jean pierre Deffo Fotso
46 Followers · About Follow

Microservices Vs SOA Vs Monolithic Approach

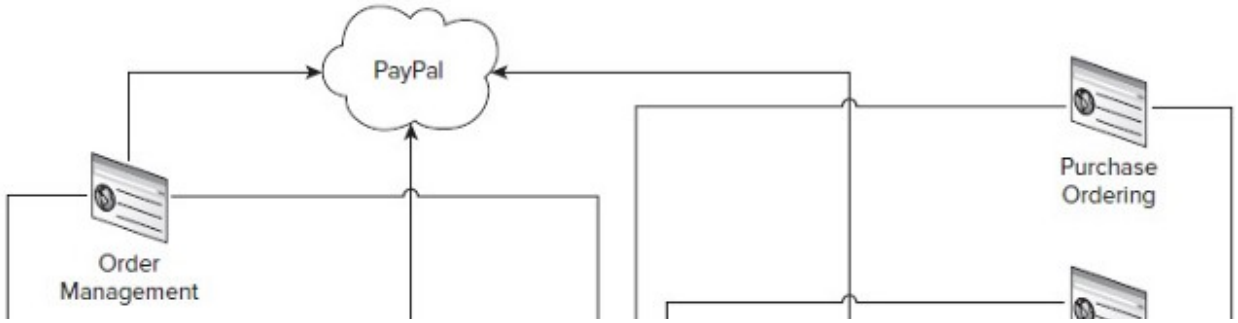
 jean pierre Deffo Fotso Sep 11, 2019 · 4 min read

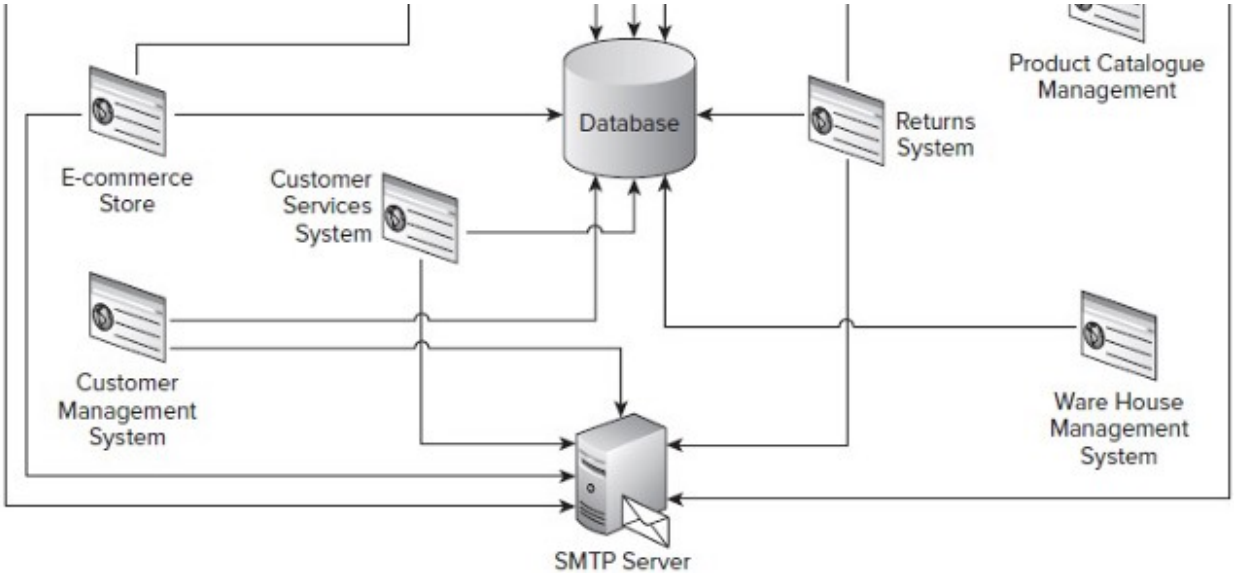
In this post, I'll try to point out some simple difference between this topics which are source of many confusion for developers and architects like me.

*Let's says immediately that DDD (**domain driven design**) pattern can be apply to all those contexts, but it fit very well as rule to be use in a microservice design as the pattern divide a complex problem to different small domain that we can wrap inside a single process*.

Monolithic approach

Many software around the world are build base on a ***monolithic approach***. In this approach, we usually have a huge single code base sometime divide in multiple module but running in the same or single process. In this case, the application is easy to develop, but very difficult to **maintain, deploy, scale, add new features and integrations .** look at the image below**:**





Monolithic architecture

The system architecture above is for an e-commerce company;all the applications contain a business model,plus logic and rules from their point of view or context. Each application connects directly with the database to persist and retrieve the state of those business objects to the database. All applications send e-mail, so they require a connection to the SMTP server. Many of the systems such as Returns, Order Management, and the E-commerce storefront require a connection to the PayPal web services for the taking and refunding of payments. The purchase ordering department manually confirms orders with suppliers via e-mail. this is how a classic monolithic application is organised, all module/libraires use in the project at the end run in the same process and therefore are deploy together to fit buisness need.This is what we as developer, do every days to write classic software and as a dotnet developer, I usually have a single visual studio solution with many class library project use to implement the buisness requirement and sometime I use DDD, 3-tiers architecture ecc…

SOA (Service Oriented Architecture) Approach

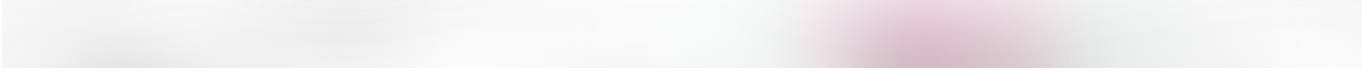
SOA on the other hand have many definitions depend on the vendor that provide platforms to hold SOA Services. ***Don Box in Microsoft*** give some guidelines which service-oriented should adhere to: ***Boundaries are explicit,Services are Autonomous, Services share schema and contract not class, compatibility is based upon policy***. The focus here is to not explain all this concept, but just look to the image bellow with our e-commerce in a SOA mode:

SOA Architecture

As you can imagine from the above image, we now divide our system in many *modules/services* which can be implement as independent project and process, in this case, we can have different team for each project, and the projects can be developed in different program language. Service always comunicate together through and "***Entreprise Event BUS***", think of Service Bus as a piece of software which have all the buisness logic to mapped and routed i**ntcoming and outgoing** messages from services, so the service bus should know the schema accepted by all services, if there is a change in one service, we have to modify and deploy again the service bus. The other thing to know is that in SOA architecture like in monolithic one, we usually have one database for all CRUD operation.

Microservices Approach

Unlike SOA, which promotes cohesion of services, microservices promote the principle of isolation of services. Each Microservice should have minimal interaction with other microservices that are part of the system this gives the advantage of independent scale and deployment of microservice. As we say before, in the context of DDD, each microservice represented a domain on the original system, let just illustrate this with the image bellow:



Microservice architecture

This image was taken from microsoft e-shop on container solution. Unlike SOA which requires all services to be connected to an ESB (Entreprise service bus), Microservices can comunicate with each other through simple message passing for example using and event bus (Different from ESB) like **rabbitmq base on AMQP protocol** to communicate asynchronously or http REST base request for synchronous communication. As a best practice, a microservice should never spawn across domain. However each domain can have multiple microservices. Microservices avoid communicating with each other and for the most part use the user interface for communication. In a DevOps perspective, each team can develop and manage a microservice rather than distributing teams around technologies and creating multiple channels of communication which can increase agility because a change can be done only on a small part of the system without impact the whole solution. The isolation between services make the adoption of continuous delivery much simpler. This allows you to safely deploy applications and roll out changes and revert deployment in case of failures. Since services can be individually versioned and deployed, significant savings are attained in the deployment and testing of microservices. The image also show that, every microservices own is database, we will avoid to use the same database accross microservices as in the SOA approach.

In this tutorial, I have not make the advantages our disadvantages of every approach, I leave the reader point out the main difference, but there are still something which I didn’t focus in this tutorial.

hope it will be helpful to people trying to know some small points of this technologies.

See you soon for the next tutorial

Regards

Microservices Soa Monolithic Domain Driven Design Design Patterns