[Team]

Team LinkedIn Account

[Alaa Magdy] [Link]

[Nagy Nabil] [Link]

[Ahmed Hisham][link]

[Reem Sameh][Link]

Difference between:

Monolith , Service oriented architecture and MicroServices

Monolithic Architecture:

What Is Monolith Architecture?
A monolithic architecture is a traditional model of a software program,
which is built as a unified unit that is self-contained
and independent from other applications.

A monolithic architecture is a singular, large computing network with one code base that couples all of the business concerns together. To make a change to this sort of application requires updating the entire stack by accessing the code base and building and deploying an updated version of the service-side interface. This makes updates restrictive and time-consuming.
Netflix is an alive example to know when monolithic architecture fails
As with the case of Netflix, monolithic applications can be quite effective until they grow too large and scaling becomes a challenge. Making a small change in a single function requires compiling and testing the entire platform, which goes against the agile approach today's developers favor.

organizations choose between monolithic or microservices depending on number of factors
and here is the advantages and disadvantages of using monolithic architecture

Advantages	Disadvantages
● **Easy deployment** – One executable file or directory makes deployment easier. ● **Development** – When an application is built with one code base, it is easier to develop. ● **Performance** – In a centralized code base and repository, one API can often perform the same function that numerous APIs perform with microservices. ● **Simplified testing** – Since a monolithic application is a single, centralized unit, end-to-end testing can be performed faster than with a distributed application. ● **Easy debugging** – With all code located in one place, it's easier to follow a request and find an issue. ● **The problems of network latency and security** are relatively less in comparison to microservices architecture**.**	● **Slower development speed** – A large, monolithic application makes development more complex and slower. ● **Scalability** – You can't scale individual components. ● **Reliability** – If there's an error in any module, it could affect the entire application's availability. ● **Barrier to technology adoption** – Any changes in the framework or language affects the entire application, making changes often expensive and time-consuming. ● It is very difficult to adopt any new technology which is well suited for a particular functionality as it affects the entire application, both in terms of time and cost. ● **Lack of flexibility** – A monolith is constrained by the technologies already used in the monolith. ● **Deployment** – A small change to a monolithic application requires the redeployment of the entire monolith. ● **For any new developer joining the project, it is very difficult** to understand the logic of a large Monolithic application even if his responsibility is related to a single functionality.

Service oriented architecture: (SOA)

Service-Oriented Architecture (SOA) is a stage in the evolution of application development and/or integration. It defines a way to make software components reusable using the interfaces.

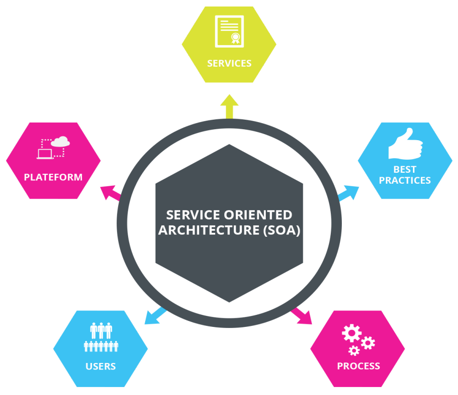
Formally, SOA is an architectural approach in which applications make use of services available in the network. In this architecture, services are provided to form applications, through a network call over the internet. It uses common communication standards to speed up and streamline the service integrations in applications. Each service in SOA is a complete business function in itself. The services are published in such a way that it makes it easy for the developers to assemble their apps using those services. Note that SOA is different from microservice architecture.

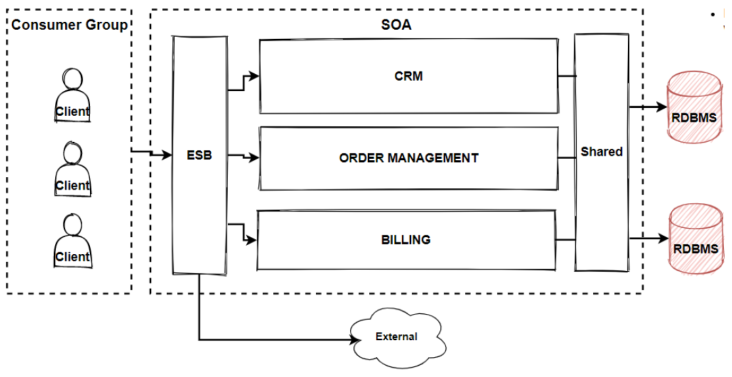
- SOA allows users to combine a large number of facilities from existing services to form applications.
- SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
- SOA-based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

It means we split up into multiple smaller services that are deployed independently.

and these services don't communicate with each other directly, but they communicate with each other using middleware.

and all of these use the same database





Advantages of SOA	Disadvantages of SOA
• **Service reusability:** In SOA, applications are made from existing services. Thus, services can be reused to make many applications. • **Easy maintenance:** As services are independent of each other they can be updated and modified easily without affecting other services. • **Platform independent:** SOA allows making a complex application by combining services picked from different sources, independent of the platform. • **Availability:** SOA facilities are easily available to anyone on request. • **Reliability:** SOA applications are more reliable because it is easy to debug small services rather than huge codes • **Scalability:** Services can run on different servers within an environment, this increases scalability	• **High overhead:** A validation of input parameters of services is done whenever services interact this decreases performance as it increases load and response time. • **High investment:** A huge initial investment is required for SOA. • **Complex service management:** When services interact they exchange messages to tasks. the number of messages may go in millions. It becomes a cumbersome task to handle a large number of messages.

What are Microservices?

As Amazon describes them: Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. These services are owned by small, self-contained teams.

Microservices architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features.

Advantages of microservices:

1-Improved productivity: Breaking an application down into smaller autonomous fragments makes it easier to build and maintain. Each service can be developed, deployed, and managed independently, and can utilize different programming languages, different technology, and different software environments based on the needs of each. The reduced codebase of each modular element of an application makes releasing, scaling, deploying and testing different services more manageable, and associated tasks can be divided among development teams and worked on simultaneously.

2-Better resiliency: Implementing microservice-based architecture adds ease to the process of identifying and resolving the root cause of performance issues. The improved fault isolation offered by individual modules means larger applications remain unaffected by a single failure. Consequently, the risk of downtime is reduced since developers can roll back an update or make changes to a module without redeploying the entire application.

3-Increased scalability: The fact that each service can be written in a different language or technology allows DevOps teams to choose the most appropriate tech stack for each module without concerns about incompatibility. Individual services can also be scaled independently, and new components can be added without requiring downtime and redeployment of the entire system. Services can also be deployed in multiple servers which reduces the performance impact of more demanding components.

4-Continuous integration/continuous delivery (CI/CD):

Continuous integration and continuous delivery are key concepts of both the DevOps philosophy and the agile approach. Microservices architecture allows cross-functional teams to develop, test, problem-solve, deploy, and update services independently, which leads to faster deployment and troubleshooting turnaround times. The ability to share the workflow burden and automate manual processes shortens the overall lifecycle of the development process.

5-Optimize business functionality: When the focus is on a specific service versus the entire application, it’s easier to customize the needs of each component to improve business functionality. Working on individual modules allows teams to focus on business capabilities instead of technologies. Existing services can then be adapted for use in different contexts without recreating an entire module from scratch.

Disadvantages of Microservices:

1-Higher Complexity: Although microservices offer many advantages, they also come with a higher degree of complexity. This complexity can be a major challenge for organizations that are not used to working with microservices. Additionally, because microservices are so independent, it can be difficult to track down errors and resolve them.

2-Increased Network Traffic: Since microservices are designed to be self-contained, they rely heavily on the network to communicate with each other. This can result in slower response times (network latency) and increased network traffic. In addition, it can be difficult to track down errors that occur when multiple microservices are communicating with each other.

3-Increased Development Time: Microservices also require more development time than monolithic applications since microservices are more complicated and require more coordination. Additionally, because microservices are deployed independently, it can take longer to get them all up and running. Also, developers need to be familiar with multiple technologies in order to work on a microservice-based application.

4-Limited Reuse of Code: Microservices also have a limited ability to reuse code, which can lead to increased development time and costs because microservices are typically written in multiple programming languages and use different technology stacks. Therefore, it can be challenging to share code between microservices.

5-Dependency on DevOps: In order to be successful with microservices, organizations need to have a strong DevOps team in place. This is due to the fact DevOps is responsible for deploying and managing microservices. Without a good DevOps team, it can be difficult to successfully implement and manage a microservice-based application.

6-Difficult in Global Testing and Debugging: Testing and debugging a microservice-based application can be difficult because the application is spread out across multiple servers and devices. In order to effectively test and debug an application, you need to have access to all of the servers and devices that are part of the system. This can be difficult to do in a large, distributed system.

Reference :

Microservices vs. monolithic architecture | Atlassian
Monolithic vs Microservices architecture - GeeksforGeeks
https://aws.amazon.com/microservices/
https://www.appdynamics.com/topics/benefits-of-microservices#~4-continuous-delivery-continuous-integration