What are microservices?

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are:

- Independently deployable
- Loosely coupled
- Organized around business capabilities
- Owned by a small team

The microservice architecture enables an organization to deliver large, complex applications rapidly, frequently, reliably and sustainably - a necessity for competing and winning in today's world.

Let's look at why its important to deliver software rapidly, frequently, reliably and sustainably.

Enabling rapid, frequent and reliable software delivery

In order to thrive in today's volatile, uncertain, complex and ambiguous world, businesses must be nimble, agile and innovate faster. Moreover, since modern businesses are powered by software, IT must deliver that software rapidly, frequently and reliably - as measured by the DORA metrics.

Rapid, frequent, reliable and sustainable delivery requires the success triangle, a combination of three things:

- Process - DevOps as defined by the DevOps handbook
- Organization - a network of small, loosely coupled, cross-functional teams
- Architecture - a loosely coupled, testable and deployable architecture

Teams work independently most of the time to produce a stream of small, frequent changes that are tested by an automated deployment pipeline and deployed into production.

Let's now look at when you typically need to use microservices in order to have a loosely coupled, testable and deployable architecture.

When you outgrow your monolithic architecture

Let's imagine that you responsible for a business critical business application that has a monolithic architecture and you are struggling to meet the needs of the business. Should you consider migrating to a microservice architecture? The short answer is that it depends.

It's important to make the most of your monolithic architecture, e.g. adopt DevOps, and reorganize into loosely coupled, small teams.

In many cases, once you have embraced the success triangle, your monolithic architecture is sufficiently loosely coupled, testable and deployable to enable rapid software delivery.

But sometimes an application can outgrow its monolithic architecture and become an obstacle to rapid, frequent and reliable software delivery. This typically happens when the application becomes large and complex and is developed by many teams. For example, its deployment pipeline become a bottleneck. When this occurs, you should consider migrating to microservices.

My presentation *Considering Migrating a Monolith to Microservices? A Dark Energy, Dark Matter Perspective* describes how to decide whether to migrate to microservices.

If you have decided to migrate to microservices then the next step is to design a target architecture.

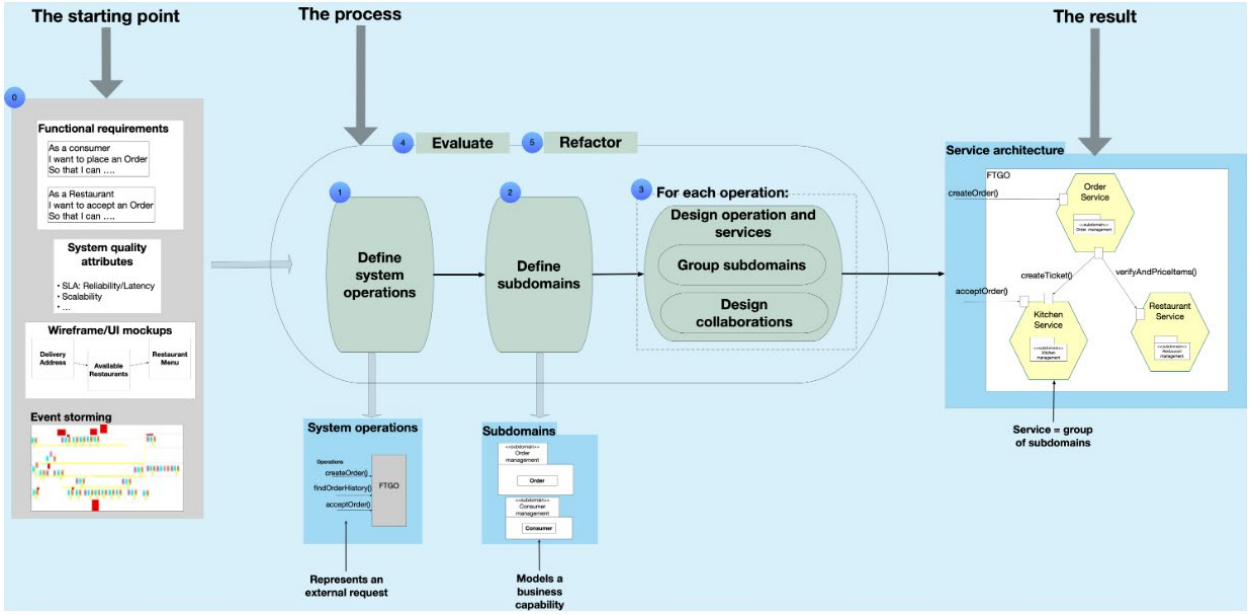
Designing a microservice architecture

Picking technologies - Kubernetes, message broker etc - is important

But what's critically important is designing a good service architecture: identifying services; defining their responsibilities; their APIs and collaborations. If you get it wrong you risk creating a distributed monolith, which will slow down software delivery.

What's more, designing the service architecture is challenging because it's a creative activity - not something you can buy, download or read in a manual.

Assemblage is an architecture definition process that you can can use to define your microservice architecture. It distills your requirements into system operations and subdomains; uses the dark energy and dark matter forces to group the subdomains into services; and designs the distributed system operations.



Assemblage works in conjunction with the Microservice architecture pattern language, which is your guide when designing a technical architecture

After defining a target microservice architecture you then need to refactor your existing monolith.

Migrating from a monolith to microservices

There are numerous principles for migrating a monolithic application to microservices.

One key principle is to incrementally migrate to microservices using the Stranger Fig pattern - no big bang rewrite. By using this pattern, you rapidly validate your design decisions and deliver new, useful functionality much earlier.

It's also important to avoid the Microservices adoption antipatterns.

This text is presented here in case of changed or broken links, so the learner may still have access to the information.