University of York
Europe Campus
CITY College

Introduction to Service Orientation

CCS3341 Cloud Computing

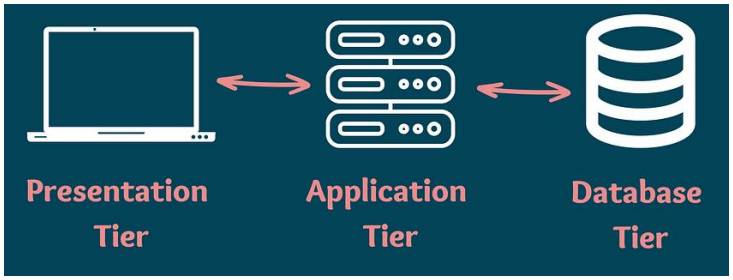
Dr S. Veloudis

Web Client-Server Era

2000s-10s

With the ascendancy of the web in the mid 2000s and the need to access data ubiquitously, possibly through mobile devices with limited capabilities, the pendulum was swung back to thin clients…

3-tier paradigm



Note: When smart hand-held devices were originally introduced, their processing capabilities were indeed limited. Nevertheless, nowadays hand-held devices have higher capabilities which has swung the pendulum again (!), this time towards Single Page Apps (SPAs).

A Single Page Application (SPA) is a web application that functions without the need for the page to reload during use

Note: SPAs use JS frameworks that run in the user's browser. When a user loads an SPA, the initial HTML, CSS, and JavaScript files are downloaded to the user's device, and the framework is responsible for rendering the content and handling user interactions. The frameworks then communicate with the server asynchronously to retrieve/update data as needed. This approach reduces the amount of data that needs to be sent back and forth between the server and the client.

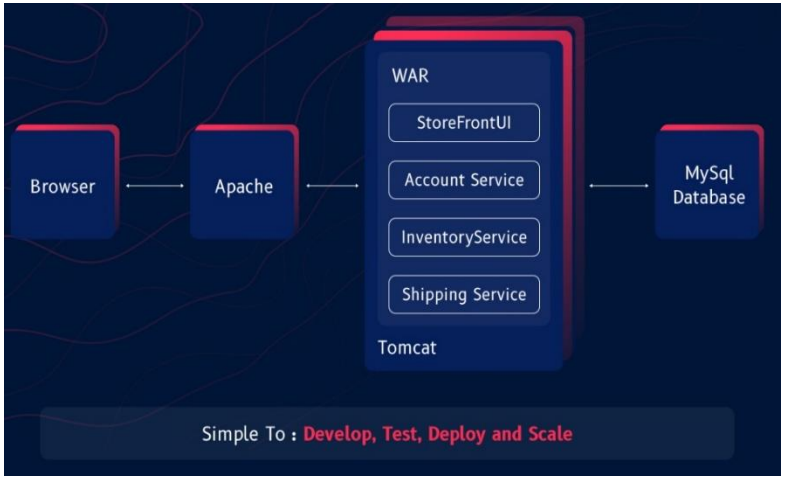
Web Client-Server Era

- The need for separation of concerns brings about further componentization
  - **n-tier architecture**

    - A business logic tier
    - A security tier
    - A tier to control integration…

    Paves the way for SOAs!

Note: Functionally different aspects (e.g., data I/O, data processing, error handling, security, etc.) are interwoven in the same code



- Although monolithic middle-tiers have proven relatively successful, they do suffer important drawbacks

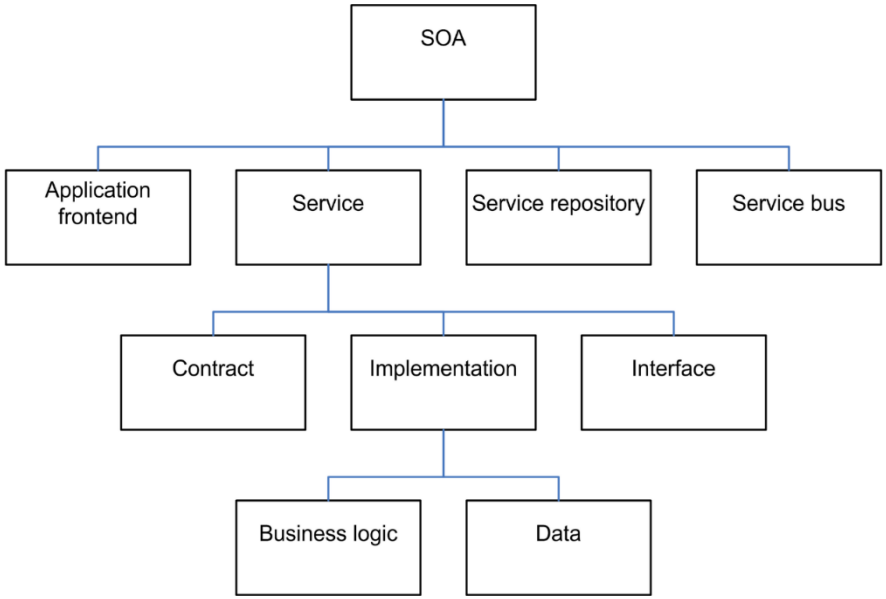
  - Maintenance & extensibility
  - **Complex CI/CD**
  - No "best-of-breed" alternatives
  - **Monolithic scaling**

Continuous Integration (CI): regularly merging code changes from multiple developers into a single codebase. Involves automating the build process to compile the code and run automated tests.

Continuous Deployment (CD): automating the deployment of code changes to production environments. Involves setting up a pipeline to deploy code changes once they pass automated tests.

SOA

- SOA represents a distinct approach to separating concerns
- An application is developed as a set of collaborating services, each responsible for delivering a particular set of functionalities



Note: In the literature, SOA is often confused with Web services

A Web service is a service offered by an electronic device to another electronic device, communicating with each other via the WWW

Web services are particularly suitable for implementing SOA. SOA is therefore intertwined today with Web services – contemporary SOA represents an architecture that promotes service-orientation using Web services

SOA – Autonomy & Abstraction

- **Autonomy**

  - A seminal characteristic of SOA is the fact that services are autonomous
    - They are **deployed** and **governed** independently of each other
      - May be deployed over heterogeneous platforms
    - They are **self-contained** entities with full control over the business logic that they encapsulate

- **Abstraction**

  - Services hide the business logic that they encapsulate
  - This logic is only accessible through well-defined interfaces
    - This is analogous to objects in OO programming
  - This interface is called an application programming interface (API)

Extensibility/ maintainability

Loose service coupling promotes simpler extensibility and maintainability

SOA – Autonomy & Abstraction

- **Composability** (**of utmost importance!**)

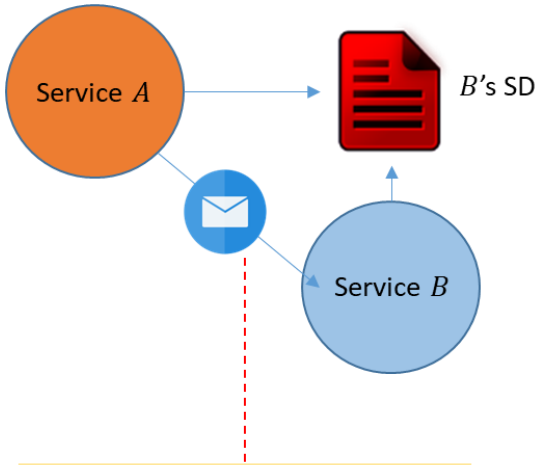
  - Supports the automation of flexible and highly adaptive business processes
  - Promotes agility and enables better scalability
  - Allows for “best of breed” alternatives
  - Facilitates maintenance/extensibility

- **Interactivity**

  - For services to be composable, they must be able to interact:
    - Services must be aware of each other i.e., service A is aware of the existence of service B and of what B can do
    - Services are able to exchange information

SOA – Interaction

- For a service A to be aware of the capabilities of a service B, as well as of how to interact with B, A must be in possession of B's service description (SD)
  - A SD establishes the name and the whereabouts of a service, as well as its data exchange requirements
    - **Does not reveal information about the service's internal workings**
  - SDs must be expressed in a formalism understandable by all services
- Services exchange information through messaging
  - This preserves a loosely-coupled relationship between services



Note: Messages must be outfitted with enough information so that they can be understood and processed successfully at the receiving service without any knowledge of the code at the sender side that created the message

SOA – Open Standards

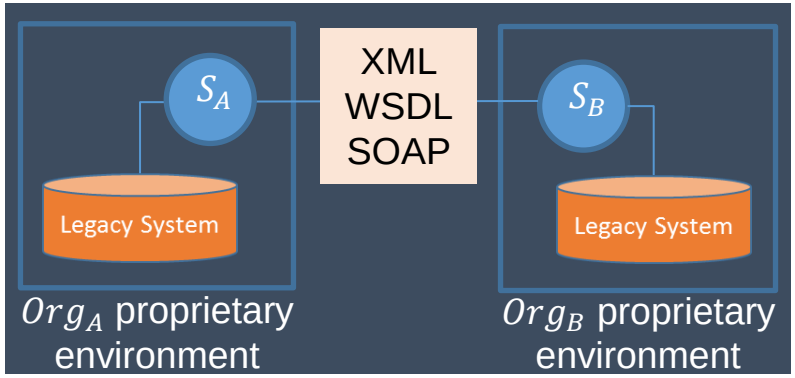
- A seminal characteristic of SOA is that services exchange data by adhering to vendor-neutral **open standards**
  - **XML**
  - **WSDL**
  - **SOAP**

The use of such standards brings about

- Interoperability
  - Standards bridge the heterogeneity in services
    - Reusability
    - Simplifies cross-organisation development

SOA – Other Features

- Federation
  - SOA facilitates the federation of legacy application logic by outfitting it with services that expose it via the open standards communications framework
  - Previously isolated legacy environments can now interoperate without requiring the development of expensive and fragile point-to-point integrations



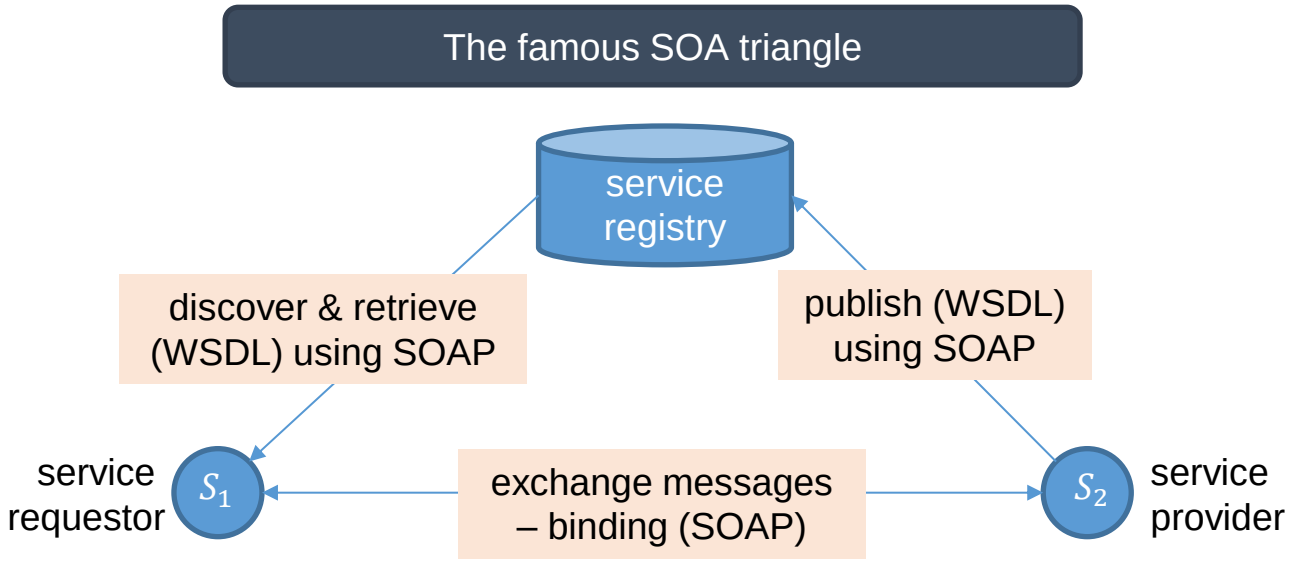
Note: SOA fosters organisational agility by enabling agile SW development, deployment, maintainability, and federation processes

- Reusability
  - Services may be reused across different applications

Service granularity is an important factor…

SOA – Benefits

- Discoverability
  - Services are designed to be outwardly descriptive so that they can be discovered, assessed and accessed
  - Discoverability typically relies on some form of service registry or directory which maintains and manages service descriptions



SOA – Service Roles

- **Service provider**
  - A service acts as a provider when:
    - It is invoked by an external agent (the service requestor)
    - It provides a published service description
  - The term may also refer to the organisation (or individual) responsible for providing the service

- **Service registry**
  - Maintains and manages service descriptions

UDDI has not been as widely adopted as anticipated

Note: A notable service registry is the one included in the UDDI (Universal Description, Discovery, Integration) protocol (UDDI also includes a mechanism to register and locate web services). UDDI comprises three components:

- White pages – provide information about the business supplying a service.
- Yellow pages – provide a classification of services based on standard taxonomies.
- Green pages – provide information on how to access a service (e.g. its address and required parameters).

SOA – Service Roles

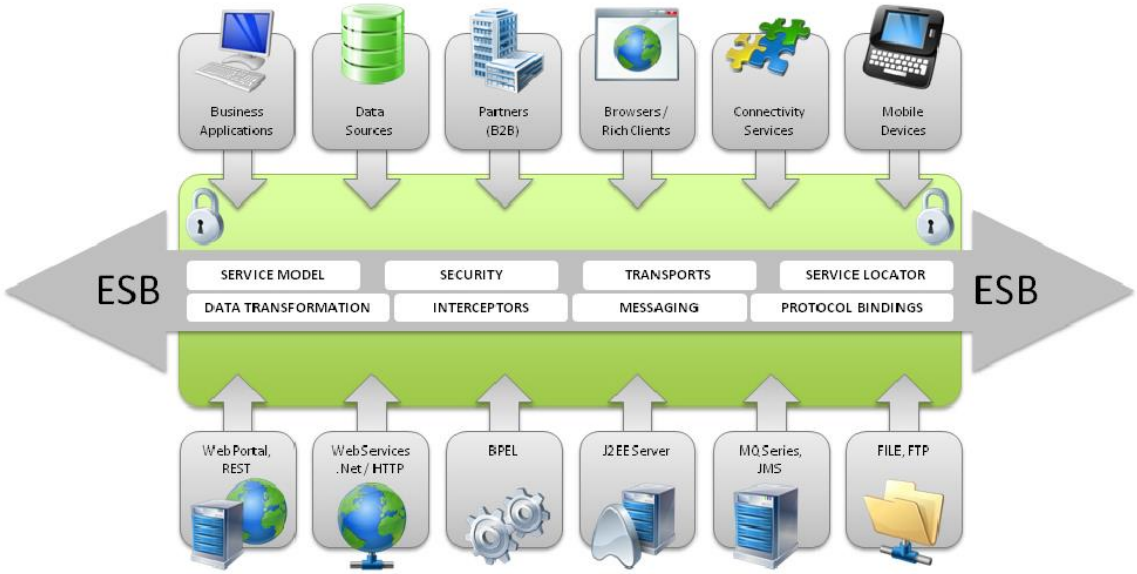
- **Service requestor**
  - A service acts as a requestor when:
    - Invokes (or binds to) a service provider by sending it a message
    - Queries a registry to locate an appropriate electronic service
- **Service intermediaries**
  - Services that intercept messages to process them and/or route them
    - For example, a load balancing service or a security service
  - Service intermediaries act as:
    - Service requestors when they initially intercept a message
    - Service providers when they resend the message after processing it
  - Two kinds :
    - Passive intermediaries – they only read but not change the intercepted message (e.g., a load balancing service)
    - Active intermediaries – they may alter the intercepted message
      - For example, a security service may digitally sign the contents of the intercepted message before resending it to its destination

SOA – Service Models

- Business service
  - The most fundamental service model
  - Encapsulates domain-specific business logic with a well-defined functional boundary
- Utility service
  - Provides generic functionality typically not associated with domain-specific business logic
  - E.g., a load balancing service
- Controller service
  - Service compositions comprise independent business and utility services each contributing to the overall execution of a business task
  - The integration and coordination of these services is often a task of its own and can be assigned to a service – the controller service
    - The controller service may be a dedicated service or a utility or business service capable of executing this task

SOA – Service Models

- Controller service – Enterprise Service Bus (ESB)
  - An ESB is a software architecture model used for designing and implementing communication between interacting services
  - Its primary purpose is to provide an enterprise application integration platform for services deployed across heterogeneous platforms (e.g., Java, .NET, etc.)
    - Combines messaging, web services, data transformation, intelligent routing and security features



SOA – Service Models

- ESB core functionalities
  - Routing
    - Enterprise developers traditionally write both business and communication logic
    - The latter makes assumptions about communication senders and receivers
      - These assumptions reflect into the code
      - A change to enterprise connections means changes in the code
  - To tackle this problem in SOA, ESB has the ability to provide intelligent routing
    - Pub-sub messaging
      - Asynchronous, decoupled communication with any number of senders/receivers
      - Ability to define topics to which receivers can subscribe and receive messages
    - Message brokering – route message based on payload type/content
    - Load-balanced routing
    - Failover routing – redirect messages to alternate location when a recipient service (provider) is not responsive

SOA – Service Models

- Message mediation
  - ESB can bridge differences in protocols, message formats, security models, and communication semantics
    - ESB can transform an incoming message that abides to a particular format into several outgoing formats (e.g. XML to JSON, XML to Java objects)
  - ESB can "enhance" a message by, for instance, appending informational data to it, or by changing the date format or the number of decimal points
  - ESB may offer security mediation (e.g. mediation between Windows integrated security and X.509 certificates)
- Uses standards for external connections (WSDL, SOAP)

SOA – Service Models

- ESB offers centralised management
  - Provides a single place to handle control function
  - Makes change management straightforward
  - ESB is logically centralised but may be physically decentralized
    - The various services that it offers may themselves be distributed across different physical machines