Introduction to Architectures

Relevel
by Unacademy

Topics To Be Covered

Introduction to Architectures

Building blocks of an Architectural Design

Types of Architectural Patterns of Software Engineering

Monolithic Architecture

Service-Oriented Architecture

Differences b/w Microservices and SOA

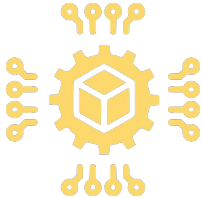
When should we choose which architecture?

Case Study: Industry Example

Introduction to Architectures

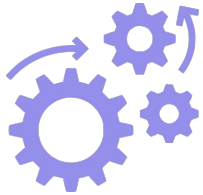
- **Architectural Design:** The process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system
- **Architectural Pattern:** It can be defined as a general, reusable solution to a commonly occurring problem in software architecture within a given context.

Building blocks of an Architectural Design



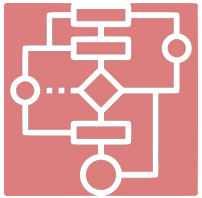
- A set of hardware and software components eg: database, etc.
- Communication Links called interfaces
- Rules that define how these components are integrated.
- A Model gives a basic understanding of the whole flow of the software.

Types of Architectural Patterns of Software Engineering



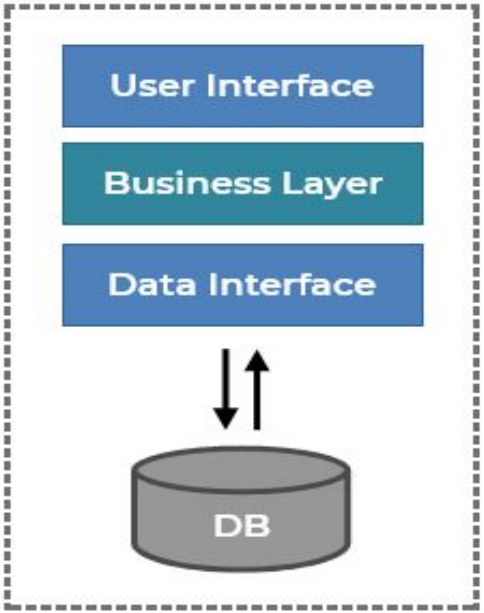
- *Monolithic Architecture*
- *Service-oriented Architecture*
- *Microservices*

Monolithic Architecture

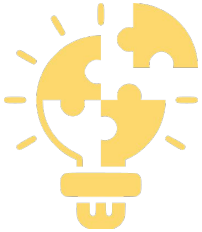


- It is used to build a software system that executes a set of highly related tasks. Thus, all the components will be assembled into a single large application with a huge codebase.
- Tightly coupled, meaning every component fully depends on every other and is meaningless if used alone.
- This is a highly centralized application, and if any one of the components fails, the entire application breaks.

Monolithic Architecture

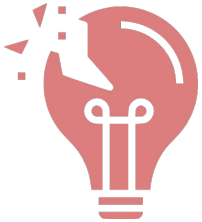


Advantages of Monolithic Architecture



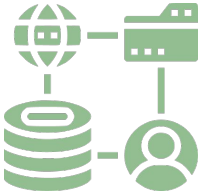
- Easy to develop, test, and deploy the application. Integration testing is comparatively better as disintegration is less and we don't have multiple pieces to take care of.
- Security is easy to maintain owing to fewer moving pieces.
- Have better throughput i.e. the rate of production or the rate at which something is processed. Because the amount of data that enters and goes through a system in a given time is very high than the other modular styles of architectural applications. Also, here we don't have dependencies and wait time for getting responses from other services. So, in a small amount of time, we can process more.
- When we want to create latency-sensitive applications like live video streaming applications. We can't rely on more HTTP or gRPC calls on other services. If there is only one service then we have to only call internal methods and functions which is quite fast in providing the real-time experience.
  Note: gRPC is similar to the HTTP protocol which helps in communication b/w services. We will discuss services shortly.

Disadvantages of Monolithic Architecture



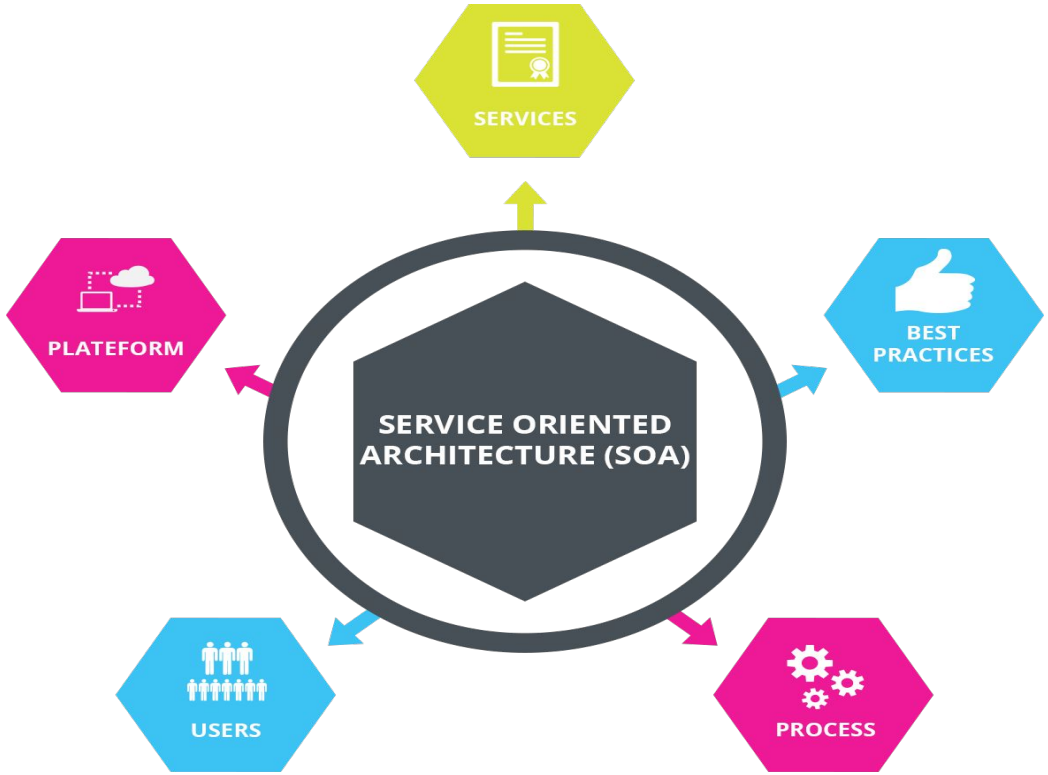
- Difficult to understand such a large codebase
- The Startup time of such applications is quite high.
- Not flexible: Each component largely depends on the other
- Updating a logic requires recompiling, retesting, and redeployment of the entire application.
- It is very challenging to scale the application as each component has different resource requirements, and you can’t scale just a single component as it largely depends on other components.
- Not Reliable: A bug in one component breaks the entire application.

Service Oriented Architecture



- Service can be defined as a self-contained, smallest unit of functionality in a large software system that does not depend on the state of any other services and can fulfill a specific function.

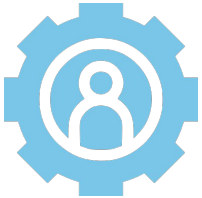
- In SOA, services are provided to other software components by application components through a communication protocol over a network.


SERVICES
PLATEFORM
BEST PRACTICES
SERVICE ORIENTED ARCHITECTURE (SOA)
USERS
PROCESS

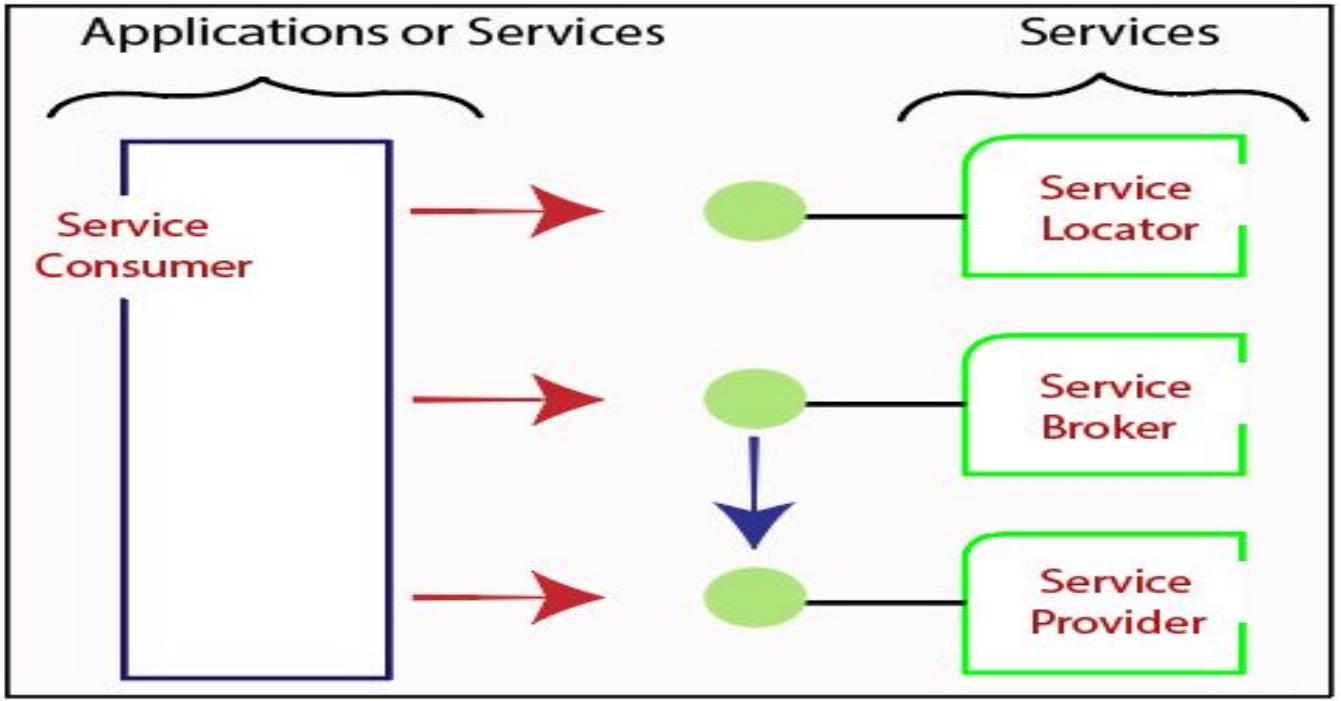
Properties of a service

- It logically represents a repeatable business entity and performs a specific desired outcome.
- It should be self-contained.
- It should be a black box for its end consumers.
- It may be composed of other small services and share some common components and the database with the main application.
- It should be secure.
- It should provide a consistent result.

Roles and Components In SOA

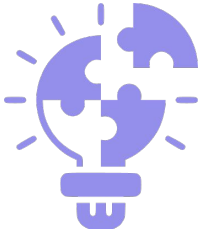


- Service provider - Implements a service based on requirements and provides it to the end consumers.
- Service consumer - The requestor or client calls a service provider.
- Service locator - This is a service provider acting as a registry.
- Service broker - This acts as a mediocre, which is also a service provider but that passes service requests from consumers to one or more additional service providers.
- Service Communication Protocol allows the service provider and the service consumer to communicate.

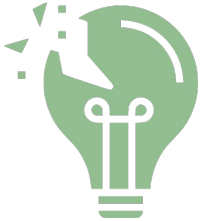

Applications or Services
Services
Service Consumer
Service Locator
Service Broker
Service Provider

Advantages of SOA

- Reusability
- Platform independent
- Scalability
- Easy to debug/Reliable
- Loosely Coupled
- Maintainability
- Fault Isolation/Tolerant

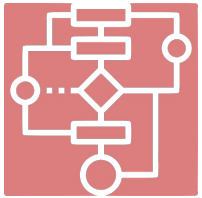


Disadvantages/Challenges of SOA



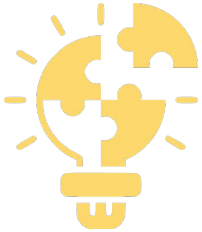
- High Overhead: It requires extra overhead time invalidating the input params
- Managing millions of messages over a network protocol is a very cumbersome task.
- This requires a high cost of the initial investment.

Microservices Architecture



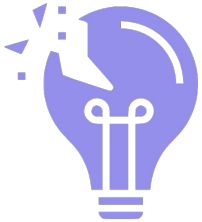
- Extension of SOA, where each service is more fine-grained and functionally independent of the other.
- These are mainly used to design complex applications and promote loose coupling between various app functionalities.
- Communicate with various other services through APIs to achieve a common business goal.
- These individual services have a different code base and are individually handled by another small set of developers.
- In contrast to SOA, we can see that each microservice has its database and independent components, whereas in SOA.

Advantages of Microservices Architecture



- Faster Development
- Small Focussed Team of Developers
- Smaller Codebase
- New technologies can be easily integrated with the main application
- If a microservice goes down and the main application is handling the exceptions well, the entire application won't break.
- It is easier to scale the main application by scaling each of the smaller microservices.
- Each microservice shares a different database, so if some schema update operation fails, it won't break the entire database as various parts of the application are not touching or accessing data from a single database.

Disadvantages of Microservices Architecture



- It is a complex system; integration testing i.e. testing the entire flow, is needed before deploying the main application.
- Writing and maintaining test cases for integration testing is more difficult than unit testing.
- When a microservice is updated, a small version update operation also needs to be done in the main application.
- While communicating over the network, a microservice may add up network latency.
- There are operations where we need to update multiple databases for a single event, which challenges maintaining data consistency.

Differences b/w Microservices and SOA

	Microservices	SOA
Architecture	Host services that can function independently	Share resources across services
Component sharing	Little bit or no component sharing involved	Component sharing is done frequently
Granularity	Fine-grained services	Services are more modular and larger in nature
Data storage	Independent data storage(Optional) for each service	Data storage is shared between services

Governance	Teams need to collaborated to discuss the set of protocols to be followed	Protocols are common across teams
Size and scope	Used mainly for smaller and web-based applications	Used mainly for large scale integrations

When should we choose which architecture?

- Monolithic architecture is usually chosen in case of building a small application, which even in future, won't require team growth.
- Microservices architecture will be used today, in most of the applications, as there is a strict requirement to be scalable as everyday there are new technologies coming up like Devops, Docker, Kubernetes, Lambda, etc. which can be integrated easily if implemented in an altogether different service unlike Monolithic architecture. Other benefits include optimal resource usage and real-time demand.
- If some of the services are dependent on each other, i.e. share some set of resources, then, for that part of the application, we can go with Service oriented architecture.

Case Study: Rainyday Grocer

Their business model targets “rainy day” moments where the customer can’t visit a grocery shop, and there is an urgent need for groceries.
MICROSERVICES ARCHITECTURE IMPLEMENTATION

- Rainyday Grocer main application is broken down into various microservices:
  - Account Login Microservice
  - Receive order microservice
  - Provide Current Market Price Estimates Microservice
  - Confirm Order microservice
- Further, Receive order microservice is broken down into smaller services to perform specific functions.

Practice/HW

- Given an e-commerce application like Amazon. List all its primary functions and give strong points to support which architectural design will be the best for it.

- Write difference b/w monolithic and microservice architecture.

Next Class

1. In the next class, we would be looking into the Introduction of the CRM App.

Thank You!