Service Oriented Architecture (SOA) Introduction

Syllabus

- SOA origins, benefits and applications
- WS-* Web Services
  - XML, SOAP, WSDL etc.
- RESTful HTTP Web Services
  - Deriving REST, HTTP verbs, Richardson Maturity Model etc.
- Implementing REST and connecting to a database.
- MicroServices
- Event-Driven SOA

SOA Assessment

- In class XML assessment 10%, Week 11
- In class coding assessment 20%, Week 12
- Project 10%, due week 12

- Final Exam, 60%

SOA History

- Where did it come from?
  - philosophies and paradigms that influenced it

- 1970's
  - mainframes
  - monolithic software, terminals

- 1980's
  - PC's, spreadsheets, terminal emulation
  - (Thick client, peer-to-peer, e.g. shared printers)

Windows 3.1(1990)

SOA History

- 1990's
  - Graphical User Interfaces (GUIs) became more popular with the release of Windows.
  - Object Oriented Programming (C++, Java)
    - easier maintenance, reuse
  - thin OOP GUI front-end client to mainframe
    - OOP only used at front-end; mainframe untouched
    - GUI just communicated with the monolithic software on the server.

SOA History

- Late 90's
  - Software Components
    - e.g. Enterprise JavaBeans, Java EE allowed objects on the server
  - three tier architecture became popular
    - presentation layer (client)
    - business logic (middle tier)
    - data tier
  - attempts made to replace mainframe were not always successful
    - risky - mainframe had the core business logic
    - weak Return on Investment (ROI)

SOA History

- IT Infrastructure eventually resulted in disparate heterogeneous systems running side by side:
  - Mainframe (monolith)
  - component based servers (e.g. Java EE)
  - Multiple systems may end up with redundant data
  - So there was a requirement for integration
    - Enterprise Application Integration (EAI)

Enterprise Application Integration

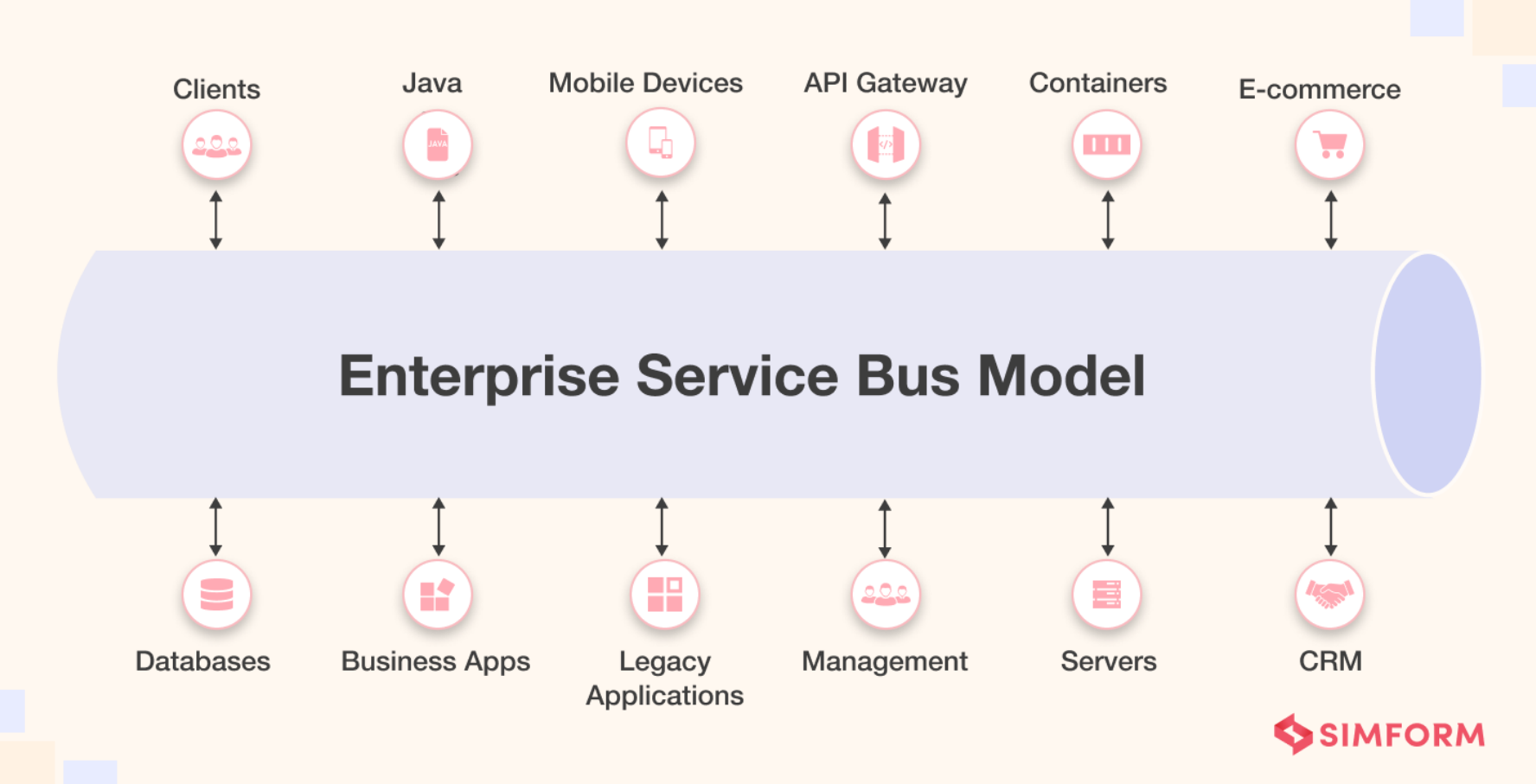


Diagram: https://www.simform.com/blog/enterprise-application-integration/

Enterprise Service Bus

- The ESB is the messaging infrastructure that connects all the disparate systems.

- The ESB has the following functions :
  - message exchange patterns (MEPs)
    - a/synchronous request/response, publish/subscribe
  - routing
  - mediation/transformation
  - Complex Event Processing (CEP)
    - event correlation, pattern matching

SOA Services (1)

- The basic idea with SOA is to encapsulate various functions in services, but having software expose services is nothing new…
  - Some referred to SOA as meaning “Same Old Architecture”
  - CORBA, Java EE and Web Services (SOAP/RESTful Web Services) are all SOA implementations

- The big change with SOA is a mindset shift:
  - Rather than thinking, we have these separate IT systems (Sales, Inventory, Billing etc…) that talk to each other, we now start with a business process that IT has to execute.

SOA Services (2)

- The focus now is on the business process and how to provide services that can support the process
  - We can re-factor the applications as a set of services
  - Then orchestrate the services together into a business process flow
    - Example on next slide

SOA: Example Services

- Customer Authentication Service: When a customer places an order, the first step is to authenticate their identity.
- Product Selection Service: This service allows the customer to browse and select products.
- Stock Service: Once products are selected, this service checks the inventory for product availability.

- You would also need *Payment*, *Order* and *Delivery* Services etc.
- These services would then be orchestrated (arranged) to deliver the business process. (to order a product in this example)

Loosely Coupled Services

- The services outlined on the previous slide can be assembled into an orchestrated workflow
- Each service is a modular component that performs a specific function to create a smooth and efficient order processing system.
- This modularity and flexibility are key advantages of SOA, allowing each service to be independently developed, maintained, and updated without disrupting the overall workflow.

SOA: Example Services

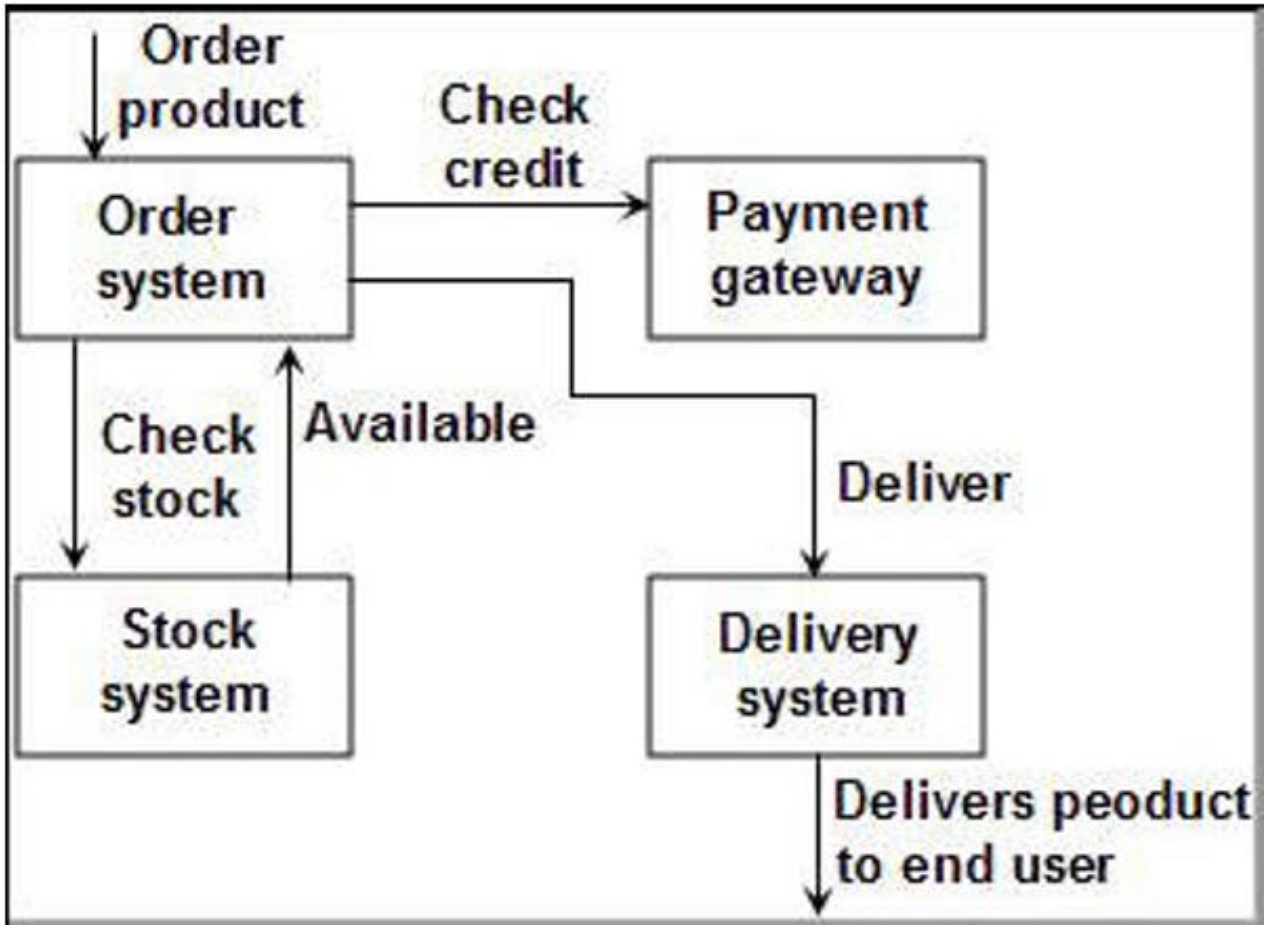


Diagram: https://www.c-sharpcorner.com/UploadFile/shivprasadk/service-oriented-architecture-soa-faqs/

SOA Service Design

- The main idea is that the services expose their functionality through these interfaces which are designed to be independent of the underlying implementation. This allows for:
- **Interoperability:** Different applications can interact with the service regardless of their underlying technologies.
- **Reusability:** The same service can be used by multiple applications, reducing the need to duplicate functionality.
- **Abstraction:** Consumers of the service don't need to understand the complex logic behind the service; they only need to know how to interact with its interface.
- **Loose coupling:** Services are designed to minimize dependencies, which allows for easier integration and changes.

SOA enabled reusing old monolithic applications (1)

- Enterprises no longer wanted their business functionality buried deep in monolithic applications which are difficult to integrate with.

- They wanted to abstract the business logic out into services so that these services can be invoked and re-used from a variety of locations and platforms.

SOA enabled reusing old monolithic applications (2)

- It made sense to get more out of earlier investments when developing new applications as labour is the largest IT cost, so if existing software could be reused, then enterprises would save time and money.

- That was the original argument, but times have moved on and most old monolith applications are now decommissioned. We'll focus on building new applications with SOA/Microservices.

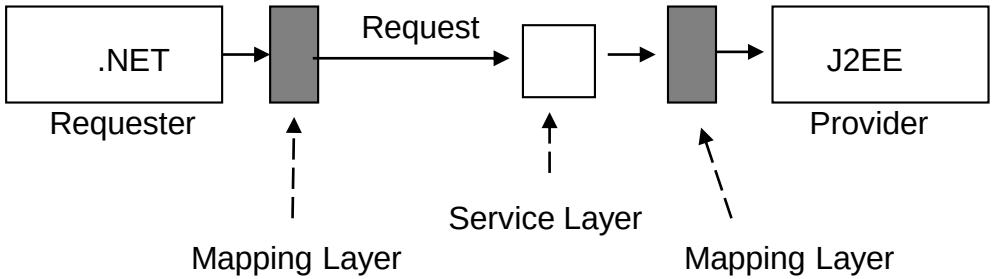
Terminology: Service Interface and Mapping

- A service interface is a machine-readable description of the messages a service receives and optionally returns.

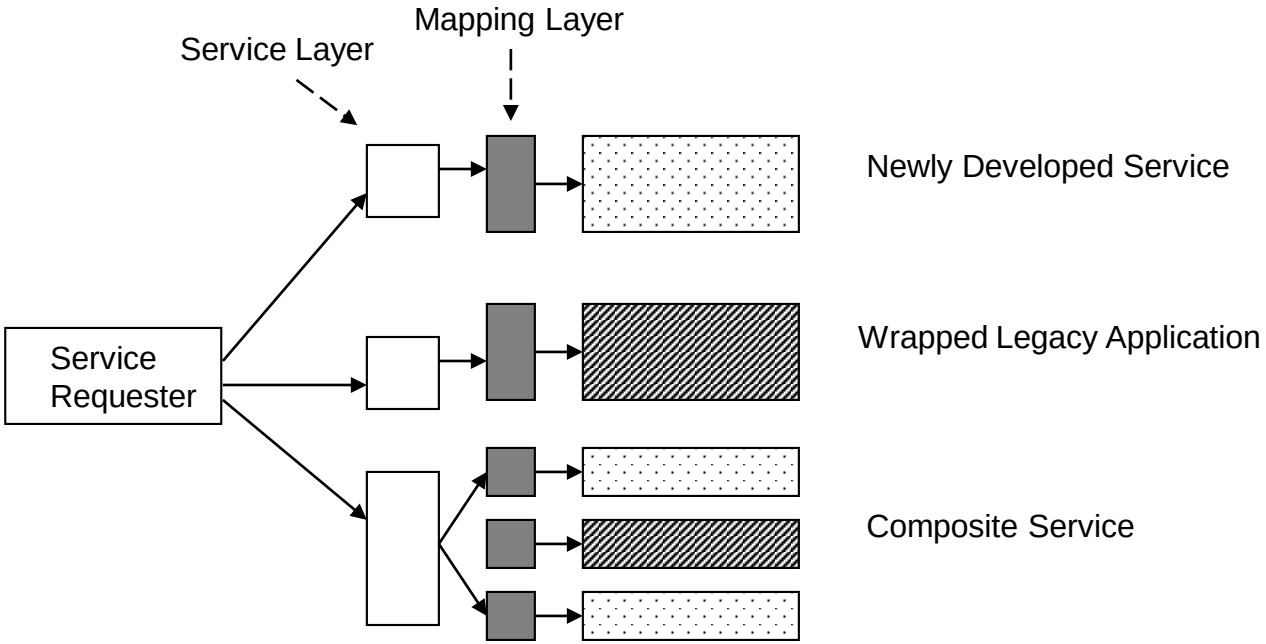
- The mapping layer is an essential component in bridging the gap between different systems that may use different data formats or data types. It's responsible for converting data between the client's native data types and the data types expected by the service platform.

- E.g. Client-to-XML Mapping involves converting the client's data, often in its native format (e.g., JSON or custom objects), into XML format. XML is a common choice for representing structured data in a machine-readable way.

Mapping layer to map two systems:



Mapping layer used to map to many different system types:



SOA Implementations

- SOA is not Web Services. You do not need Web Services to have an SOA, although these days they are usually implemented with RESTful Web Services.

- SOA implementations:
  - CORBA
  - Java RMI
  - Web Services (SOAP or REST)
    - XML and JSON are key enablers

Microservices vs SOA

- Microservices and Service-Oriented Architecture (SOA) are both architectural approaches for designing and building distributed systems that promote modularity, reusability, and flexibility in software development.

- While they share some similarities, they also have distinct differences. These are outlined on the following slides.

Scope and Granularity

- Microservices: Microservices are typically smaller, more fine-grained services that focus on specific, narrowly defined functionalities. Each microservice is a self-contained unit that can be developed, deployed, and scaled independently.

- SOA: SOA encompasses a broader approach to service-oriented architecture, often including larger and more coarse-grained services. These services may encapsulate more extensive business processes and might have more complex interactions.

Scope and Granularity

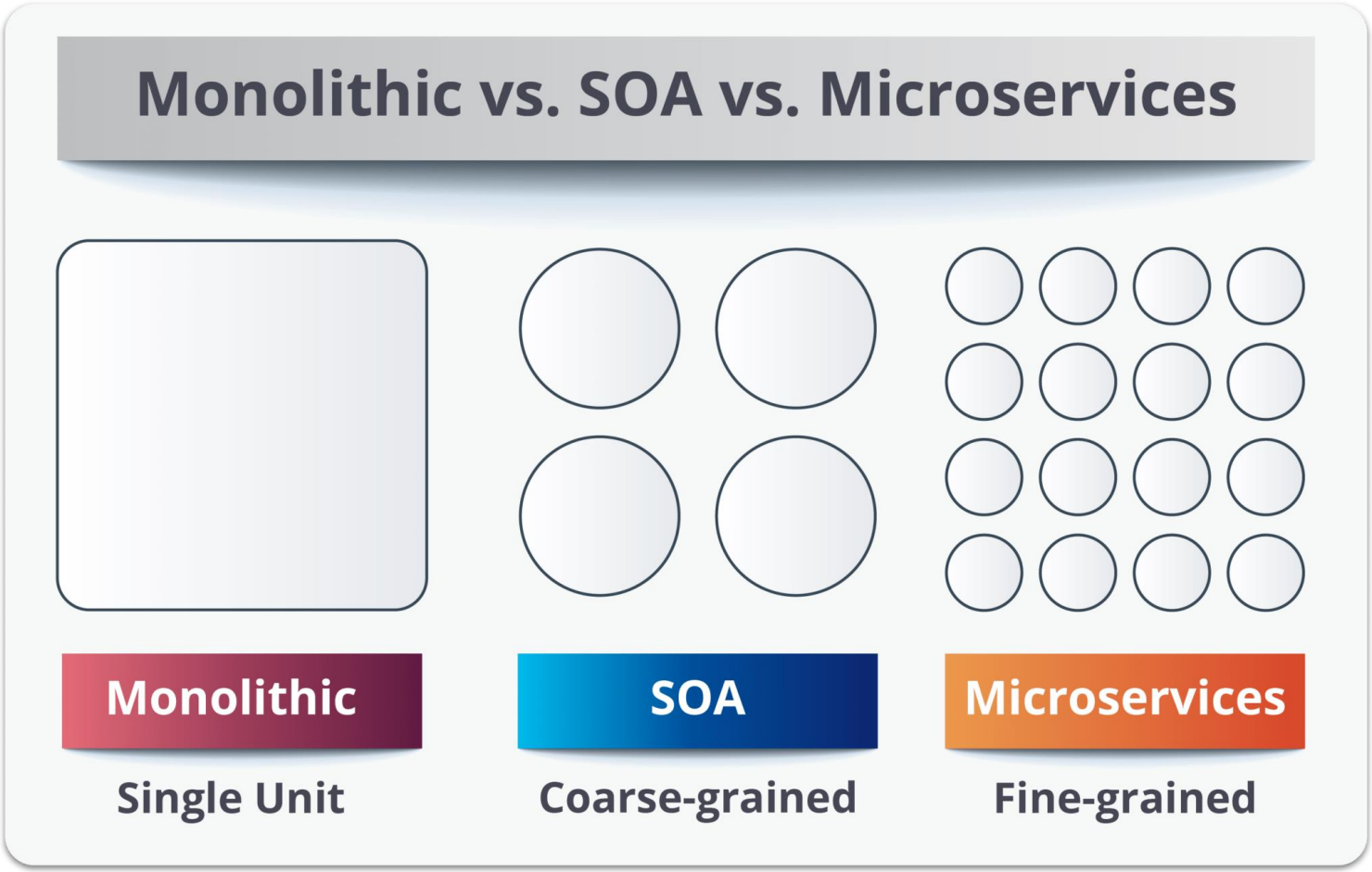


Diagram: https://dzone.com/articles/microservices-vs-soa-whats-the-difference

Independence and Deployment

- Microservices: Microservices emphasize the independence of services, allowing teams to develop, deploy, and scale them autonomously. This promotes faster development cycles and agility.

- SOA: SOA also promotes service independence, but it may involve more centralized governance and coordination. SOA services might be managed by a central service bus or repository.

Data Management

- Microservices: Microservices often advocate for decentralized data management, where each microservice manages its own database or data store.

- SOA: SOA may support various data management approaches, including centralized data services.

Data Management

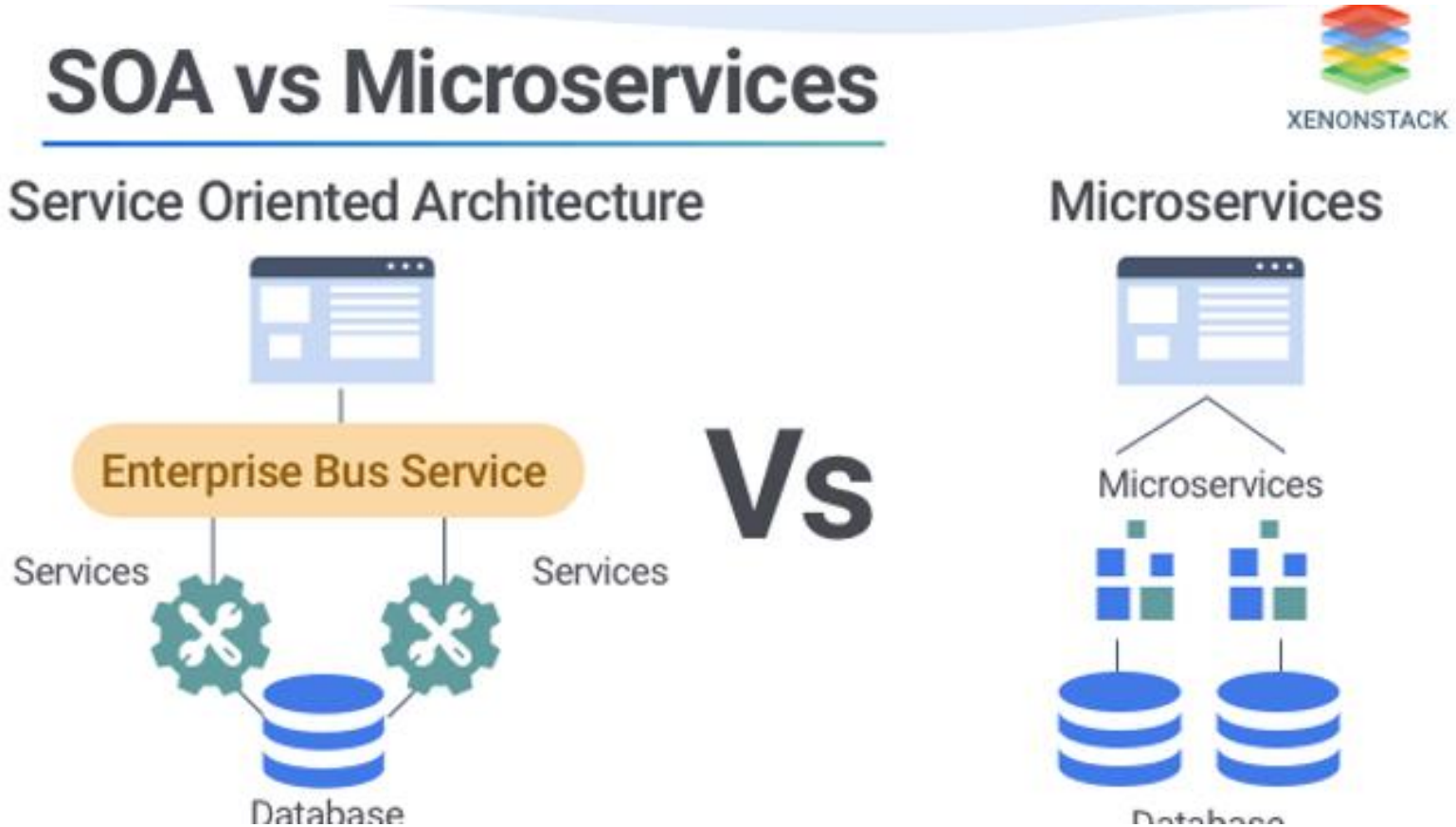


Diagram: https://www.integrate.io/blog/soa-vs-microservices/

Technology Stack

- **Microservices**: Microservices often leverage modern technology stacks and tools, such as containerization (e.g., Docker), orchestration (e.g., Kubernetes), and DevOps practices (e.g. CI/CD).

- **SOA**: SOA can be associated with a broader range of technologies, including older enterprise service buses (ESBs) and more heterogeneous environments.

Next steps in course

- First, we'll have a review of XML – this is used primarily in SOAP, but also in REST.
- Then we'll look at SOAP Web Services
- Then onto RESTful Web Services
- More detail on Microservices will follow throughout the course