Service Oriented Architecture: Key Concepts

Lecture 7

University of Technology, Mauritius

Service Oriented Architecture

WAT 5101C

G.Suddul

Outline

- Defining SOA
  - Software Architecture
- Elements of SOA
  - Front-end Applications
  - Service
  - Service Repository
  - Service Bus

Software Architecture

- Different definitions of software architecture:
  - *"An architecture is the set of significant decisions about the organisation of a software."* – [2]
  - *"The software architecture of a program or computing system is the structure or structures of the system, which comprise of software elements, the externally visible properties of those elements, and the relationships among them."* – [3]

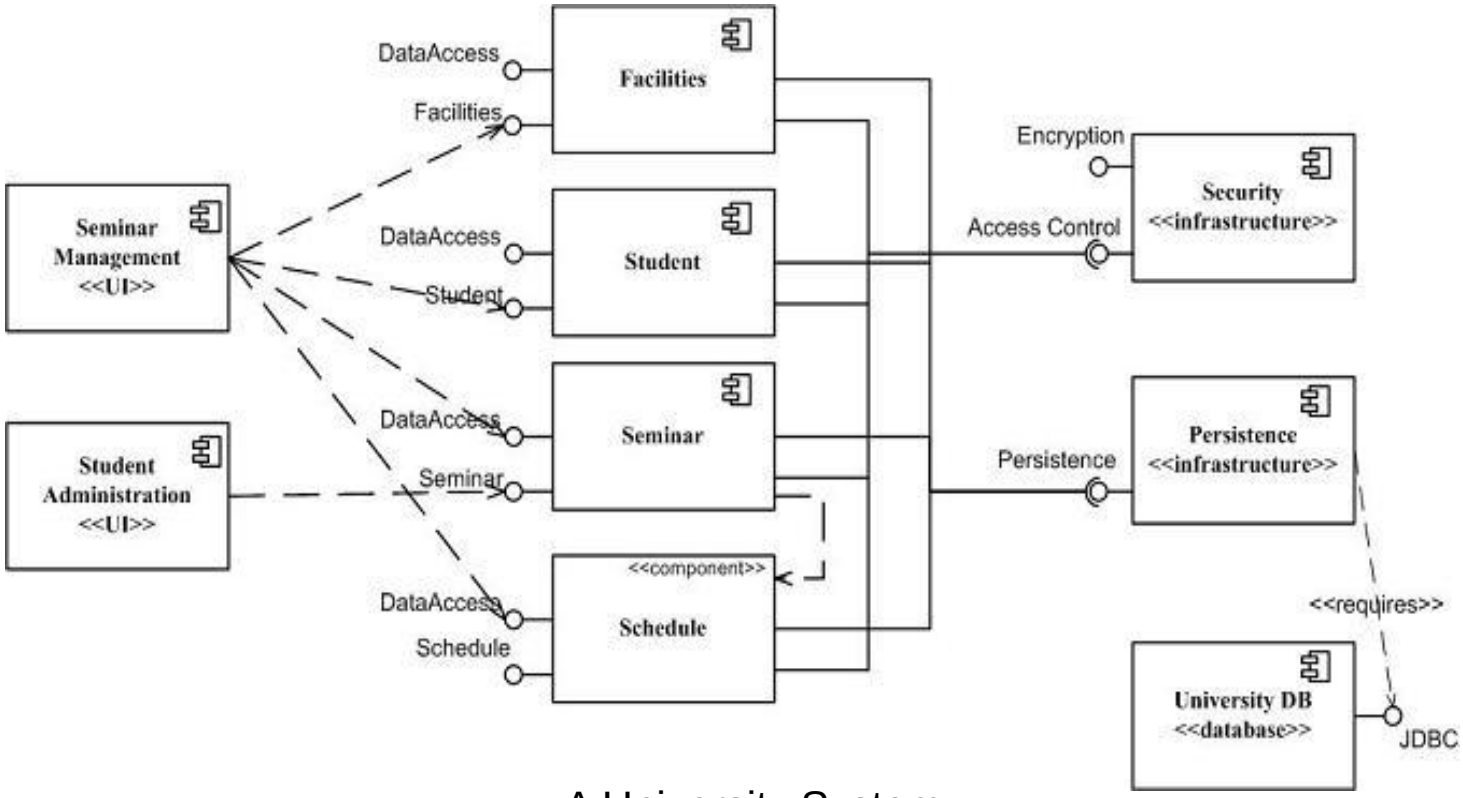
Software Architecture

- *"A software architecture is a set of statements that describes software components as assigns the functionality of the system to these components. It describes the technical structure, constraints, and characteristics of the components and the interfaces between them. The architecture is the blueprint for the system and therefore the implicit high-level plan for its construction."* – [3]

- Note:
  - Software Components - (technical structure, constraints and interfaces).
  - Blueprint for the system.

How to model high-level Software Components ?

- UML 2.0 notation consists of component diagrams which enable the modeling of the high-level software components, and more importantly the interfaces to those components *(see below).*



A University System

Application & Application Landscape

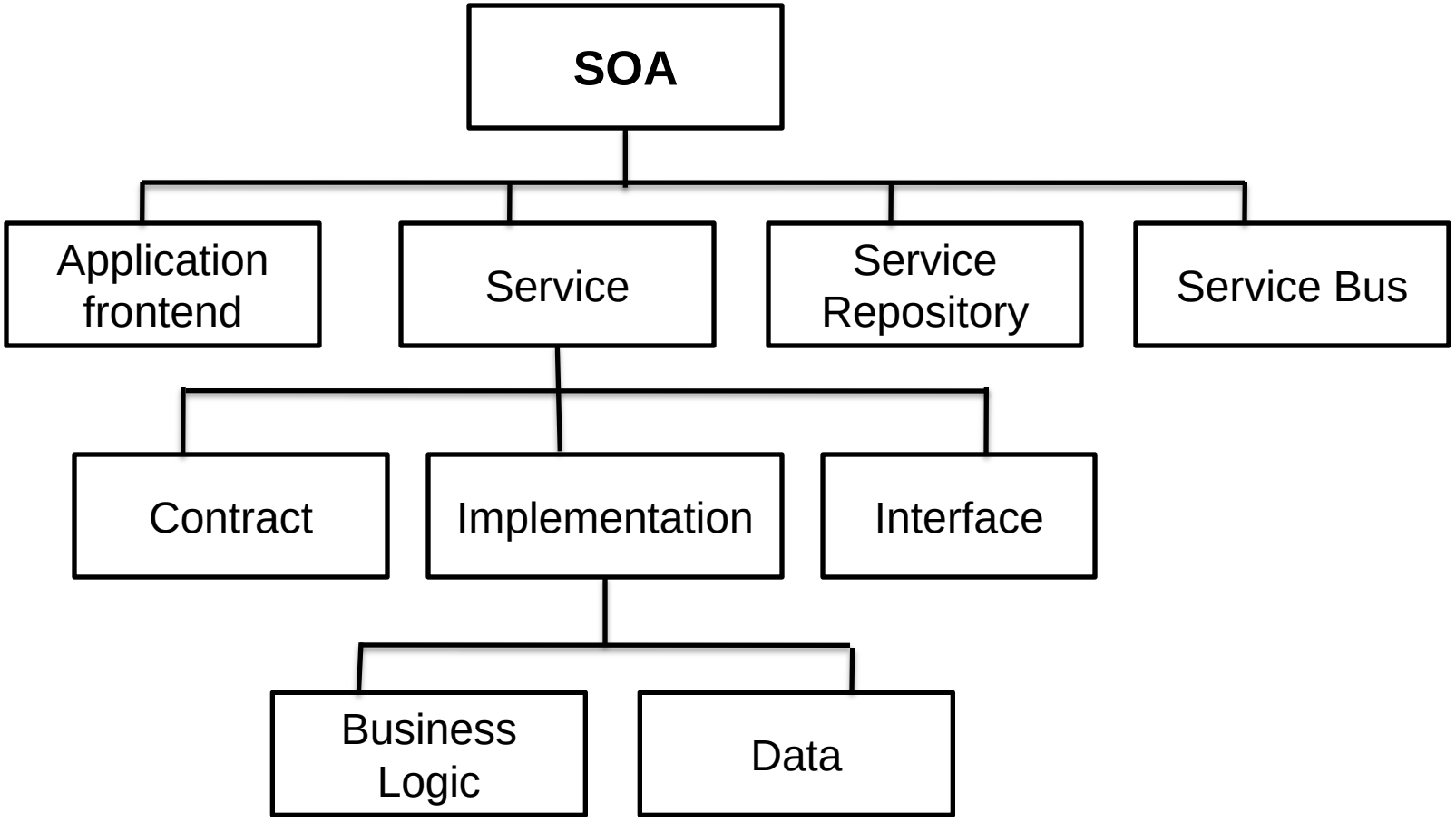
- Application
  - A set of software components that serves a distinctive purpose
- Application landscape
  - The sum of all applications of an organisation.
- Ideally all applications of an application landscape comply with a single architectural blueprint.

Service Oriented Architecture

- An SOA is based on four key abstractions:

1) **Application frontend**
   - Owner of the business process.
2) **Service**
   - Provide business functionality that the application front-end and other services can use.
3) **Service repository**
   - Stores the service contracts of the individual services of an SOA.
4) **Service bus**
   - Interconnects the application frontends and services.

SOA Artifacts



Definition of SOA

- *"A service-oriented Architecture is a software architecture that is based on the key concepts of an application frontend, service, service repository, and service bus. A service consists of a contract, one or more interfaces, and an implementation."* – [3]

- Service (business service):
  - SOA point of view assumes business service rather than technical service.
  - E.g. Making airline reservation, and the business operations are getReservation, cancelBooking, as opposed to technical services like persistency service, transaction service.

Elements of SOA: Application Frontends

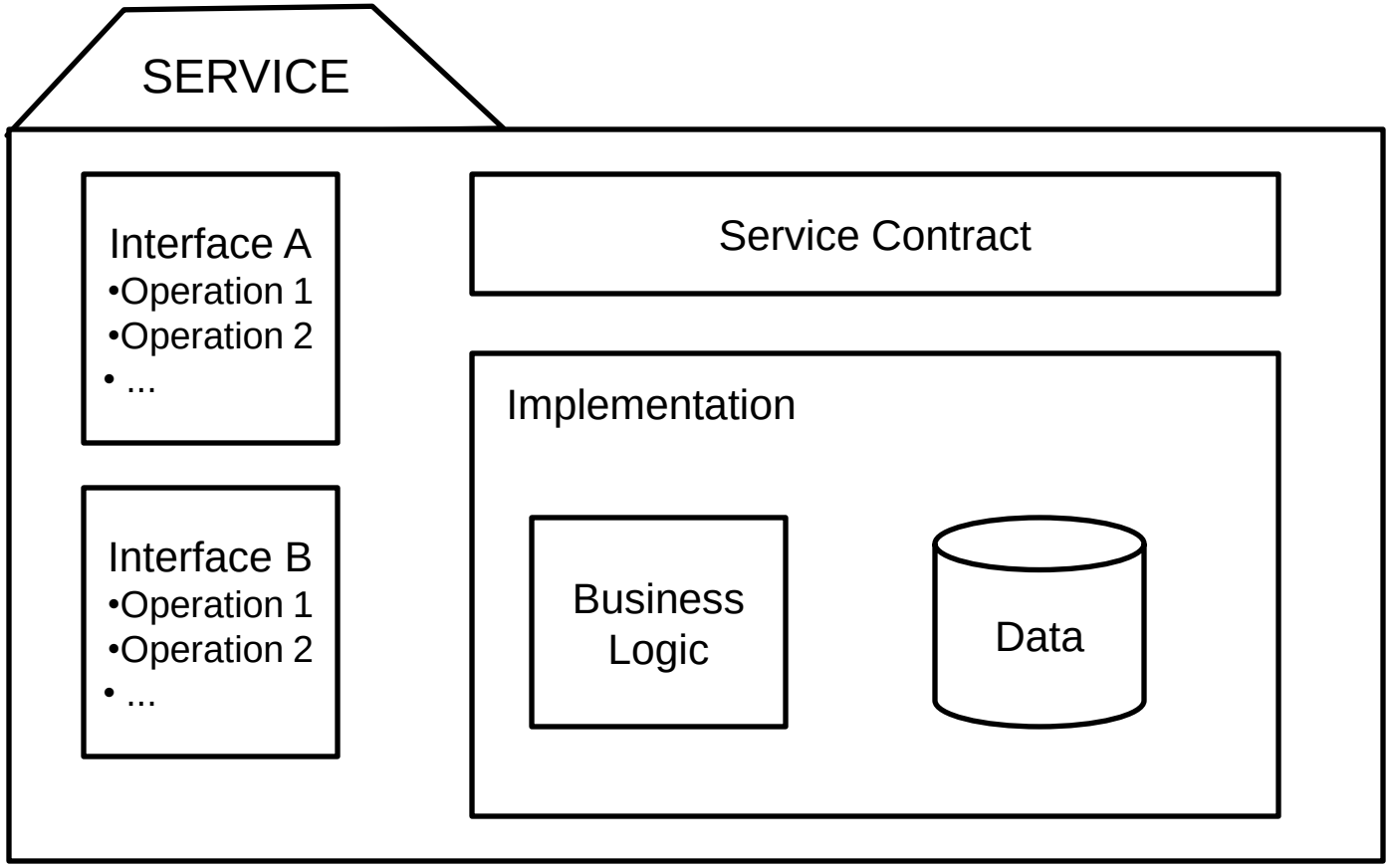
- Application Frontends:
  - Initiate and control all activity of the enterprise system.
  - Different types of application frontends (GUI: web based application or rich client)
  - **Do not necessarily have to interact with end users**, can also be in the form of batch programs or other processes that invoke some functionality periodically or as a result of a specific event.

- Application frontend initiates a business process and receives the results.

Elements of SOA: Services

- A service is a software component of distinctive functional meaning that typically encapsulates a high-level concept.
- Client's perspective: it is a black box entity.
- It basically consists of:
  - A contract.
  - An implementation of the business logic connected to a data source.
  - An Interface.

What is a service made up of ?



Contract

- The service contract provides an **informal specification** of the purpose, functionality, constraints, and usage of the service.
- The form of this specification can vary depending on the type of service. (e.g. Name of service, version, owner, functional specifications, non-functional specifications)
- Formal description with WSDL (Web Service Description Language) or IDL (Interface Definition Language)
  - This is not mandatory, but provides significant benefits.

Interface

- Service interface exposes the functionality of the service to network clients.
- Description of the interface forms part of the service contract.
- Physical implementation of the interface consists of service stubs, which are incorporated in the client (application frontend)

Implementation

- Physical representation of the business logic and data.
- The technical realisation of the business logic which defines the service contract.
- Implementation consists of:
  - programmes,
  - configuration data, and
  - databases.

Elements of SOA: Service Repository

- Provides facilities to discover services and acquire all information to use the services.
- Provides additional information (other than those in the service contract) such as: physical location about the provider, contact persons, usage fees, security issues, and available service level.
- Note that we can build an SOA implementation without establishing a service repository, but it will become indispensible in the long term.

Information in a Service Repository

- Information that an enterprise-wide service repository should contain:
  - Service, operation and argument signatures (e.g. in the form of a WSDL)
  - Service owner
  - Access rights (access list and underlying security mecahnism)
  - Information about the intended performance and scalability of the service. (average response time, throughput levels) – SLA.
  - Transactional properties of the service (read/write/update characteristics)

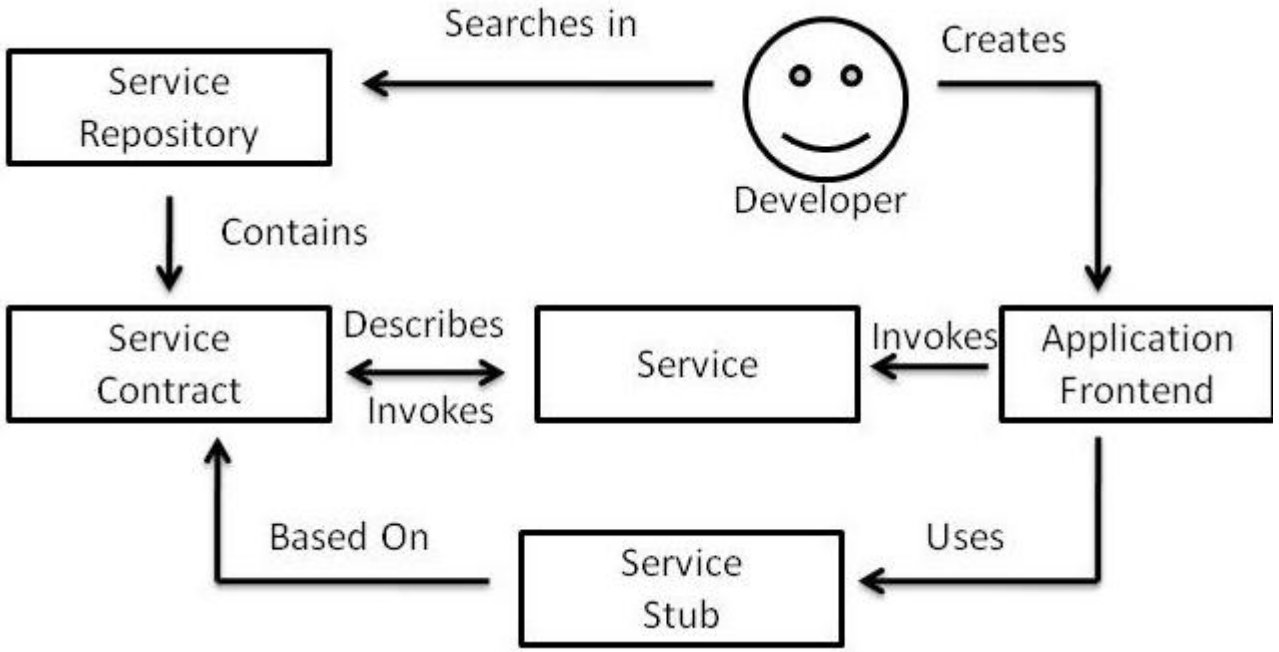
Binding of Services

- Binding refers to the way in which service definitions and service instances are:
  - located,
  - incorporated in the client application, and
  - bound to at network level.

- There are two particular way to bind to a service:
  - At development time
    - The signatures of the operations, the protocol and physical location are know in advance.
  - At runtime
    - More complex, client dynamically binds to the service.

Service Binding – Development Time

1. Development-Time Binding:
   - Developer looks for the service (service repository), and gets the service signature which is then integrated with the client application (frontend).
   - Simple model, sufficient for most purposes (*see below*).



Service Binding - Runtime

2. Runtime Binding:

1. **Runtime service lookup by name**
   - Service definition known at development time, and client is designed accordingly.
   - At runtime, client connects to different service instances and look for services with the specific names.

2. **Runtime service lookup by properties**
   - Same as above, except that service is discovered by properties, rather than names. (e.g. Look for print services, Floor == 2, DocumentType == pdf)

3. **Runtime service discovery based on reflection**
   - Very complex, Specification of service definition is not known at development time.
   - Reflection mechanism must be implemented on client side.

Elements of SOA: Service Bus

- Service Bus connects all participants of an SOA (services and application frontends).
- Allows the client (application frontend) to communicate with (invoke) a service.
- Comprises of a variety of products and concepts (technologies).

The "Bus" Metaphor

- The term service bus is derived from computer bus

  (the wires that connect the CPU, RAM, I/O and so on)

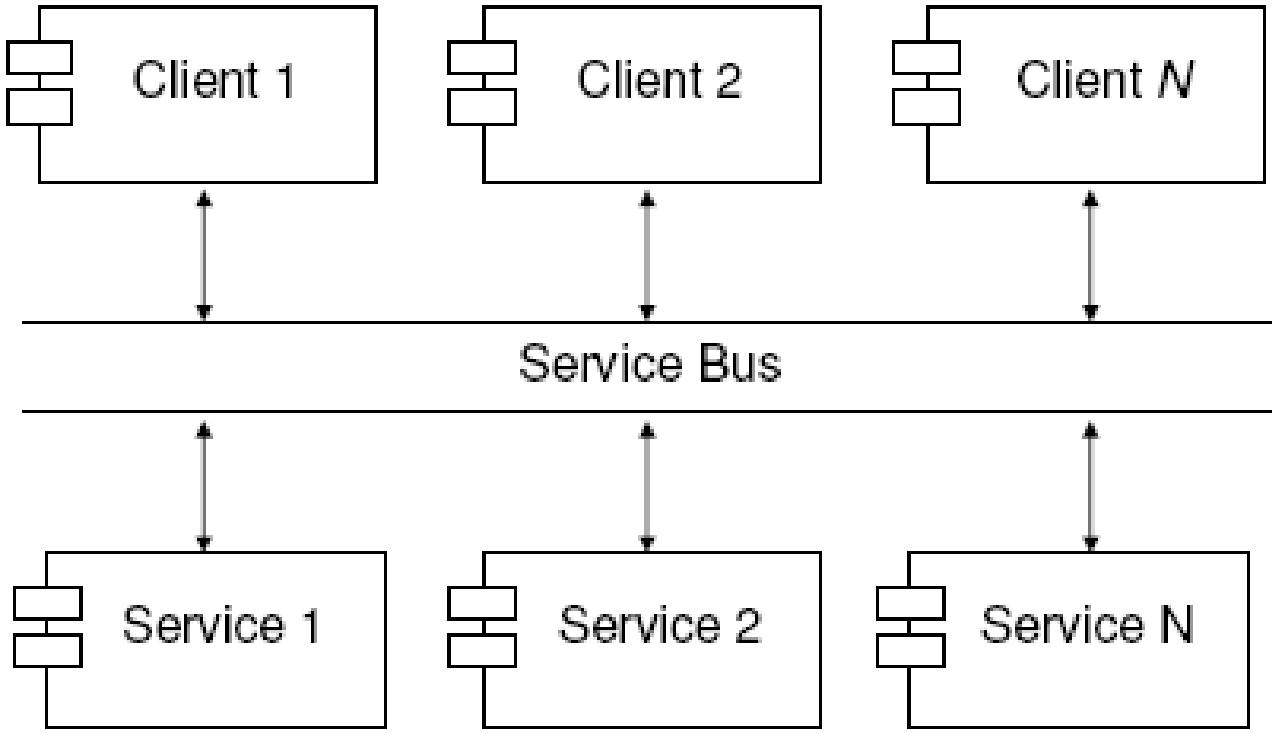
  - Each component provides an interface to the bus, which allows it to communicate indirectly with a large number of different kinds of components.

  - A component X places a message on the bus, and component Y receives it.

- The term service bus is used metaphorically to describe a subsystem that connects clients and servers using the same underlying protocols and technologies.

The "Bus" Metaphor

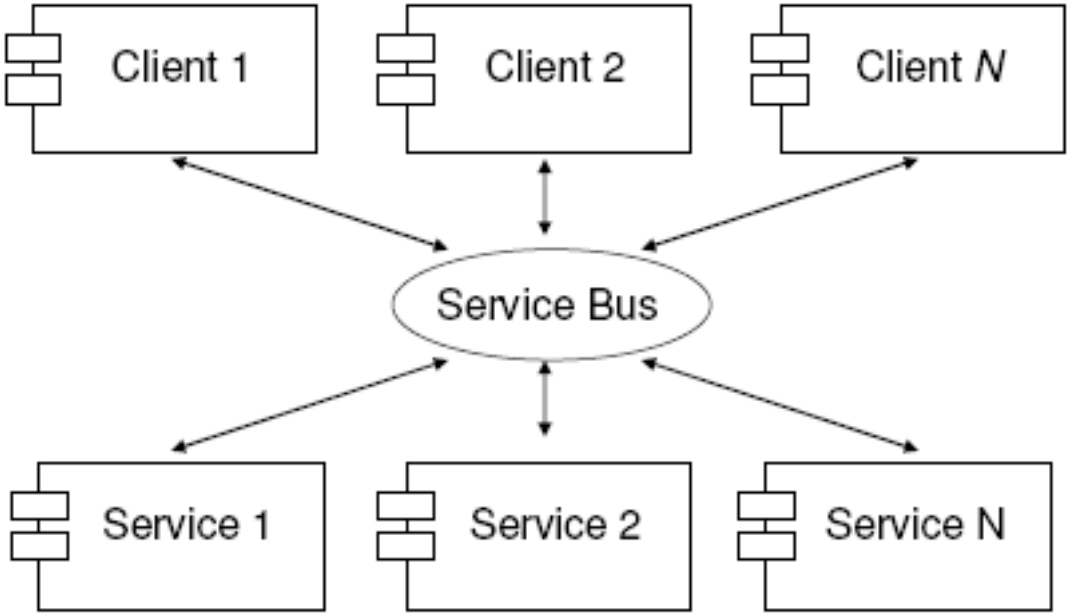
- The "service bus" is a single communication subsystem used by all the clients and services in a SOA.



“Bus” not “Hub”

- Some people *mistakenly (or purposely) interpret the “service* bus” to be a messaging hub.

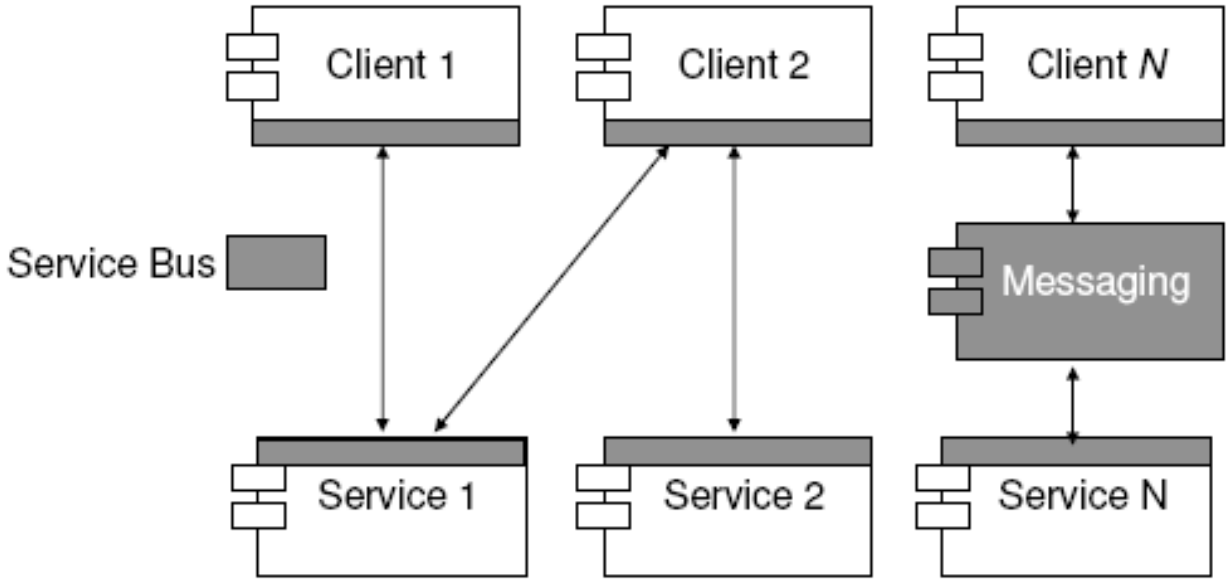
- Hub-spoke architectures are difficult to scale, because the messaging hub becomes a bottleneck.



Distributed Libraries & Daemons

- In reality, a “service bus” is a set of distributed libraries and daemons that facilitate direct *and indirect communication* between clients and services.

- This approach scales well.



Characteristics of Service Bus

- **Connectivity**
  - Primary purpose is to connect the participants.
- **Heterogeneity of Technology**
  - Connect participants based on different programming languages, operating system, or runtime environments.
- **Heterogeneity of Communication Concepts**
  - Supports a variety of Communication concepts (synchronous, asynchronous)
- **Technical “services”**
  - On top of communication: Logging, auditing, security, transaction...

Enterprise Service Bus

- The term ESB lacks formal definition.
  - The term emerged around 2003 and has become synonymous with SOA.
  - It has been hijacked by marketers and vendors and liberally redefined to suit their product sets.

- An ESB is a software product that provides underlying communication infrastructure for software components.
  - The “enterprise” in ESB stresses that the product has features like security and transactions and is suitable for use in a large-scale enterprise.

Conclusion

- Defined the software architecture, and SOA in terms of components and their interactions.
- Acts as blueprint for a system.
- Key Concepts of an SOA:
  - Application frontend – *clients*
  - Services – *basic building blocks*
  - Service Repository – *published and discovered*
  - Service Bus – *communication mechanism*

References

1. B. Len, P. Clements, R. Kazman, *Software Architecture in Practice*, Addision-Wesley, 2003.

2. G. Booch, J. Rumbaugh, I. Jacobson, *Unified Modeling Language User Guide*, Addision Wesley 1999

3. D. Krafzig, K. Banke, D. Slama, *Enterprise SOA, Service-Oriented Architecture Best Practices*, Pearson Education, 2006.

4. Scott W. Ambler, *The Object Primer: Agile Model-Driven Development with UML 2.0*, Cambridge University Press.