Architecture

UDC 72:65

Mozolevskyi Dmytro
Full stack software engineer

Roilian Mykyta
Senior Software Engineer

Terletska Khrystyna
Senior Software Engineer

THE IMPACT OF ARCHITECTURAL DECISIONS ON THE LONG-TERM PERFORMANCE OF ENTERPRISE SYSTEMS

Summary. *This article examines the impact of architectural decisions on the long-term effectiveness of enterprise systems. It analyzes key aspects of architectural design, such as modularity, scalability, extensibility, and maintainability. Based on research and practical examples, it demonstrates how architectural choices affect the performance, cost of ownership, and adaptability of systems in the face of rapidly changing business requirements. Recommendations for designing architectures that contribute to improving the long-term effectiveness of enterprise systems are offered. Particular attention is paid to examples from the practice of Netflix and Airbnb, which have successfully implemented modern architectural approaches.*

Key words: *architectural decisions, effectiveness of enterprise systems, architectural design, Netflix, Airbnb.*

Introduction. Modern enterprise systems are complex hardware and software systems designed to automate business processes and support decision-making. The effectiveness of such systems is determined not only by their functionality, but also by the architecture, which is laid down at the design stage. Architectural decisions have a significant impact on the long-term effectiveness of the system, including such aspects as performance, scalability, maintenance costs, and the ability to adapt to changes.

The purpose of this article is to examine how different architectural approaches affect the long-term effectiveness of enterprise systems and offer recommendations for choosing the best solutions. Particular attention is paid to the types of architectures and their characteristics that determine the success of the system in the long term. Practical examples from the experience of Netflix and Airbnb are also considered, demonstrating how the right choice of architecture can lead to a significant improvement in business efficiency.

Key aspects of architectural decisions. Architectural decisions have a significant impact on the long-term effectiveness of enterprise systems. Let's consider the key aspects that need to be taken into account when designing an architecture.

Modularity. Modularity involves dividing the system into independent components that can be developed, tested, and deployed separately. This simplifies system support and upgrades, and reduces development complexity. Modular architecture allows you to update individual components without having to rebuild the entire system, which reduces the cost and time of making changes. For example, microservice architecture is a prime example of a modular approach that provides high extensibility and adaptability.

Scalability. Scalability refers to a system's ability to handle increased load efficiently. Both monolithic and microservices architectures can be scaled vertically by increasing the capacity of existing servers. However, only architectures designed for distributed workloads, like microservices, can effectively scale horizontally by adding new servers. Horizontal scaling is crucial for systems handling large numbers of users or data, as it allows for better load distribution and fault tolerance. Microservices architectures, particularly in cloud environments, excel in horizontal scalability, enabling dynamic resource allocation based on demand. In contrast, monolithic architectures often face limitations in horizontal scaling due to tightly coupled components.

Extensibility. Extensible architecture is a design paradigm that enables a system to accommodate new functionalities, technologies, and optimizations with minimal disruption to existing components. It is achieved through modularity, loose coupling, and well-defined interfaces, allowing independent evolution of system elements. Extensibility is often supported by architectural patterns such as abstraction layers, plug-in mechanisms, event-driven communication, and standardized APIs, ensuring seamless integration of new capabilities without requiring extensive rework.

Maintainability. Maintainability in software architecture ensures efficient updates, modifications, and optimizations as technologies and business processes evolve. It minimizes technical debt, reduces obsolescence risk, and lowers long-term costs.

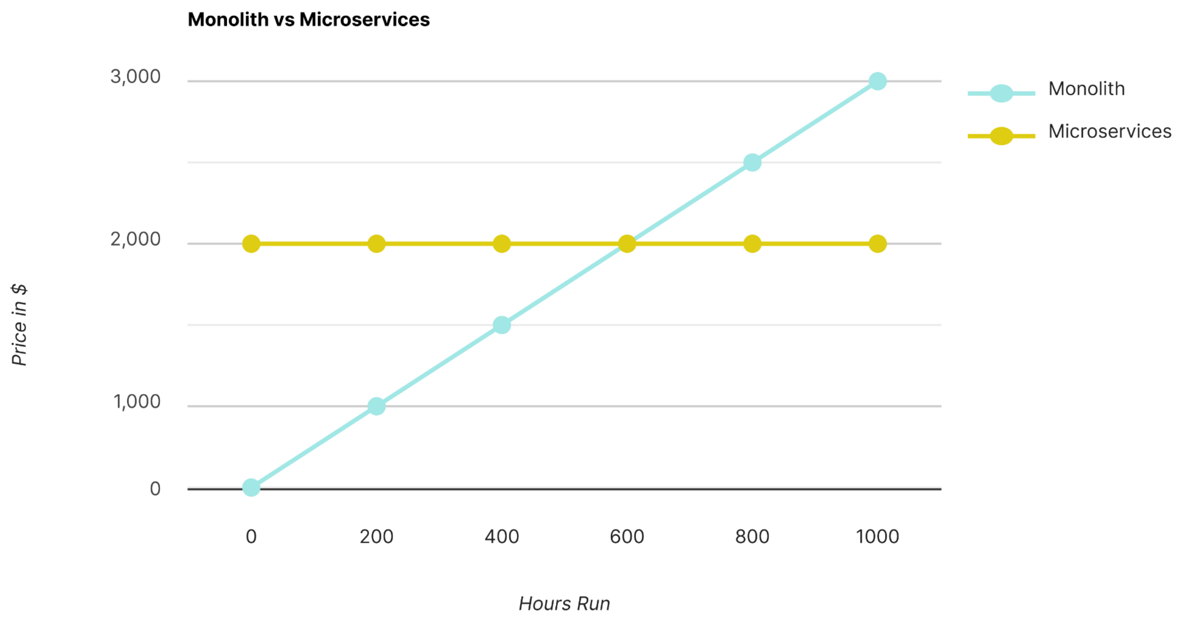
Key factors include clear code structure, consistent documentation, and adherence to coding standards. Automated testing, CI/CD pipelines, and static analysis tools streamline updates and prevent regressions.

At the architectural level, maintainability is reinforced through versioned APIs, well-defined service contracts, and observability mechanisms (logging, monitoring, tracing), ensuring system reliability, stability, and ease of management.

Types of architectures and their impact on efficiency. The architecture of a corporate system is the foundation that determines its long-term efficiency. There are several main types of architectures, each with its own characteristics, advantages and disadvantages. Let's consider them in more detail.

Monolithic Architecture. A monolithic architecture is a single, tightly coupled system where all components interact within a single process. This approach is easy to develop and deploy initially, making it popular for small projects. However, as the system grows, a monolithic architecture becomes less efficient. The main problems include difficulty scaling, high cost of making changes, and low fault tolerance. For example, adding a new feature may require rebuilding the entire system, which increases time and costs. In the long term, a monolithic architecture can become an obstacle to system evolution, especially in the face of rapidly changing business requirements.

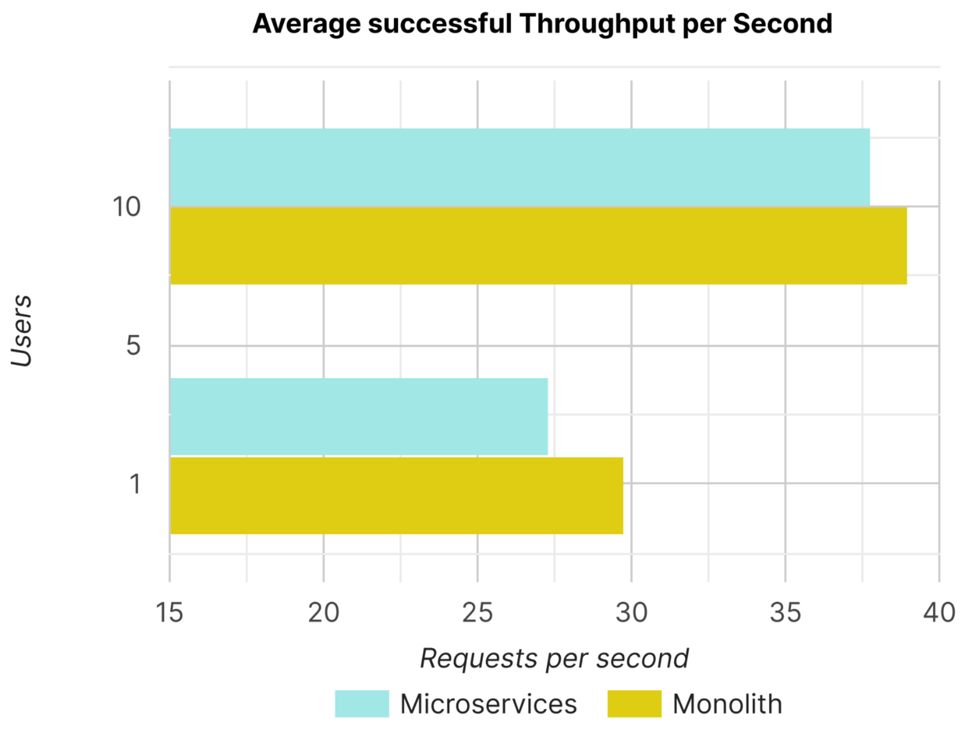
Microservices Architecture. Microservices architecture involves dividing a system into multiple independent, loosely coupled services, each of which performs a specific function. This approach provides high extensibility, scalability, and fault tolerance. Microservices allow you to develop, test, and deploy components independently, which speeds up the process of introducing new features. However, microservices architecture requires more complex infrastructure and management, which can increase the cost of development and operation. For example, Netflix successfully switched to a microservices architecture, which allowed the company to significantly increase the scalability and fault tolerance of the system. In the long term, microservices provide high system adaptability, which makes them an ideal choice for dynamically developing companies.



Service-oriented architecture (SOA). Service-Oriented Architecture (SOA) sits between monolithic and microservice architectures, offering modularity and reusability. It enables standardized communication through SOAP, REST, or messaging but relies on an Enterprise Service Bus (ESB), which can introduce bottlenecks. Unlike monoliths, SOA improves extensibility and legacy system integration but lacks the fine-grained scalability of microservices. Compared to microservices, SOA services are larger, often stateful, and have slower deployment cycles. It works best for enterprise systems that prioritize stability, strong consistency, and interoperability over agility and rapid scaling.

Cloud vs. On-Premises Architectures. Cloud architecture involves using cloud technologies to deploy and manage enterprise systems. This approach provides high scalability, extensibility, and reduced infrastructure costs. Cloud architecture allows you to quickly adapt to changes in load and implement new features. However, it requires careful planning and management to avoid security and performance issues. In the long term, cloud architecture provides high efficiency and reduced operating costs. For example, Airbnb successfully uses cloud architecture to scale its platform, allowing the company to quickly adapt to changes in demand.

On-premises architecture involves hosting the system on the company's own servers. This approach provides full control over the infrastructure and data, which is important for companies with high security requirements. However, on-premises architecture requires significant maintenance costs and is less efficient in the face of changing loads.



The Impact of Architectural Decisions on Efficiency. Architectural decisions have a direct impact on such aspects of the long-term efficiency of enterprise systems as performance, cost of ownership, adaptability, and reliability. For example, a microservices architecture allows you to achieve high performance and fault tolerance, but requires significant management costs. A monolithic architecture, on the contrary, is easy to develop, but can become an obstacle to scaling and adaptation.

Netflix Example. One of the most striking examples of a successful transition to a microservices architecture is Netflix. Netflix initially used a monolithic architecture, which became inefficient as the number of users and the load on the system increased. In 2009, the company began the process of migrating to a microservices architecture, which allowed it to significantly increase the scalability and fault tolerance of the system. Each microservice is responsible for a specific function, such as movie recommendations or subscription management. This allowed Netflix to quickly introduce new features and improve the user experience. In addition, the microservices architecture ensured high fault tolerance: if one service fails, the others continue to work, which minimizes downtime.

Airbnb Example. Airbnb is also an example of the successful use of modern architectural approaches. Early in its development, Airbnb used a monolithic architecture, which began to limit the platform's growth as the number of users increased and business processes became more complex. To solve this problem, the company switched to a microservices architecture, dividing the platform into independent services such as booking, accommodation search, and payment processing. This allowed Airbnb to develop and deploy new features faster and independently, increasing the agility of the system. This allowed the company to quickly adapt to changes in demand and ensure high system performance even during peak load periods. In addition, the use of cloud technologies has reduced infrastructure costs and simplified system management.

Recommendations for architecture design. Based on the analysis, the following recommendations can be formulated for designing the architecture of enterprise systems. The use of a modular approach allows you to divide the system into independent components, which simplifies its support and modernization. The use of modern technologies, such as microservices, cloud platforms, and containerization, increases the extensibility and scalability of the system. Taking into account future changes in business processes and technologies helps reduce the risk of system obsolescence. Optimizing performance at the architecture design stage ensures high system efficiency in the long term.

Conclusion. Architectural decisions play a key role in ensuring the long-term effectiveness of enterprise systems. The right choice of architecture allows you to increase productivity, reduce the cost of ownership, and ensure the adaptability of the system to changes. Modern approaches such as microservice architecture, cloud technologies, and containerization are important tools for achieving these goals.

The recommendations proposed in the article can be used to design architectures that help improve the long-term effectiveness of enterprise systems. Taking into account key aspects such as modularity, scalability, extensibility, and maintainability allows you to create a system that will function effectively in a rapidly changing business environment.

References

1. Fowler, M. (2015). Microservices: a definition of this new architectural term.
2. Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems.
3. Richardson, C. (2018). Microservices Patterns: With examples in Java.
4. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice.
5. Netflix Tech Blog. Available at: https://netflixtechblog.com (access date: 15.02.2024).

6. Airbnb Engineering & Data Science Blog. Available at: https://medium.com/airbnb-engineering (access date: 15.02.2024).

7. Amazon Web Services (AWS) Case Studies: Netflix and Airbnb. Available at: https://aws.amazon.com/solutions/case-studies/ (access date: 15.02.2024).

8. Martin Fowler's Blog on Event-Driven Architecture. Available at: https://martinfowler.com/articles/201701-event-driven.html (access date: 15.02.2024).

9. Spotify Engineering Blog. Available at: https://engineering.atspotify.com (access date: 15.02.2024).

10. Gartner Research on Cloud Architecture and Corporate Systems. Available at: https://www.gartner.com (access date: 15.02.2024).