speed. Microservices, with their modular design and agility, provide organizations with the tools to remain resilient amidst such uncertainties.

4.1. Benefits Driving Microservices Adoption

In my professional experience developing cloud business applications and integrations, microservices have emerged as a critical enabler for both large and small organizations. They allow companies to enhance operational efficiency, foster innovation, and prepare for disruptions by enabling independent updates, scalability, and composability. Furthermore, in the era of big data, microservices integrate seamlessly into data pipelines to perform specialized tasks, such as fraud detection, traffic management, and medical research [21]. The adoption of cloud computing and the need for organizations to meet disruptions and changes quickly only strengthen the upward trajectory of microservices adoption. Reference [15] highlights that microservices represent a significant evolution in software architecture, offering numerous advantages over traditional monolithic approaches.

4.2. Challenges and Limitations of Microservices

While the benefits of microservices are clear, it is equally important to consider their limitations to ensure they are the right fit for a given context. Adopting microservices comes with inherent challenges:

1) Increased Complexity

Managing distributed systems requires sophisticated tools and processes, such as container orchestration platforms (e.g., Kubernetes) and monitoring frameworks (e.g., Prometheus). This operational complexity can overwhelm organizations lacking the necessary expertise [2].

2) Higher Costs

The infrastructure and maintenance costs associated with microservices are often significantly higher than those of monolithic architectures, particularly for smaller organizations or applications [10].

3) Data Integrity and Consistency

Distributed databases introduce challenges in maintaining data consistency, especially in scenarios requiring transactional guarantees. Approaches like eventual consistency require careful consideration and trade-offs [22].

4) Dependency on Organizational Structure

Microservices work best when the architecture aligns with a decentralized, domain-focused team structure. Organizations with rigid or siloed structures may struggle to implement microservices effectively [2].

5) Latency and Reliability

Communication between services over a network introduces latency and potential failure points. These challenges require additional fault tolerance mechanisms, adding to system complexity [15].

4.3. Evaluation Metrics for Choosing Architectural Approaches

To navigate the complexities of modern software architecture, organizations must evaluate whether microservices align with their goals. The following metrics can guide decision-making:

1) Scalability Needs

If specific components of the system require independent scaling, microservices are likely the better choice. Monolithic architecture may suffice for simpler or smaller systems.

2) Operational Complexity Tolerance

Organizations must assess whether they have the tools, expertise, and processes in place to handle the complexities of distributed systems.

3) Deployment Frequency

Microservices are ideal for environments where frequent updates and releases are critical to maintaining competitiveness [7].

4) Resilience and Fault Tolerance Requirements

If high availability and fault isolation are priorities, microservices provide a significant advantage due to their modular design [12] [15].

5) Team Readiness and Structure

Microservices require cross-functional, autonomous teams capable of owning the full lifecycle of a service. Without such a structure, adopting microservices may lead to inefficiencies.

6) Time-to-Market Priorities

Microservices enable rapid feature development and deployment, which is crucial in highly competitive industries [22].

7) Cost-Benefit Analysis

The potential benefits of microservices should be weighed against the costs of implementation and maintenance, particularly for organizations with limited resources [10].

4.4. Balancing the Benefits and Drawbacks

While microservices offer unparalleled flexibility, scalability, and resilience, they are not a universal solution. Monolithic architecture still holds merit for small applications or organizations that prioritize simplicity and cost-efficiency. Serverless architecture, meanwhile, offers an alternative for workloads with sporadic demand, minimizing infrastructure management while maximizing resource uti-

lization. Selecting the right architecture requires a nuanced understanding of the system's requirements, operational context, and organizational capabilities.

4.5. The Road Ahead

Microservices are well-positioned to remain a dominant force in software development due to their alignment with trends like cloud computing, big data, and the increasing need for rapid adaptation. However, their adoption must be strategic, weighing the benefits of agility and modularity against the complexities and costs they introduce. As organizations face ever-changing geopolitical, economic, and technological challenges, the choice of software architecture will play a pivotal role in determining their ability to thrive in an interconnected world.

By critically evaluating architectural approaches using metrics and understanding the trade-offs involved, organizations can ensure that their systems are not only resilient but also capable of driving innovation and sustaining competitive advantage.

5. Conclusions

In today's dynamic and interconnected global landscape, the demand for agile, adaptable, and swiftly deployable software systems has never been greater. As reference [23] emphasizes, adopting microservices brings scalability, flexibility, and agility to software development, which are crucial for staying ahead in a competitive business environment. Microservices have emerged as a transformative architectural paradigm, enabling organizations to achieve the flexibility, scalability, and resilience required to navigate challenges ranging from geopolitical tensions to technological disruptions.

Microservices architecture, characterized by its independent deployability, loose coupling, and alignment with specific business functionalities, offers significant advantages over traditional monolithic systems. By facilitating modularity and autonomy, microservices enable faster development cycles, enhanced fault tolerance, and the ability to scale components independently [7]. This architectural shift empowers organizations to innovate and adapt with greater speed, positioning them to meet evolving market demands effectively.

However, while the benefits of microservices are compelling, their adoption requires a nuanced approach. Transitioning from monolithic architecture involves not only technical refactoring but also cultural and organizational transformation. These shifts demand meticulous planning and significant investment, particularly for smaller organizations that may lack the necessary resources or expertise [22]. Additionally, the increased complexity of managing distributed systems introduces operational challenges that must be carefully addressed to ensure long-term success [10].

Despite these challenges, the upward trajectory of adoption of microservices reflects their alignment with global trends such as the rise of cloud computing, big data, and the demand for rapid adaptation. As [21] notes, microservices are par-

ticularly well-suited for big data applications, where their modularity and scalability offer unparalleled advantages in managing large-scale, complex workloads.

Looking ahead, microservices are expected to play a pivotal role in shaping the future of software development. Their ability to enable composable and resilient architecture makes them an essential tool for organizations striving to achieve operational efficiency, foster innovation, and build resilience in an increasingly digital and interconnected world. By embracing microservices, organizations can not only address the challenges of today but also position themselves for sustained success in the evolving technological landscape.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Kaloudis, M. (2024) Evolving Software Architectures from Monolithic Systems to Resilient Microservices: Best Practices, Challenges and Future Trends. *International Journal of Advanced Computer Science and Applications*, **15**, 1-10. https://doi.org/10.14569/IJACSA.2024.0150901

[2] Newman, S. (2021) Building Microservices: Designing Fine-Grained Systems. 2nd Edition, O'Reilly Media.

[3] Krug, D.S., Chanin, R. and Sales, A. (2024) Exploring the Pros and Cons of Monolithic Applications versus Microservices. *Proceedings of the* 19*th International Conference on Evaluation of Novel Approaches to Software Engineering* (*ENASE*), **19**, 123-130.

[4] Amazon (2021) Microservices. https://aws.amazon.com/microservices/

[5] Microsoft (2021) Introduction to Microservices on Azure-Azure Service Fabric. https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-overview-microservices

[6] Newman, S. (2021) Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. O'Reilly Media.

[7] Samant, P.S. (2023) Microservices in the Cloud: Enabling Scalability, Flexibility, and Rapid Deployment. *Journal of Advanced Research in Engineering and Technology*, **3**, 1-10. https://www.researchgate.net/publication/381306736_MICROSERVICES_IN_THE_CLOUD_ENABLING_SCALABILITY_FLEXIBILITY_AND_RAPID_DEPLOYMENT

[8] Prabhakar, G. (2025) Microservices Architecture. *International Research Journal of Modernization in Engineering, Technology and Science*, **7**, 3296-3307.

[9] Vera-Rivera, F.H., Gaona, C. and Astudillo, H. (2021) Defining and Measuring Microservice Granularity—A Literature Overview. *PeerJ Computer Science*, **7**, e695. https://doi.org/10.7717/peerj-cs.695

[10] Richardson, C. (2018) Microservices Patterns: With Examples in Java. Manning Publications.

[11] Singh, N.P. and Deshpande, A. (2021) Challenges and Patterns for Modernizing a Monolithic Application into Microservices. https://developer.ibm.com/articles/challenges-and-patterns-for-modernizing-a-monolithic-application-into-microservices/

[12] Vemasani, P. and Modi, S. (2024) Building Resilient Distributed Systems: Fault-Tolerant Design Patterns for Stateful Workflows. *International Journal of Computer Engineering and Technology*, **15**, 169-181. https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_15_ISSUE_3/IJCET_15_03_016.pdf

[13] Amundsen, M. (2016) Microservice Architecture. Sebastopol, O'Reilly Media, Inc.

[14] Pagade, G. (2022) Difference between Cohesion and Coupling. Baeldung on Computer Science. https://www.baeldung.com/cs/cohesion-vs-coupling

[15] Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R. and Safina, L. (2017) Microservices: Yesterday, Today, and Tomorrow. In: Mazzara, M. and Meyer, B., Eds., *Present and Ulterior Software Engineering*, Springer, 195-216. https://doi.org/10.1007/978-3-319-67425-4_12

[16] Familiar, B. (2015) What Is a Microservice? In: *Microservices*, *IoT*, *and Azure*, Apress, 9-19. https://doi.org/10.1007/978-1-4842-1275-2_2

[17] Baddula, P. (2023) The Evolution of Software Architecture: Monolithic to Microservices. Medium. https://medium.com/@phanindra208/the-evolution-of-software-architecture-monolithic-to-microservices-cb62fcd7aa94#:~:text=The%20conventional%20Monolithic%20architecture%2C%20a%20unified%20model%20for,the%20advent%20of%20cloud%20computing%20and%20containerization%20technologies

[18] Microsoft. (2022) X++ Language Reference. Finance & Operations: Dynamics 365, Microsoft Learn. https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/dev-ref/xpp-language-reference

[19] Neward, T. (2009) The Polyglot Programmer-Mixing and Matching Languages. Microsoft Learn. https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/march/the-polyglot-programmer-mixing-and-matching-languages

[20] Newth, A. (2021) What Is Software Regression? https://www.easytechjunkie.com/what-is-software-regression.htm

[21] Vigliarolo, B. (2020) Microservices: A Cheat Sheet. https://www.techrepublic.com/article/microservices-a-cheat-sheet/

[22] Fowler, M. and Lewis, J. (2014) Microservices: A Definition of This New Architectural Term. https://martinfowler.com/articles/microservices.html

[23] Kothapalli, M. (2021) Securing Microservices Architecture: Best Practices and Challenges. *Journal of Scientific and Engineering Research*, **8**, 187-192.