

Microservice simply an architecture of software development although it’s not the first architecture in software development. This article is all about what are the shortcoming and limitation of previous Monolithic architecture in software development and why software developers adopt the microservice architecture for the better development of applications.

**Background**

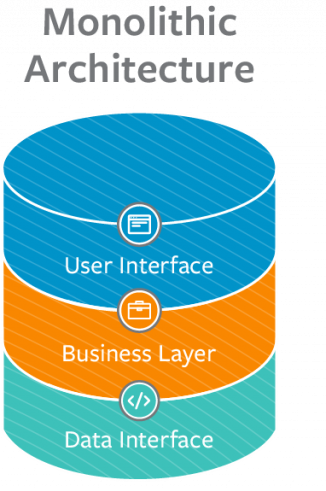
In the early days of software development, computer programming is not an easy task to do and not that much as feasible as now, only the specialist of the field can use computer programming.

In 1964 when **Basic** was introduced as a general-purpose programming language it allows the people from other departments to write a program. This leads to the rapid growth of computer applications and also increase the complexity for the software developers.

At that time computer scientist try to tackle these problems by ancient technique **‘Divide and Conquer’.** In 1972, **David Parnas** introduce the concept of **modularity and information hiding** and in 1974 **Edsger W.Dijkstra** introduces the concept of **separation of concern.** All these researches lead the software development to the modular approach and decrease the large complexity into a **loosely coupled, highly cohesive** software system. This modular approach was termed as **Monolithic Architecture.**

**Monolithic Architecture**

Monolithic means something made from a single large piece of material in software development term monolithic is an architecture developed by using single programming language and multiples modules which are divided into three layers: **presentation** layer, **business** layer and **database** layer.



In the ’90s, when the internet becomes more popular software applications rapidly grows and became more complex and large, again software developers gain one more overhead so, in 1997 **Brain Foote** and **Joseph Yoder** analyzed software applications and published **‘Big Ball Mud’** paper regarding the problem facing by software applications.

The problems facing software applications stated in the paper are:

1. Unregular Growth
2. Too many responsibilities
3. Lacks proper architecture
4. Spaghetti code
5. Make it working aka. Sweeping problems under the carpet.

**Limitation of Monolithic Architecture**

**Application Scaling:-**

Monolithic architecture based software applications support one programming language so its difficult to scale up the particular feature of an application. To scale up in monolithic we have to scale up the whole application.

**Development velocity:-**

Development velocity of the monolithic application is very slow because the modules are tightly coupled with each other and developers have a huge cognitive load.

**Development Scaling:-**

Due to the tightly coupled modules, there is less synchronization and the developers cannot work independently so hiring more developers does not produce more feature.

**Release cycle:-**

Large monolithic application takes more time to deploy usually 6 months to 2 or 3 years.

**Modularization:-**

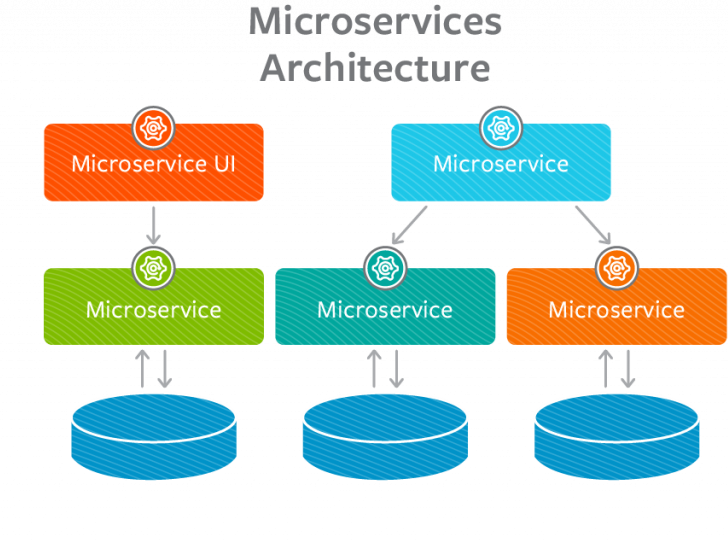
All the modules in monolithic applications are internal interfaces so they have a boundary between the modules as the size of the application increase the boundary fall apart.

**Modernization:-**

The modernization of monolithic applications is time-consuming and expensive its need a whole app to modernized without disturbing the service.

**Microservice Architecture**

Microservice architecture is an approach to build a large software application with a small unit generally called services and each service can easily deploy, developed and tested individually. However, microservice also use divide and conquer only the difference is in a monolith we have to deploy all module as whole application whereas microservice can deploy independently.



**Advantages of Microservices**

1. Applications are easier to build and maintain.
2. Adding a new feature in applications are faster.
3. Microservices can be developed independently so a developer can work on different microservice due to this less cognitive load developer take less time to write productive code.
4. Microservice can deploy independently.
5. Microservice offer loosely coupled, highly cohesive modularization.