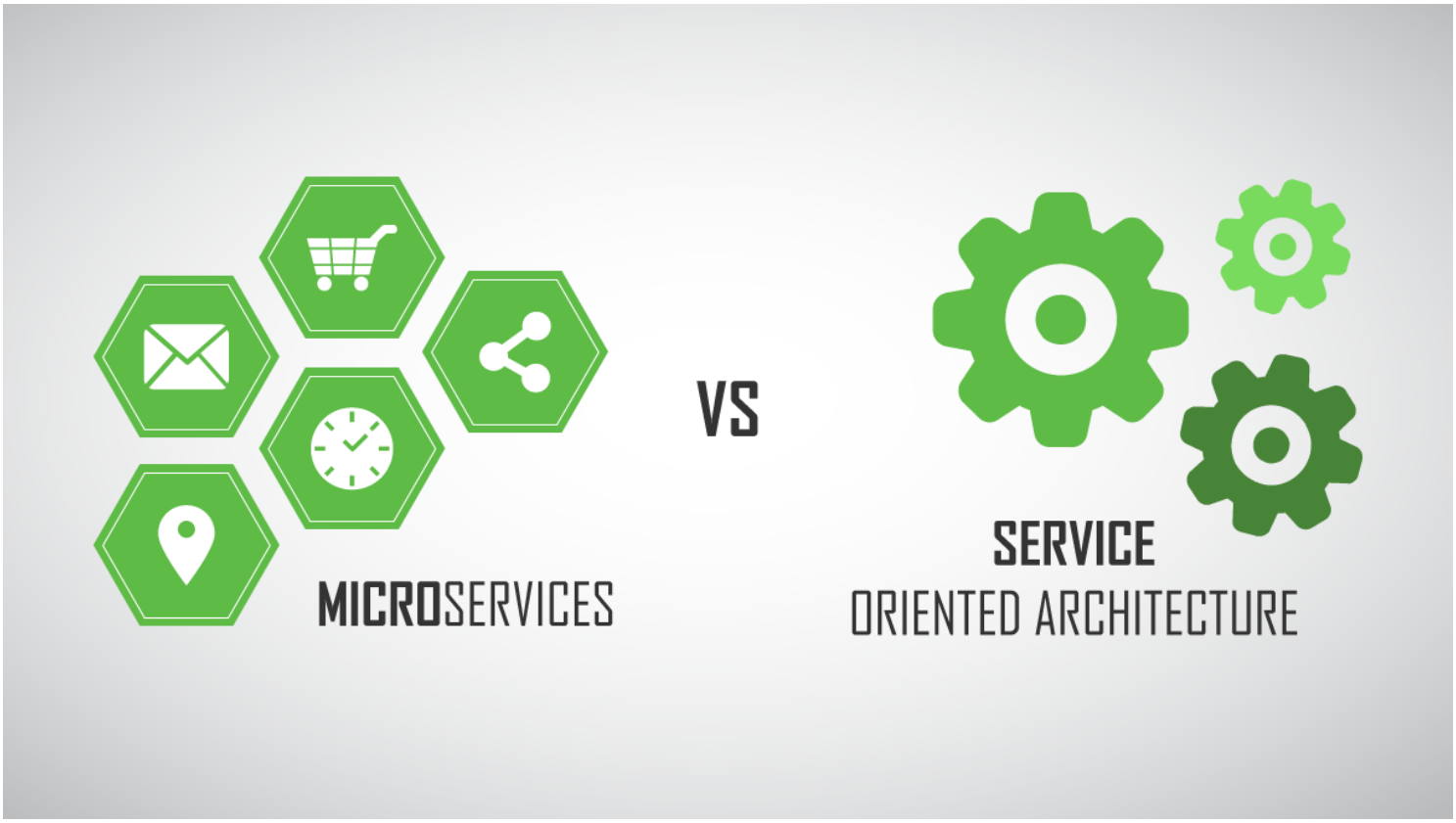
Microservices vs. SOA — Is There Any Difference at All?

Kristijan Arsov Follow
Nov 10, 2017 · 6 min read



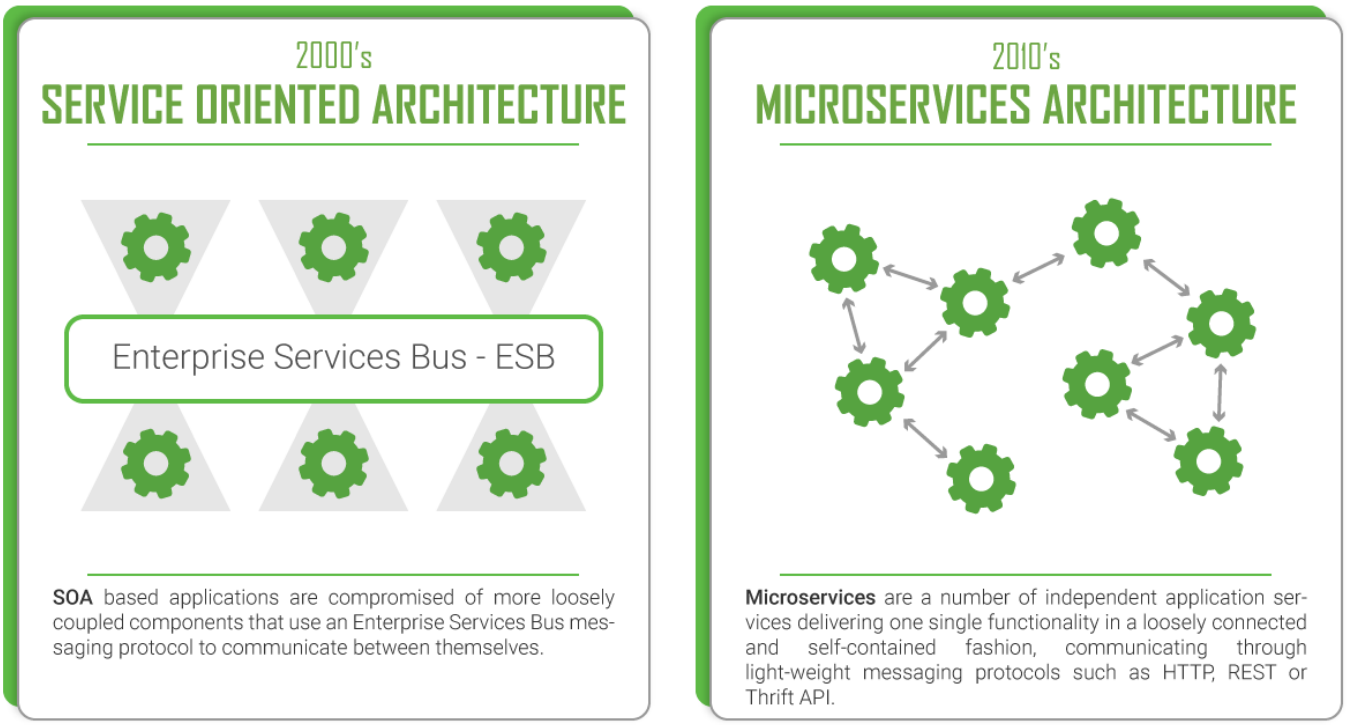
Lately, there has been a lot of fuss about the differences between these 2 types of architectures, or whether there is any difference at all. In order to delve deeper into this question that raised hundreds of debates, I shall first briefly define both SOA and Microservices architecture and their origins, and then we'll compare them and see how we can best distinguish them.

Service-Oriented Architecture (SOA)

Service Oriented Architecture is a software architecture, where distinct components of the application provide services to other components via a communications protocol over a network. The communication can involve either simple data passing or it could involve two or more services coordinating connecting services to each other. These distinct services carry out some small functions, such as validating payment, creating a user account, or providing social log-in.

Service-oriented Architecture is less about how to modularize an application, and more about how to compose an application by integration of distributed, separately-maintained and deployed software components. It is enabled by technologies and standards that make it easier for components to communicate and cooperate over a network, especially an IP network.

There are 2 main roles in SOA, a service provider, and a service consumer. A software agent may play both roles. The *Consumer Layer* is the point where users (human, other components of the app or third parties) interact with the SOA and the *Provider Layer* consists of all the services within the SOA.



Differences in communication between the 2 architectures

SOA first gets its name in the mid 90's, when a corporation called *Gartner Group* recognizes this emerging trend in software architectures, adopts it and popularize it worldwide. By doing this, they managed to greatly speed up the adoption and further evolution of this architecture pattern. However, the first records of using distributed services as software architecture date back to the early 80's.

Microservices

Microservices, in a way, are the next step in the evolution of Service-Oriented Architectures. Basically, this architecture type is a particular way of developing software, web or mobile applications as suites of independent services — a.k.a *microservices*. These services are created to serve only one specific business function, such as User Management, User Roles, E-commerce Cart, Search Engine, Social Media Logins etc. Furthermore, they are completely independent of each other, meaning they can be written in different programming languages and use different databases. Centralized services management is almost non-existent and the microservices use lightweight HTTP, REST or Thrift APIs for communicating between themselves.

The word itself originates from a workshop of software architects held near Venice in May 2011. They used the term "microservice" for the first time, to describe what the participants saw as a common architectural style that many of them had been recently exploring. In May 2012, the same group decided on "microservices" as the most appropriate name. However, major tech companies such as Microsoft, Amazon, Netflix, and Facebook, have been working with microservices for more than a decade. Other people have called them "Micro Web-Services" or "fine-grained SOA", before coming up with the commonly accepted name.

Yes, at first glance, it might seem that we are talking about the same thing as SOA. However, I will call upon the words of Martin Fowler, a pioneer in the world of microservices, who once said that we should think about SOA as a superset of microservices.

So, Where's The Difference?

We could say that both architectures have more similarities than differences, however, at the end of the day, they are 2 different types of architecture. To support my claim, first, I

will go through particular segments of the architectures which differentiate them, in a table form. Later, I will elaborate some of them in greater detail. Here we go!

SERVICE ORIENTED ARCHITECTURE	MICROSERVICES ARCHITECTURE
Maximizes application service reusability	Focused on decoupling
A systematic change requires modifying the monolith	A systematic change is to create a new service
DevOps and Continuous Delivery are becoming popular, but are not mainstream	Strong focus on DevOps and Continuous Delivery
Focused on business functionality reuse	More importance on the concept of "bounded context"
For communication it uses Enterprise Service Bus (ESB)	For communication uses less elaborate and simple messaging systems
Supports multiple message protocols	Uses lightweight protocols such as HTTP, REST or Thrift APIs
Use of a common platform for all services deployed to it	Application Servers are not really used, it's common to use cloud platforms
Use of containers (such as Docker) is less popular	Use of containers (such as Docker) is less popular
SOA services share the data storage	Each microservice can have an independent data storage
Common governance and standards	Relaxed governance, with greater focus on teams collaboration and freedom of choice

I'll get into more detail in some of the aspects shown in the table above and further explain the differences:

- **Development** — In both architectures, services can be developed in different programming languages and tools, which brings technology diversity into the development team. The development can be organized within multiple teams, however, in SOA, each team needs to know about the common communication mechanism. On the other hand, with microservices, the services can operate and be

deployed independently of other services. So, it is easier to deploy new versions of microservices frequently or scale a service independently.
You can read further about these benefits of microservices here.

- **"Bounded Context"** — SOA encourages sharing of components, whereas microservices try to minimize on sharing through "bounded context." A bounded context refers to the coupling of a component and its data as a single unit with minimal dependencies. As SOA relies on multiple services to fulfill a business request, systems built on SOA are likely to be slower than microservices.

- **Communication** — In SOA, the ESB could become a single point of failure which impacts the entire system. Since every service is communicating through the ESB, if one of the services slows down, it could clog up the ESB with requests for that service. On the other hand, microservices are much better in error tolerance. For example, if one microservice has a memory fault, then only that microservice will be affected. All the other microservices will continue to handle requests regularly.

- **Interoperability** — SOA promotes the use of multiple heterogeneous protocols through its messaging middleware component. Microservices attempt to simplify the architecture pattern by reducing the number of choices for integration. So, if you want to integrate several systems using different protocols in a heterogeneous environment, you need to consider SOA. If all your services could be accessed through the same remote access protocol, then microservices are a better option for you.

- **Size** — Last but not least, the main difference between SOA and microservices lies in the size and scope. The prefix "micro" in microservices refers to the granularity of the internal components, meaning they have to be significantly smaller than what SOA tends to be. Service components within microservices generally have a single purpose and they do that one thing really well. On the other hand, in SOA services usually include much more business functionality, and they are often implemented as complete subsystems.

Conclusion

One cannot simply say that one architecture is better than the other. It mainly depends on the purpose of the application you are building. **SOA** is better suited for larger,

complex enterprise application environments that require integration with many other applications. That being said, smaller applications are not a good fit for SOA as they don't need messaging middleware component. **Microservices**, on the other hand, are better suited for smaller and well-partitioned web based systems. Also, if you are developing a mobile or web application, then microservices give you much greater control as a developer. Finally, we can conclude that since they serve different purposes —

> Microservices and SOA are indeed distinct types of architectures.

In any case, if you or your organization decide on implementing a microservices architecture, you should definitely first check this 5 steps guideline on how to successfully prepare for microservices!

. . .

If you want to read further about microservices, here are some of my other articles on the subject:

> *What Are Microservices, Actually?*
>
> *How We Discovered Microservices, Even Before We Knew They Existed*
>
> *Why Should We Migrate To Microservices?*

Microservices Soa Microservice Architecture Cloud Computing Serverless