# IS4301 Agile IT with DevOps – Lecture 8

Adjunct Professor Foong Sew Bun

Department of Information Systems and Analytics

National University of Singapore

# Learning Objectives

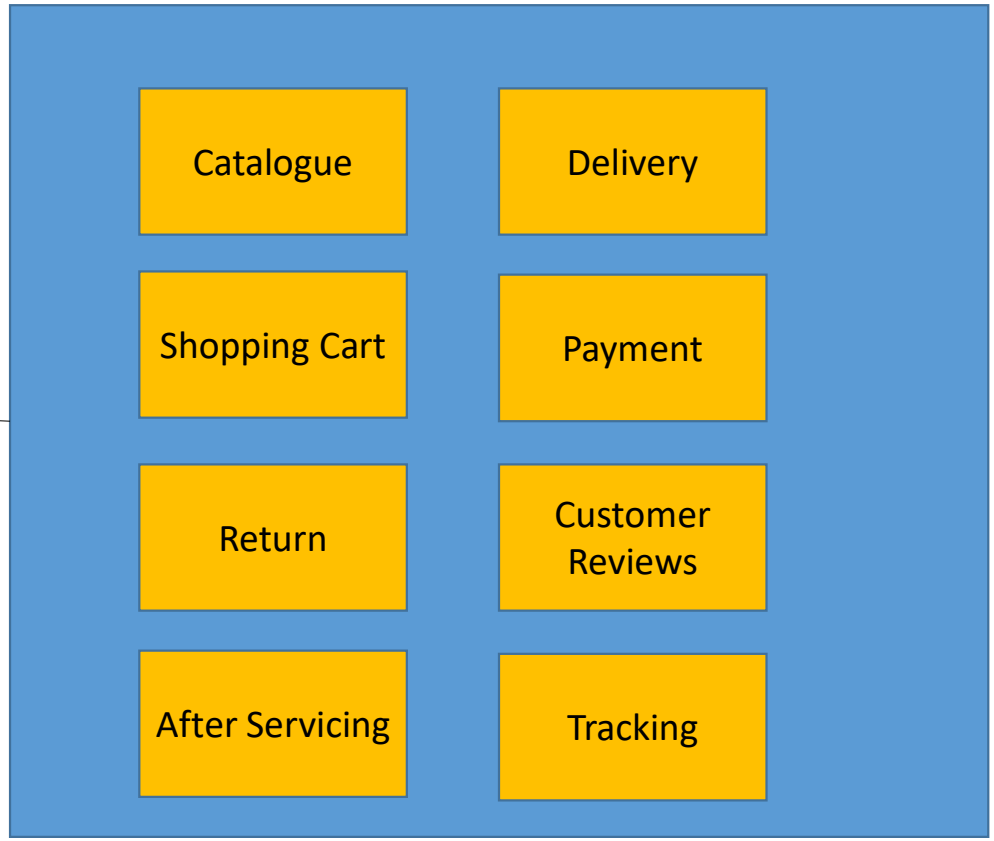
At the end of this lecture, you will understand important foundational technologies used in agile development and deployment:

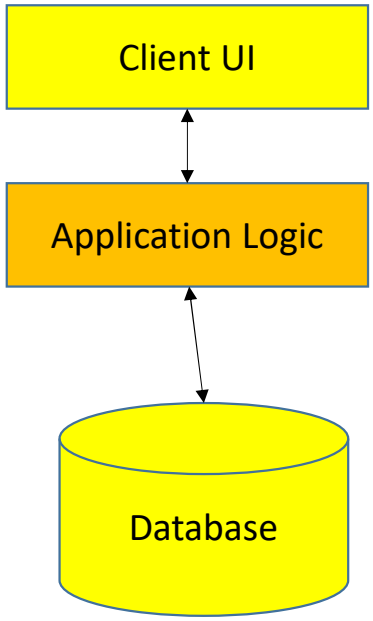
- Monolithic architecture vs. SOA vs. Microservice architecture
- How three labs will be structured to provide you with simple, get-started hands-on development using agile development open source tools

# Monolithic Architecture

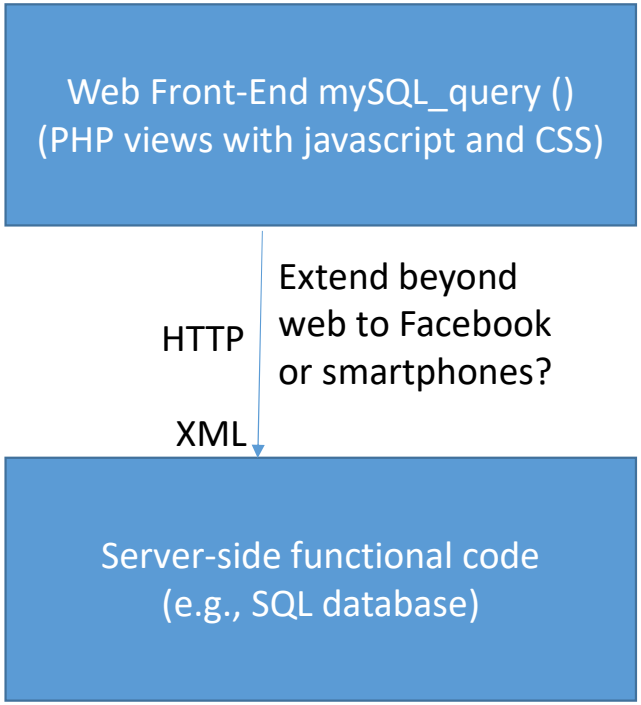
- Traditional design of applications as a single independent application. Usually consists of a 3-tier design with large code base
  - Client side user interface
  - Server-side application logic
  - Database
- Non-modular and tightly integrated

# Example of Monolithic Architecture using an Online Shopping Application

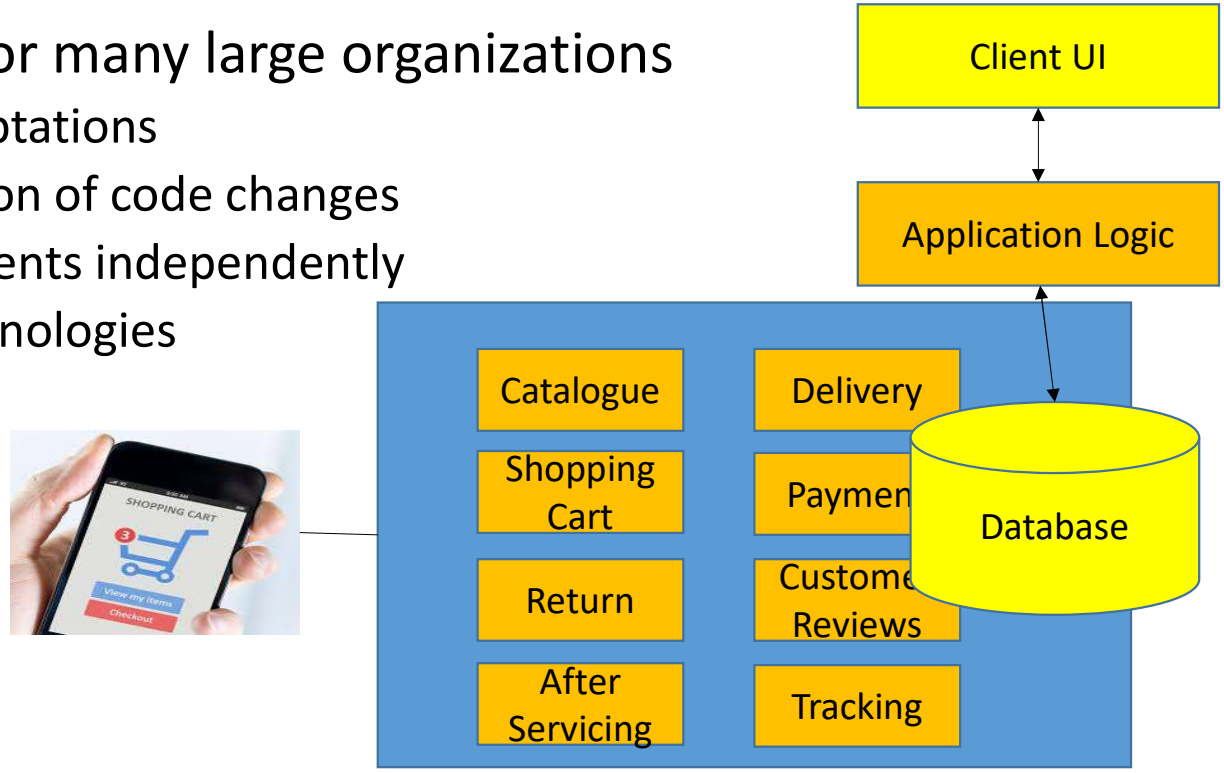



# Traditional Web Services Application using MVC pattern and Service Oriented Architecture




# Challenges of Monolithic Application

- Keeping track of changes and understanding of the changes - Large single code base written over many years by many people
- Slower time to market for many large organizations
  - Business agility and adaptations
  - High level of co-ordination of code changes
  - Unable to scale components independently
  - High barrier to new technologies



# Evolution from Monolithic to Service Oriented Architecture (SOA)

- SOA, or service-oriented architecture, defines a way to make software components reusable and interoperable via service interfaces. Services use common interface standards and an architectural pattern so they can be rapidly incorporated into new applications. This removes tasks from the application developer who previously redeveloped or duplicated existing functionality or had to know how to connect or provide interoperability with existing functions.
- Each service in an SOA embodies the code and *data* required to execute a complete, discrete business function (e.g. checking a customer's credit, calculating a monthly loan payment, or processing a mortgage application). The service interfaces provide loose coupling, meaning they can be called with little or no knowledge of how the *service* is implemented underneath, reducing the dependencies between applications.

- Definition of SOA from IBM

# Example of Monolithic Architecture using an Online Shopping Application


Catalogue

Order management

Shopping Cart

Return

Customer Reviews

Customer management

After Servicing

Delivery

Delivery Management

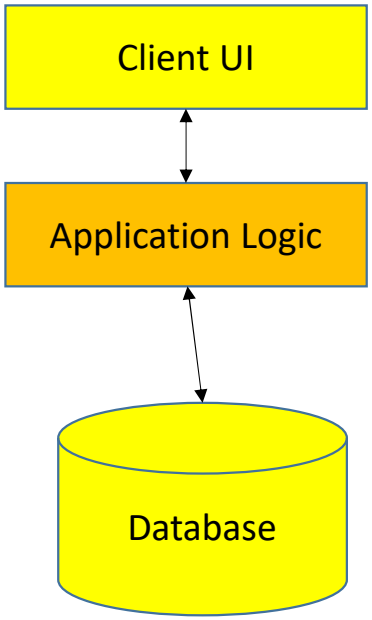
Tracking

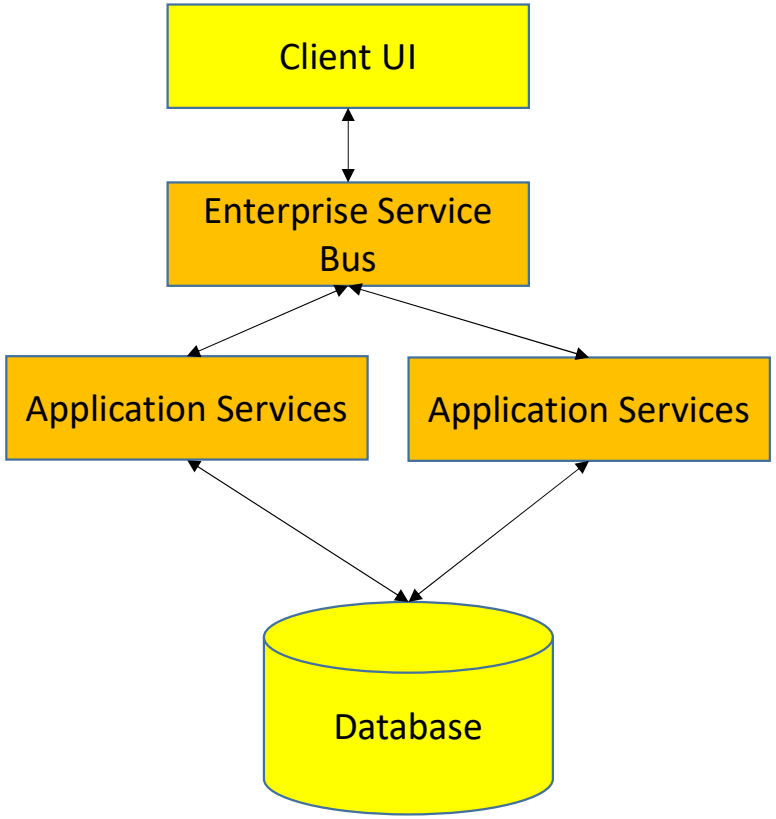
Payment Gateway

Payment Services

Service Interface over protocols of SOAP, HTTP

# Monolithic Architecture to SOA




Benefits:

- Faster time to market
- Better business agility
- Leverage legacy to expand to new business
- Better alignment to business services

# Enterprise Service Bus

An ESB, or enterprise service bus, is an architectural pattern whereby a centralized software component performs integrations between applications. It performs transformations of data models, handles connectivity/messaging, performs routing, converts communication protocols and potentially manages the composition of multiple requests. The ESB can make these integrations and transformations available as a service interface for reuse by new applications. The ESB pattern is typically implemented using a specially designed integration runtime and toolset that ensures the best possible productivity.

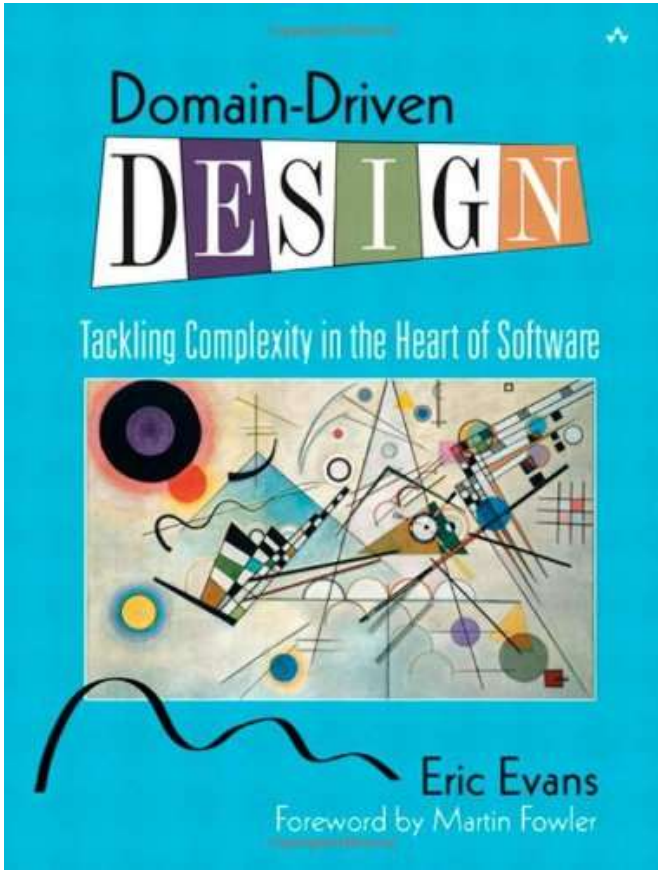
- Definition from IBM

# Microservice Architecture

- Michael Fowler: https://martinfowler.com/articles/microservices.html The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.
- IBM redbook: Microservices is an architecture style, in which large complex software applications are composed of one or more services. Microservice can be deployed independently of one another and are loosely coupled. Each of these microservices focuses on completing one task only and does that one task really well. In all cases, that one task represents a small business capability

# Characteristics of Microservice Architecture

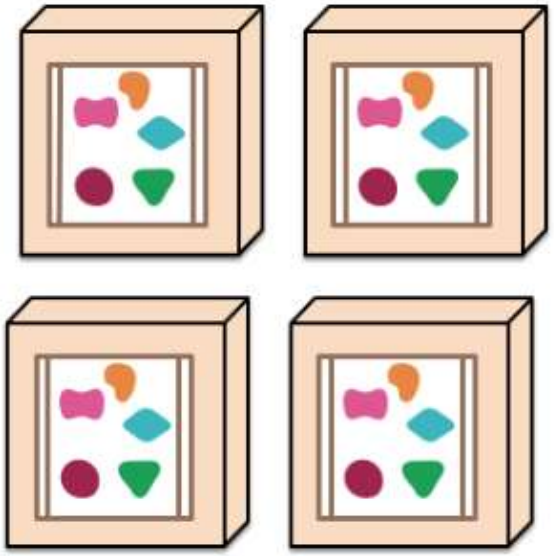
- Small and Focused
- Loosely coupled
- Language neutral
- Bounded context

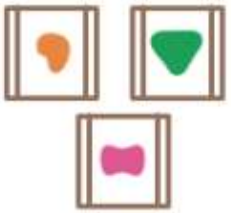
# Small and Focused

*A monolithic application puts all its functionality into a single process...*

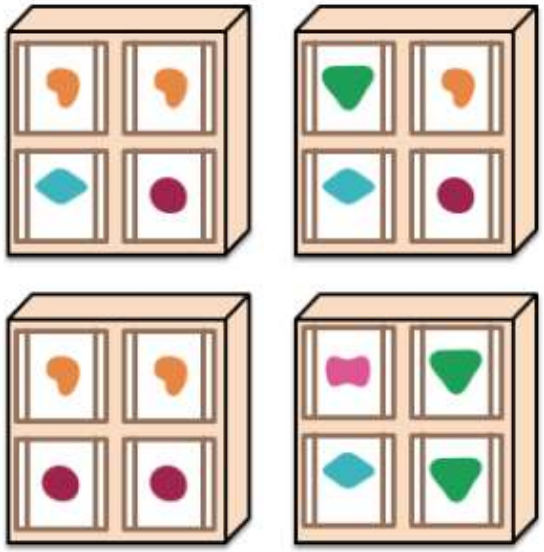
*... and scales by replicating the monolith on multiple servers*



*A microservices architecture puts each element of functionality into a separate service...*

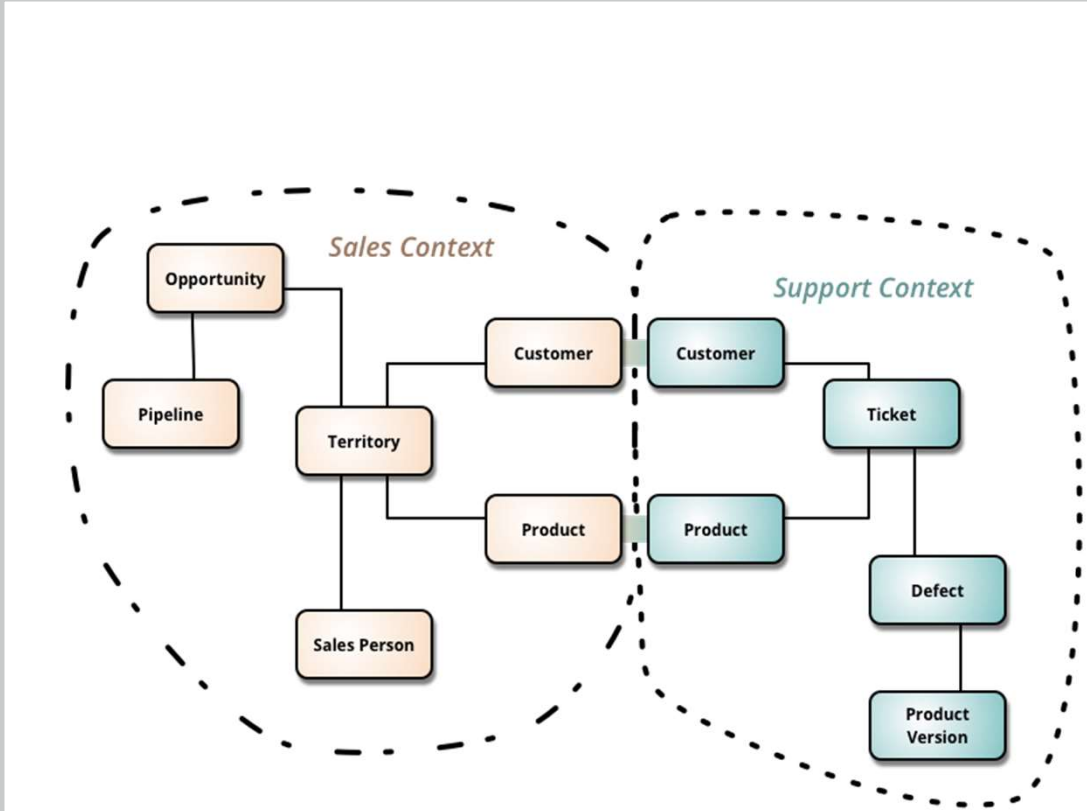


*... and scales by distributing these services across servers, replicating as needed.*



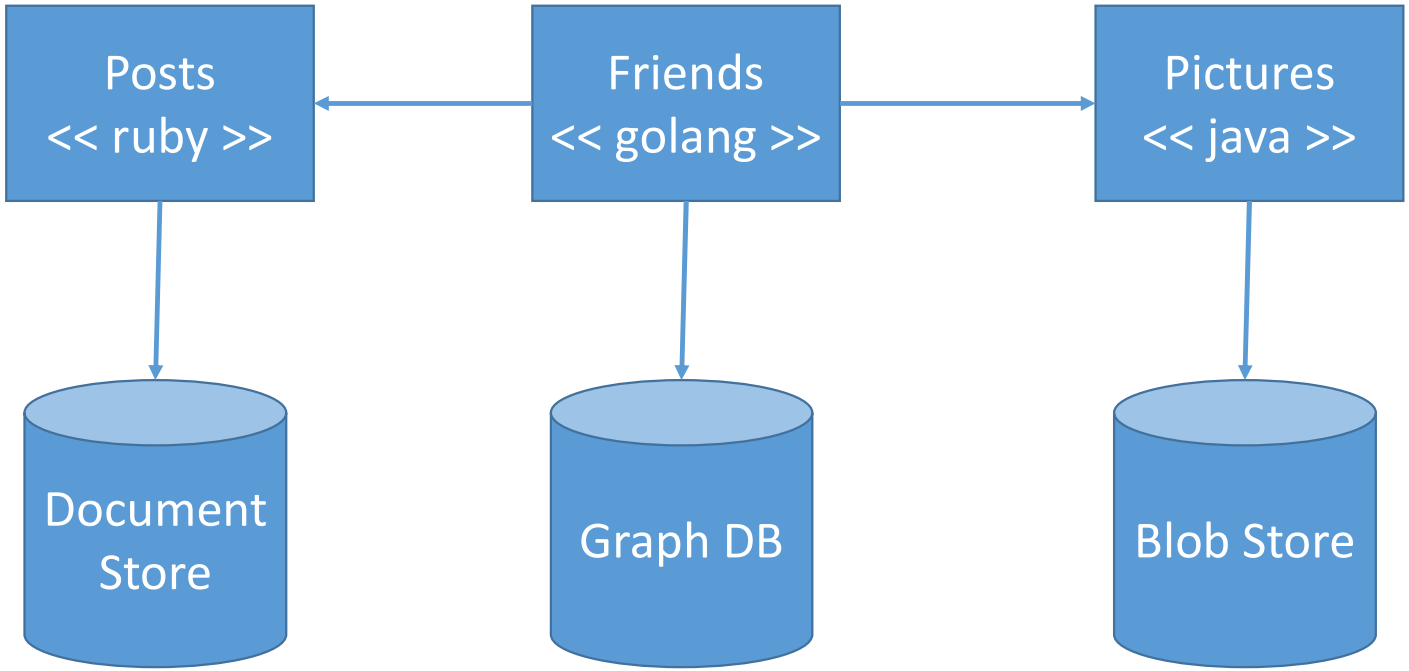
# Bounded Context

- Important pattern in domain-driven design.
- Dividing a large domain model into bounded contexts and make explicit the relationships among the contexts.

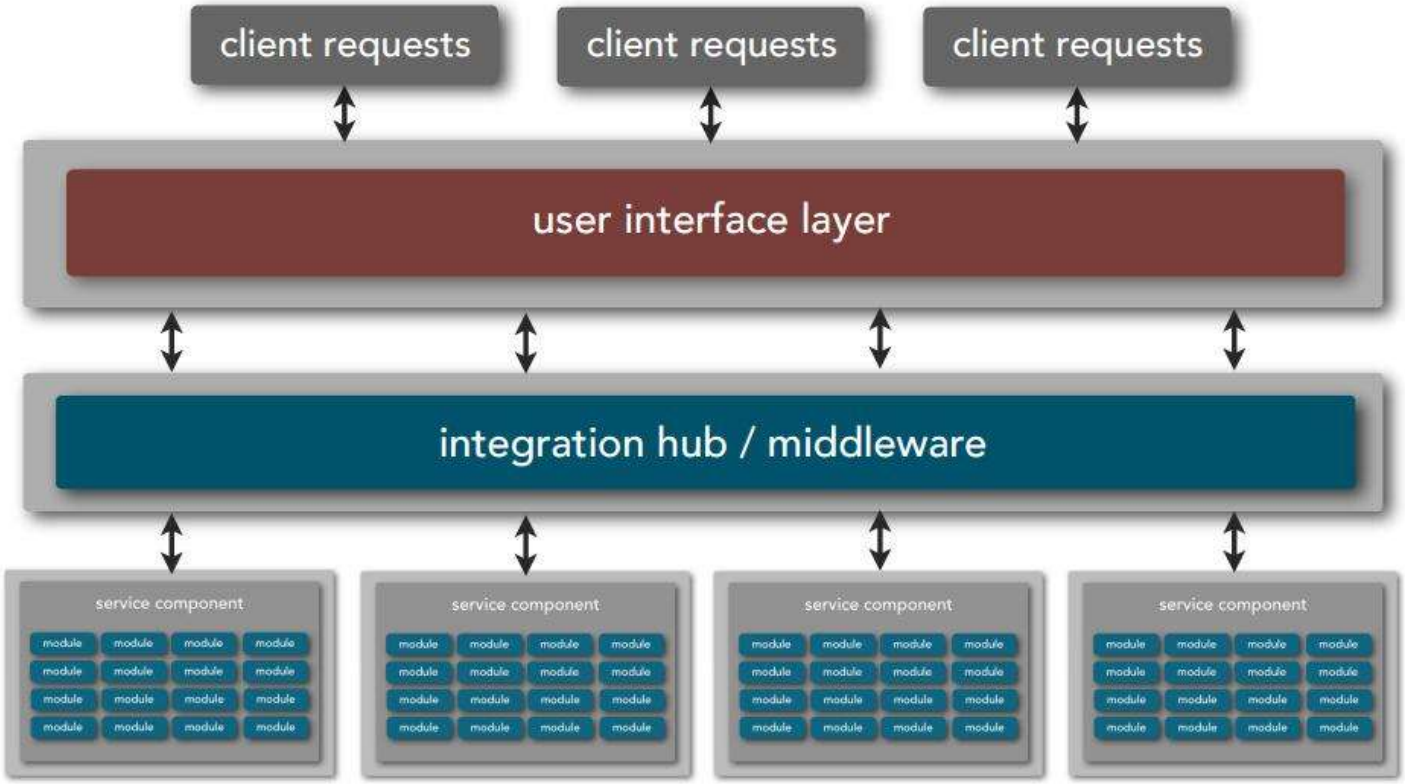


# Technology Heterogeneity with Microservices

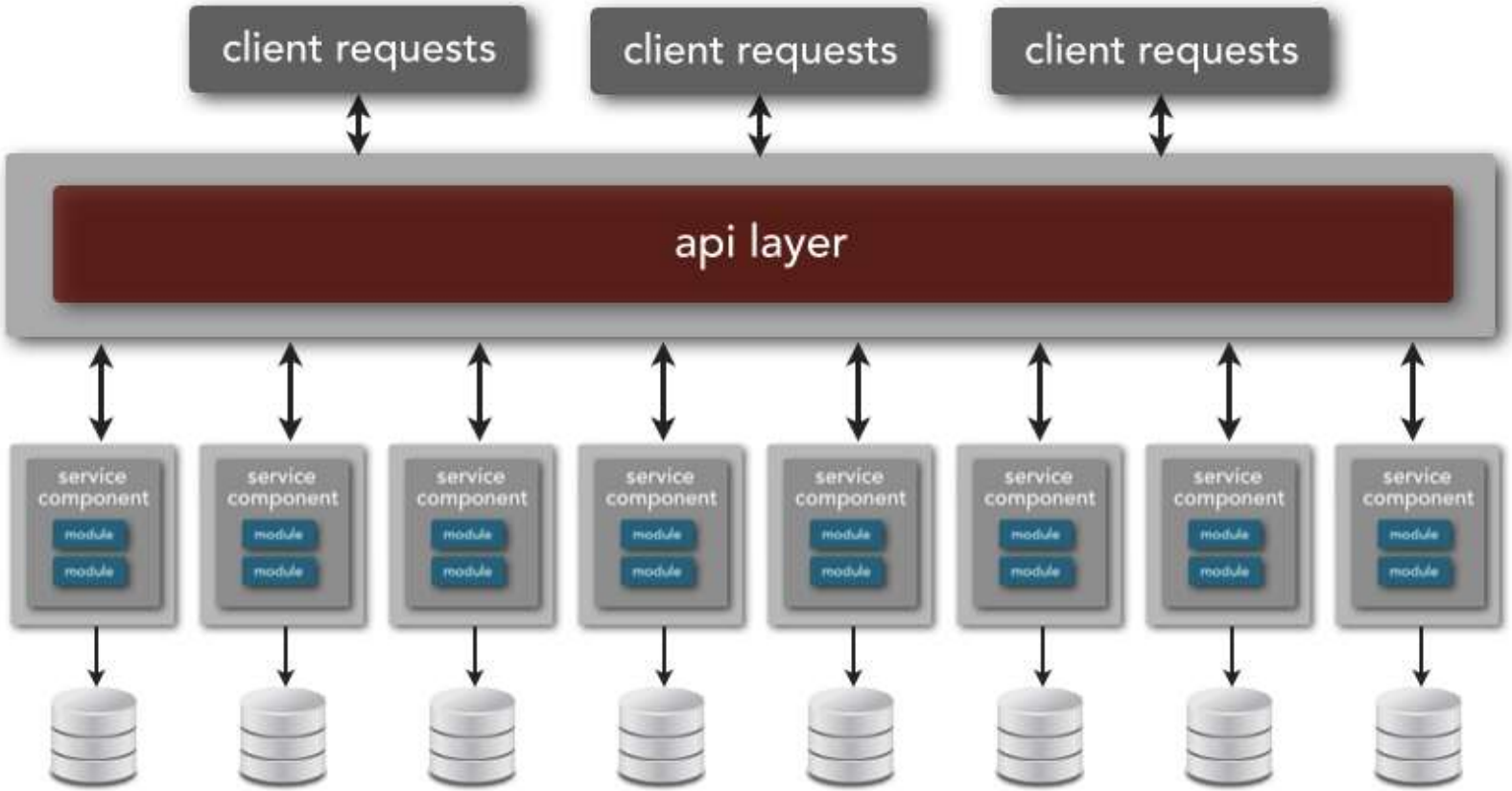
Example:



# Service Oriented Architecture (SOA) – foundation for many years



# Microservice Architecture – What are the differences can you spot as compared to SOA?



# SOA vs. Microservices

- Service granularity
- ESB vs API Gateway
- SOAP/XML vs. Http/JSON
- Storage

# Suggested Reads