Dev.Pro
Dev.Pro Website

Service-Oriented Architecture vs Microservices: What is the Best Systems Integration Approach?

How many ways are there to build a system's architecture? Mark Richards, a guru of software architecture design, suggests in his book "Software Architecture Patterns" that there are at least five: Layered Architecture, Event-Driven Architecture, Service-Oriented Architecture (SOA), Microservices Architecture, and Space-Based Architecture.

There is little available research that points to one of these approaches as the best option. One could say that SOA and Microservices are trendy, or that Microservices can't live up to expectations when compared to SOA. However, research from TechRepublic proves exactly the opposite.

What makes SOA and Microservices the two most popular options today? Who wins in the SOA vs Microservices dilemma?

Over a decade ago, monolithic architectures were trendy in the industry. Software systems have evolved significantly since then. Modern challenges drive new requirements for software architecture design, calling for **more scalable, resilient, and cost-effective solutions**.

For this article, we picked two of the most popular options to compare.

What is Service Oriented Architecture (SOA)?

Service-Oriented Architecture is a software design approach where your system consists of independent services (also known as components), each responsible for a relatively broad business function (like payment processing or website login).

These components are reusable and can be implemented as standalone applications, making them an excellent option for enterprise-grade companies. SOA applications range from retail (Walmart, Best Buy, and McDonald's) to the government sector (the US Air Force).

What is Microservices Architecture?

Microservices follows a similar approach. The main difference is that this solution is better at application level, rather than the enterprise level. The services communicate via an Application Programming Interface (API) to build a single application for a specific business purpose.

Microservices architecture is cloud-native and is used by SaaS giants like Amazon, Netflix, and Uber.

Download SOA Creation Stages and Integration Roadmap
SOA Creation Stages and Integration Roadmap

SOA vs Microservices: Pros and Cons

Here is the catch: It's more appropriate to think about how these services complement each other, rather than how they stand in opposition to each other. They are pretty similar (aside from **the scale**) and follow the same concept (see *Figure 1*).

Figure 1. The differences between types of architecture design.

When it comes to choosing an approach, here are a few facts that illustrate the high-level differences:

- Microservices architecture is generally more scalable and flexible than SOA
- SOA helps handle multiple tasks, while Microservices Architecture is typically built with the Single Responsibility principle in mind
- The primary purpose of Microservices Architecture is to provide a resilient infrastructure and exclude dependencies, while SOA is all about resource sharing;

Now let's talk about each in greater detail.

Service-Oriented Architecture Overview

SOA applications are built on the principle of loose coupling, meaning the SOA

components are designed for minimum dependency on each other. This design approach allows you to use services at a specific capacity even if one of the dependent elements goes down. For example, you could still use Google Docs if one of Google servers goes down. However, some Google Docs features might be temporarily unavailable.

SOA: Pros	SOA: Cons
The reusable components save development time and can be modified or debugged easily.	DevOps practices are not widely adopted yet, so deployment can be complicated.
A wide range of messaging protocols is available.	Software size is significant.
A small range of technologies is used	Strict governance, protocols, and processes are in place.

Table 1. Pros and cons of SOA

All in all, SOA's reusable components are a blessing and a curse. On the one hand, you get a rapid development process. On the other hand, if one of those components fails, issues will likely arise across several systems that share the same component.

SOA services share access to Enterprise Service Bus (ESB), a mechanism that ensures communication between services. It is possible to have SOA without ESB by directly connecting each of the services. That would make development costs significantly higher.

Microservices Architecture

Microservices Architecture employs the idea of loose coupling, as well. However, there can be many more interconnected services. A microservice system could easily feature over a dozen services as opposed to only three or four in SOA. However, this also means Microservices Architecture can be tricky when it comes to Operational Complexity.

Different services can be developed separately by other development teams. That means more servers, more data processing, and more potential inconsistency and connectivity issues. That's another reason why DevOps is a natural combination with Microservices Architecture.

Microservice Architecture: Pros	**Microservice Architecture: Cons**
Deployment is easy, DevOps practices are broadly used.	Testing can be difficult and maintenance can be expensive.
Software size is small.	Only HTTP/REST, JMS, and Thrift APIs are typically used as messaging protocols.
Relaxed governance, protocols, and practices.	A large range of technologies used

Table 2. Pros and cons of Microservices Architecture

All things considered, Microservices Architectures is a popular choice. A recent study

by IBM found that at least 56% of surveyed enterprises are likely to adopt Microservices in the next several years.

When to Use Service-Oriented Architecture

SOA is a good option **for large and diverse environments** that generally have more than one application. As far as multiple company departments are concerned, SOA can help with reusable business processes. Heterogeneous applications and various messaging protocols can be successfully handled by Enterprise Service Bus, while strict governance can facilitate and streamline standard procedures. At the same time, it can take fewer technologies to build SOA, meaning more occasional maintenance.

SOA vs Microservices: When to Go for Microservices

Implementing Microservices has several pros and cons. Microservices is a rather complex architecture, and it should only be used when it's essential to the application.

Microservices allows for a great deal of development flexibility. If you're unsure which direction the application development will take, Microservices may be the correct option. That's because each component can be removed, modified, or added painlessly (horizontal scaling). Its cloud-native approach allows for easy deployment and a simplified collaboration process between development team members. This makes Microservices a great approach **for a single yet complex application**.

Some common Microservices frameworks are moleculer (NodeJS), microdot (.NET), nameko (Python), rackstep (Ruby), and Spark (Java).

SOA vs Microservices: Which one Should you Choose?

The bottom line is that both approaches could be a significant first step for monolithic application migration. One approach doesn't necessarily exclude the other — there are instances where both architecture solutions are used within the same project.

If you still have questions, don't hesitate to contact the experts at Dev.Pro. We can offer advice and share our vision for your integration approach.