Distributed Systems Architectures

Distributed Systems
Architectures

Moumita Asad
IIT, DU

Overview

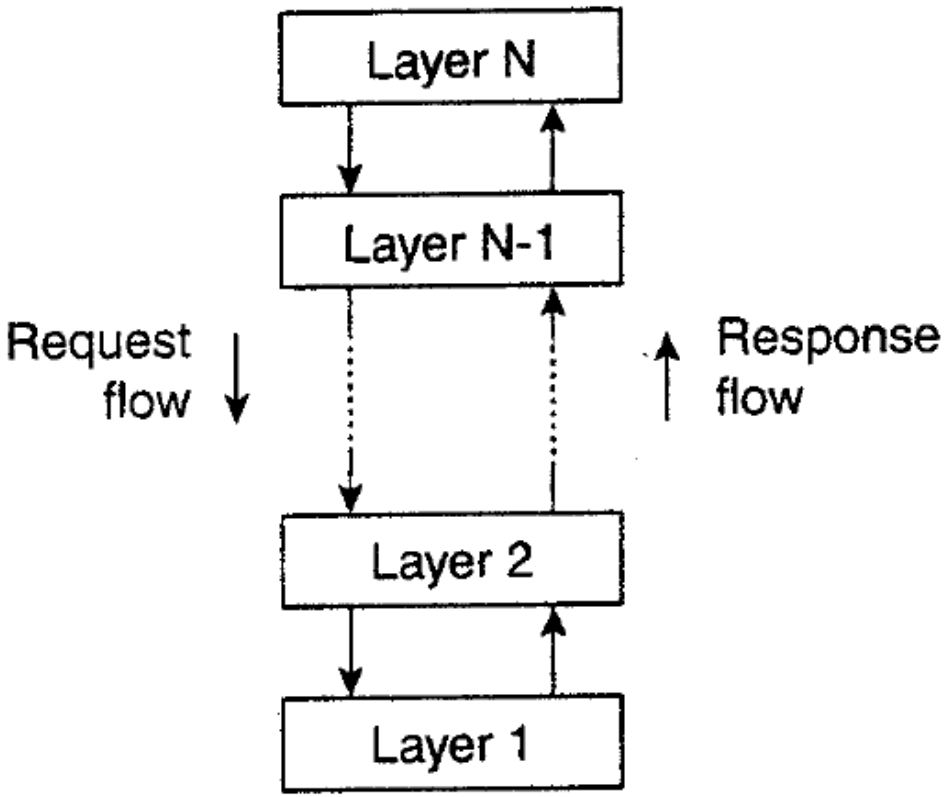
- Distributed systems are often complex pieces of software
- To master this complexity, systems must be properly organized
- Two key organization:
  1. **Software architecture:** the logical organization of a distributed system into software components
  2. **System architecture:** the placement of software components on physical machines

Software Architecture

- Important styles of architecture for distributed systems:
  - Layered Architectures
  - Object-Oriented, Service-Oriented Architectures, Microservices
  - Publish-Subscribe Architectures

Layered Architectural Style

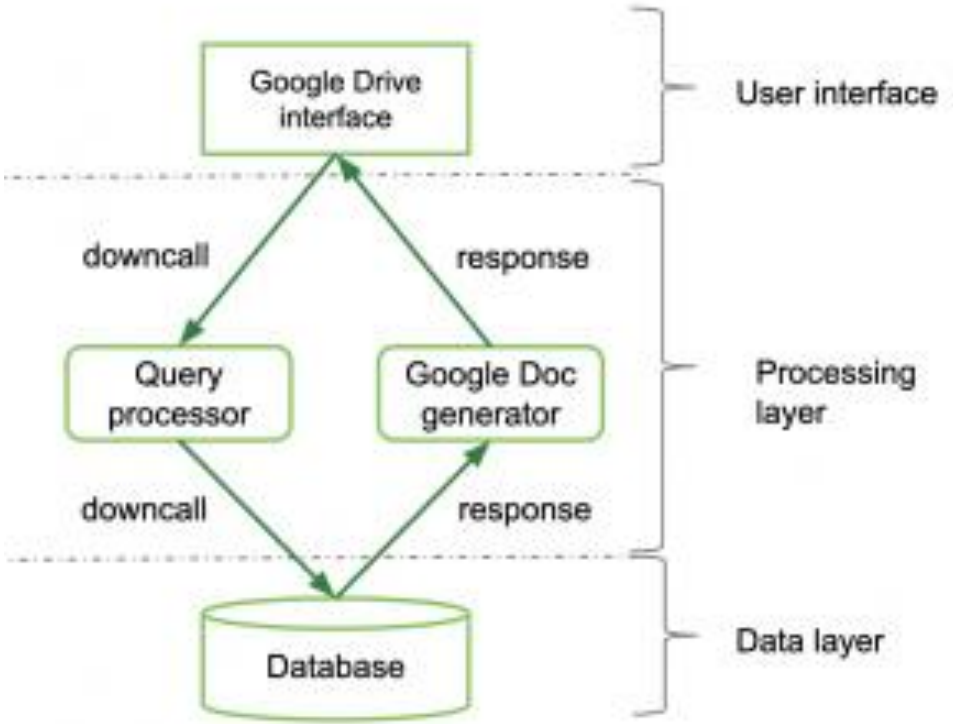
- Components are organized in layers
- Components on a higher layer make downcalls (send requests to a lower layer)
- Lower layer components respond to higher layer requests



Layered Architectural Style

- Google Docs consists of 3 layers:
  1. Interface layer: you request to see the latest doc from your drive.
  2. Processing layer: processes your request and asks for the information from the data layer.
  3. Data layer: stores persistent data (your file) and provides access to higher-level layers.
- The data layer returns the information to the processing layer which in turn sends it to the interface where you can view and edit it.

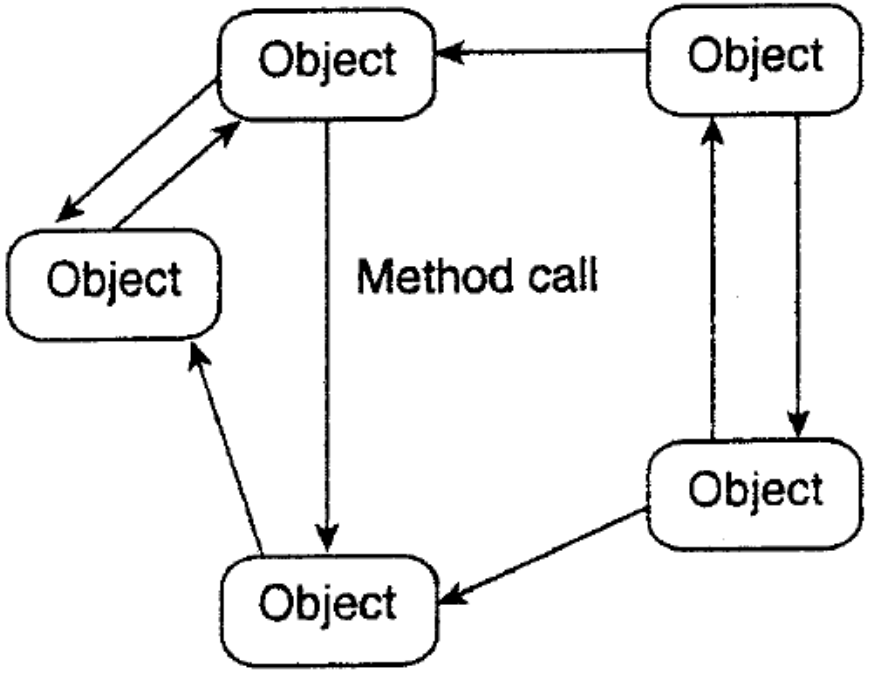
Layered Architectural Style



Object-based Architectural Styles

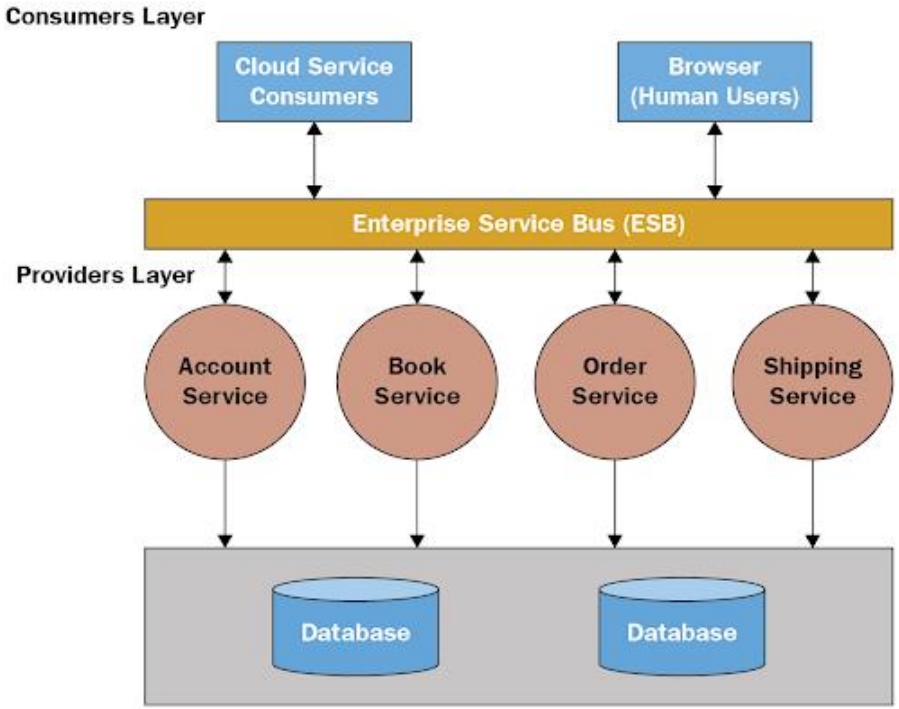
- A programming methodology
- Logical components are grouped together as objects
- Each object has its own encapsulated data set, referred to as the **object's state.**
- An **object's method** is the operations performed on that data.
- Objects are connected through **procedure call mechanisms** (an object "calls" on another object for specific requests)

Object-based Architectural Styles



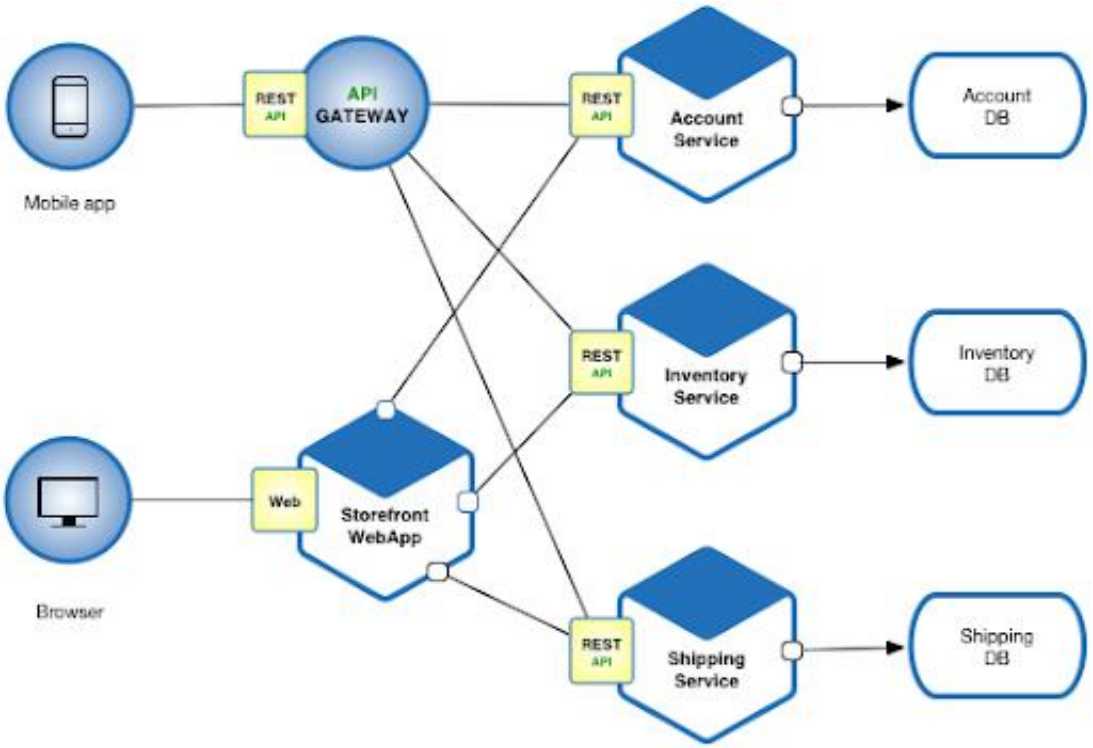
Service-Oriented Architecture

- An interface (the Enterprise Service Bus unifies all services together and exposes APIs for the frontend clients to communicate with the Providers layer



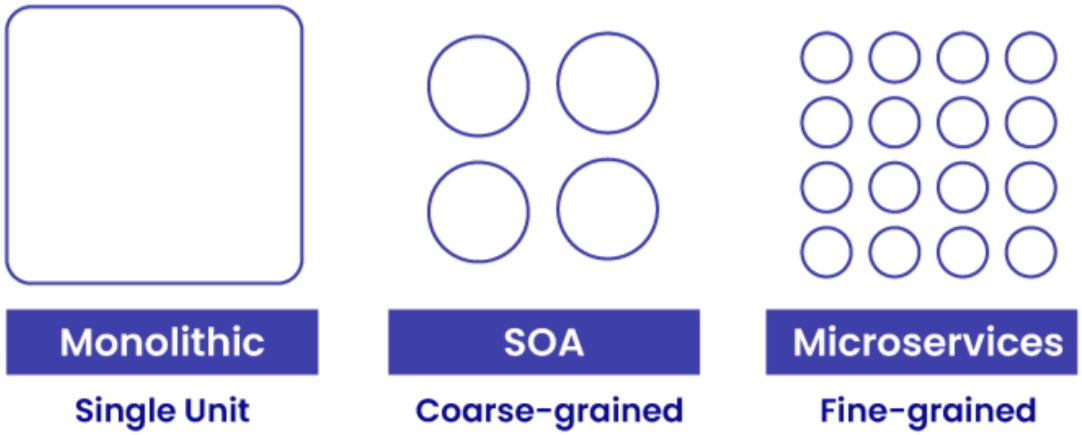
Microservices

- Microservices are smaller than services in an SOA, less tightly coupled, and more lightweight



Difference between SOA and Microservices

- SOA is “coarse grained”, meaning it focuses on large, business-domain functionalities
- Microservices is much “finer grained”, creating a mesh of functionalities that each has a single focus called a bounded context



Publish-Subscribe Architectures

- A loosely coupled architecture that allows processes to easily join or leave.
- The key difference here is how services communicate.
  - Instead of calling and getting a response, services send one-way, usually asynchronous messages, generally not to a specific receiver.
  - They rely on a configurator, administrator, or developer to configure who'll receive what message.
  - In some cases, the receivers themselves can sign up to receive messages.

Publish-Subscribe Architectures

- Example: how you get your breaking news push notifications. The Washington Post, for instance, publishes a news item categorized as “breaking news” and whoever subscribes to these updates will receive it

Remarks

- Software architectures aim at achieving distribution transparency (at a reasonable level)
- However, it requires making trade-offs between performance, fault tolerance, ease-of-programming, and so on
- There is no single solution that will meet the requirements for all possible distributed applications

Resources

- https://thenewstack.io/primer-understanding-software-and-system-architecture/
- https://scoutapm.com/blog/soa-vs-microservices