Service Oriented Architecture

.NET CORE

Service-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.

SOA – Overview

SOA involves the deployment of services that run in a network. A service has the following characteristics:

- It may handle a business process (calculating an insurance quote) with a specified outcome.
- It may handle a technical task such as accessing a database.
- It is self-contained and independent.
- It can access another service.
- It is a black box for users.

The independence of the service from the other software is called *loose coupling*.
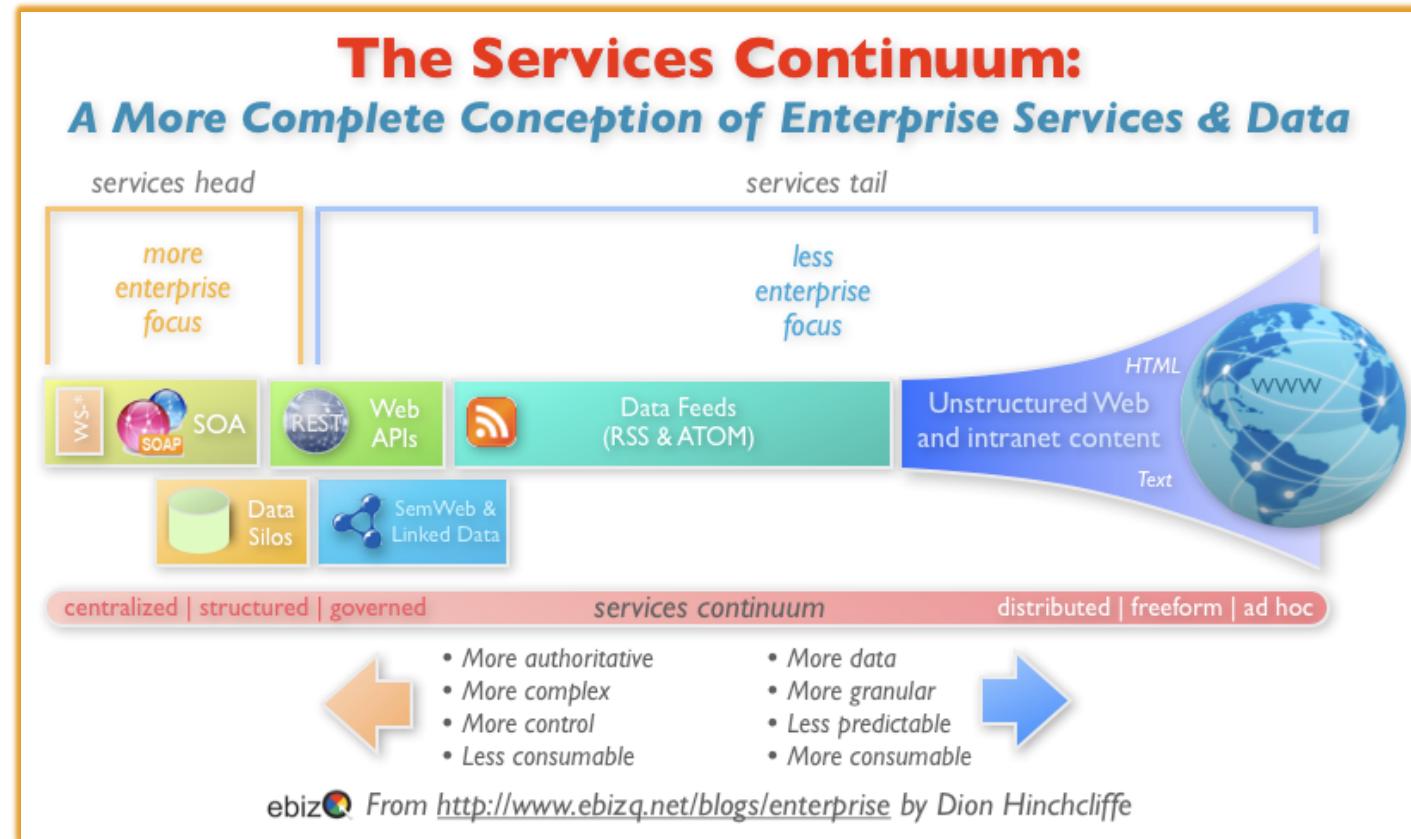Usually, the services are deployed as *Web Services* so they are independent of platforms and programming languages.
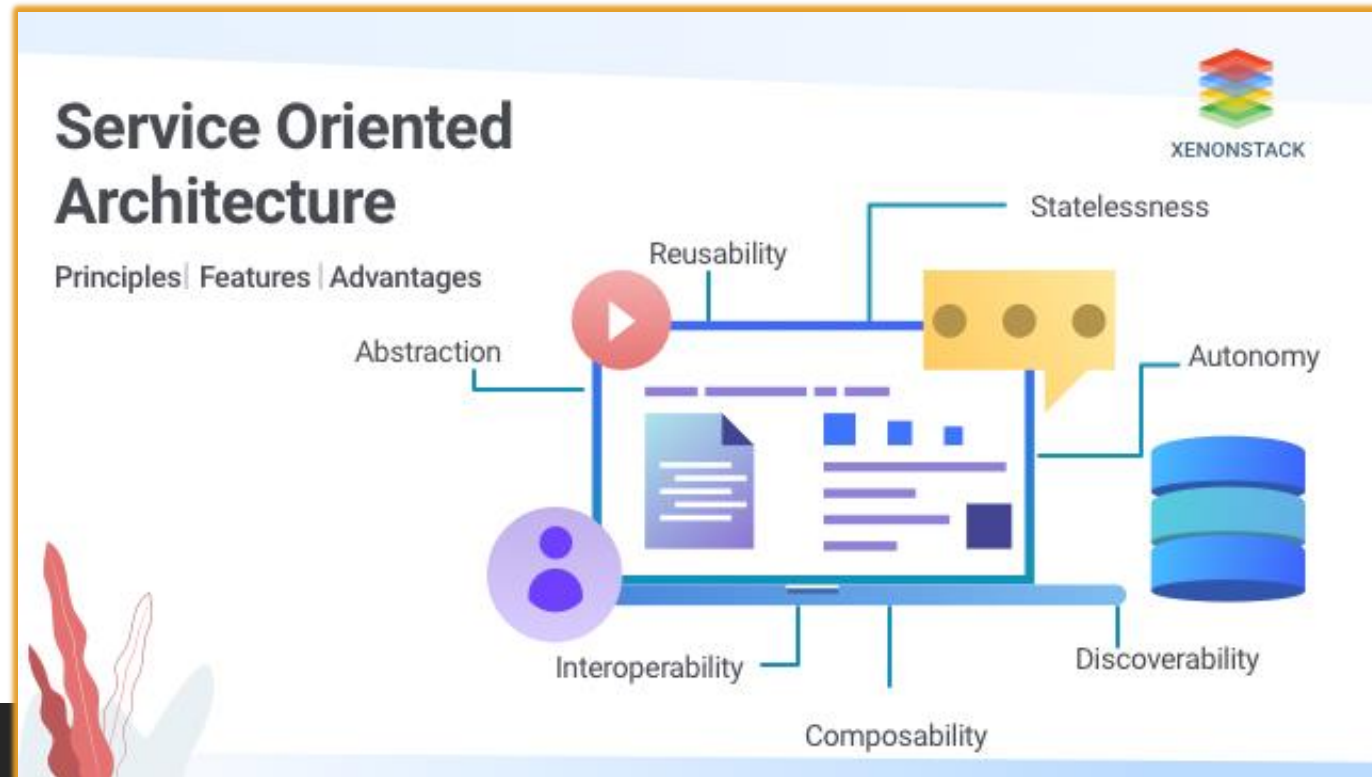
SOA – Principles

SOA can be seen as part of an organizational continuum which ranges from:

- the older concepts of distributed computing and modular programming,
- Then to SOA,
- and on to the offspring of SOA
  - Mashups,
  - SaaS, and
  - cloud computing.

The Services Continuum:
A More Complete Conception of Enterprise Services & Data

services head | services tail

more enterprise focus | less enterprise focus

WS-* | SOAP | SOA | REST | Web APIs | Data Feeds (RSS & ATOM) | Unstructured Web and intranet content | HTML | www | Text

Data Silos | SemWeb & Linked Data

centralized | structured | governed — services continuum — distributed | freeform | ad hoc

- More authoritative
- More complex
- More control
- Less consumable

- More data
- More granular
- Less predictable
- More consumable

ebiz *From http://www.ebizq.net/blogs/enterprise by Dion Hinchcliffe*

SOA – Principles

The six core values of Service-Oriented Architecture are:

1. Business value is given more importance than technical strategy.
2. Strategic goals are given more importance than project-specific benefits.
3. Intrinsic interoperability is given more importance than custom integration.
4. Shared services are given more importance than specific-purpose implementations.
5. Flexibility is given more importance than optimization.
6. Evolutionary refinement is given more importance than pursuit of initial perfection.


SOA Manifesto

modusoperantic

Values

Business value over technical strategy

Strategic goals over project-specific benefits

Intrinsic interoperability over custom integration

Shared services over specific-purpose implementations

Flexibility over optimization

Evolutionary refinement over pursuit of initial perfection

Guiding Principles

- Respect the social and power structure of the organization.
- Recognize that SOA ultimately demands change on many levels.
- The scope of SOA adoption can vary. Keep efforts manageable and within meaningful boundaries.
- Products and standards alone will neither give you SOA nor apply the service orientation paradigm for you.
- SOA can be realized through a variety of technologies and standards.
- Establish a uniform set of enterprise standards and policies based on industry, de facto, and community standards.
- Pursue uniformity on the outside while allowing diversity on the inside.
- Identify services through collaboration with business and technology stakeholders.
- Maximize service usage by considering the current and future scope of utilization.
- Verify that services satisfy business requirements and goals.
- Evolve services and their organization in response to real use.
- Separate the different aspects of a system that change at different rates.
- Reduce implicit dependencies and publish all external dependencies to increase robustness and reduce the impact of change.
- At every level of abstraction, organize each service around a cohesive and manageable unit of functionality.

SOA – Uses

SOA is mainly implemented from the ground up for new services.

There are instances where old code would be converted to SOA architecture.

- The application has UI logic, business processing, and data access tightly coupled making the code difficult to test.
- The application is hard to understand due to repeated patching rather than being rewritten when needed. Updates are time consuming and complex resulting in additional errors.
- The application inventory has duplicate logic, requiring changes in several places.
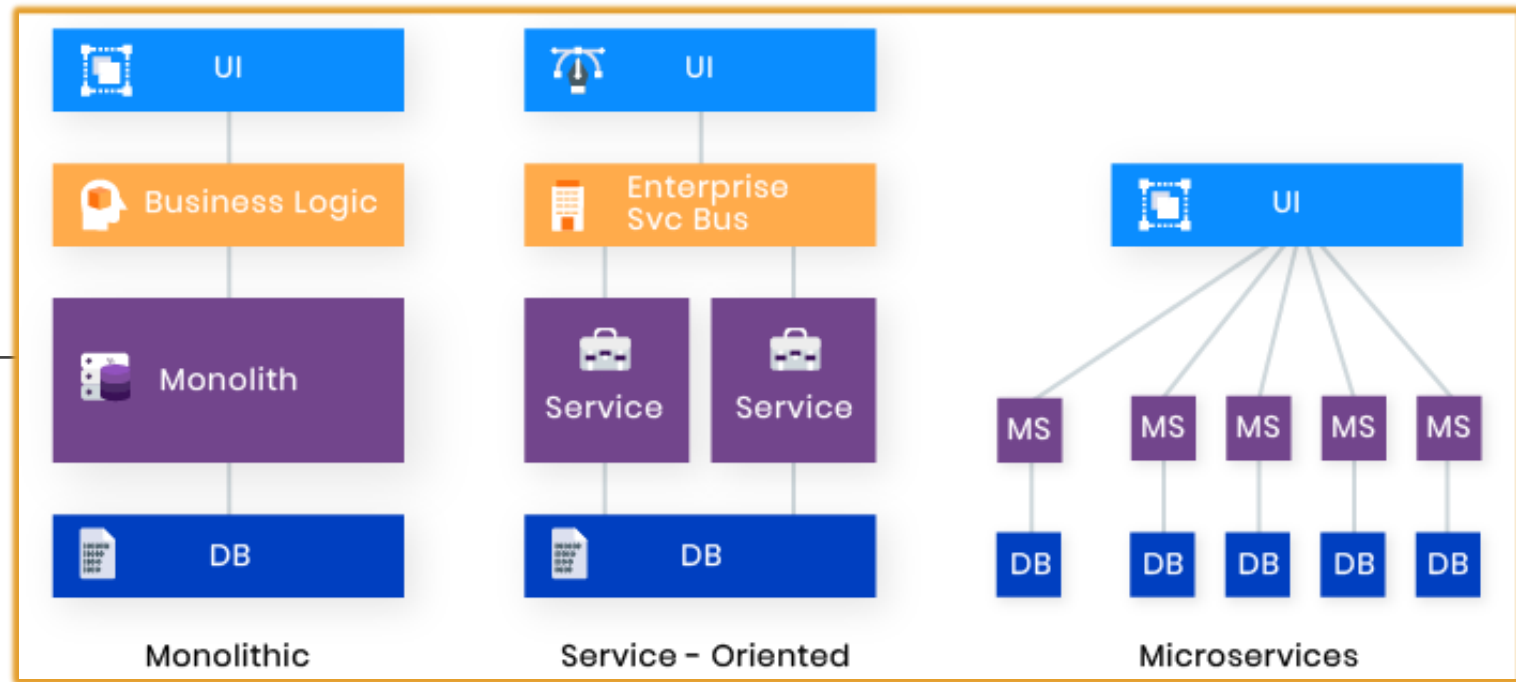
SOA Patterns

A Service-Oriented Application is a collection of independent services organized into a hierarchy.

The topmost level of the SOA receives data from the UI and appeals to the appropriate Business service.

The Business Service might manipulate the data, then appeal to any number of various Data Access Services depending on what result is needed.
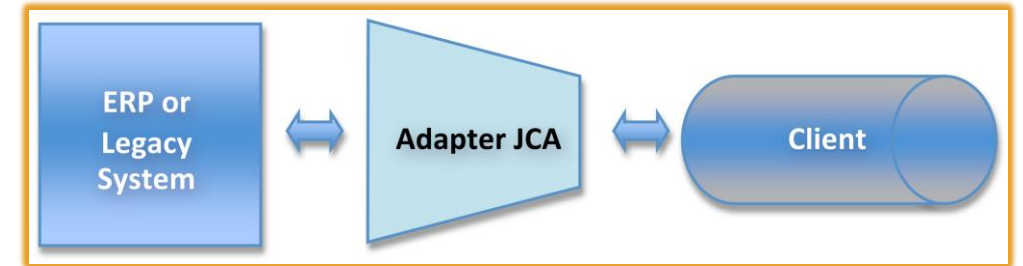
The Data Access Services access the Database for the needed information.
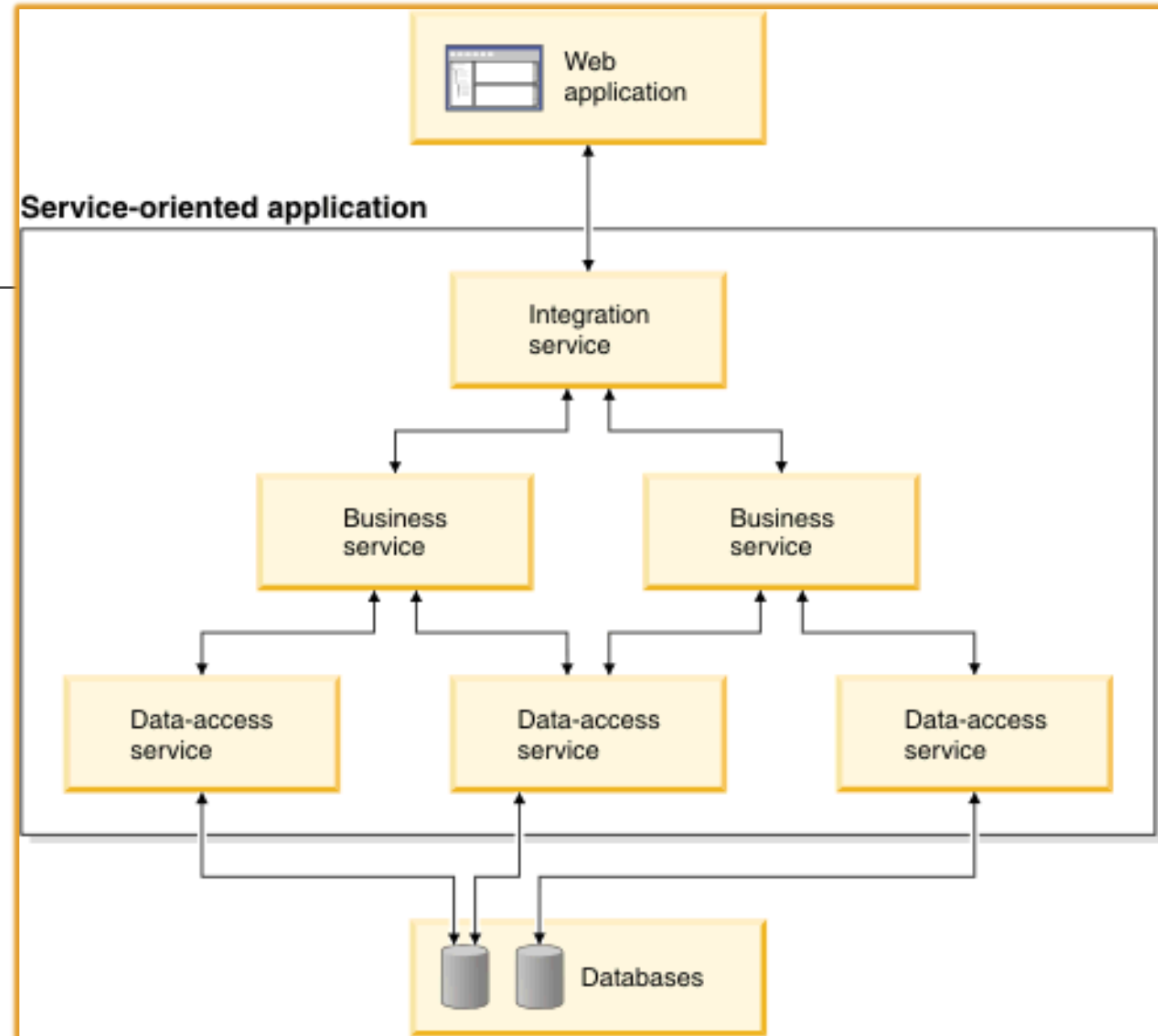
SOA – Web Service Design Patterns

https://www.mercurymagazines.com/pdf/Bst_Prctcs_Dsgn_Ptrns.pdf

- Adapter - Allows the continued use of existing applications by implementing a **wrapper** around them to modify output to what the modern client expects.

- Façade: Placed between the client and the services used. It Loosens the coupling between client and server components.

SOA – Web Service Design Patterns

- Proxy - Used to consolidate the messages sent to separate services into a single service. The *Proxy* dispatches the request to the appropriate back-end service. This simplifies interactions with multiple services.

- Controller - *Controller* is a key component of MVC architecture. A Controller can be used in SOA architectures to encapsulate the *Business Logic* of the target service.
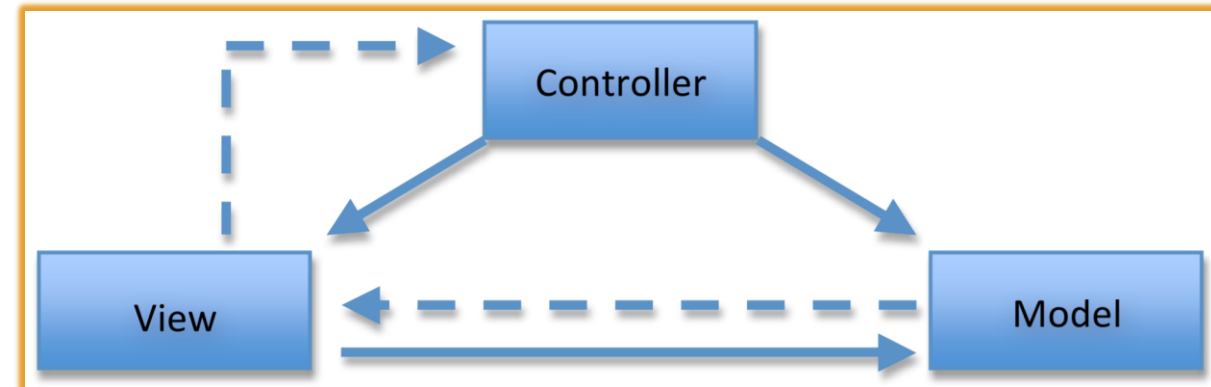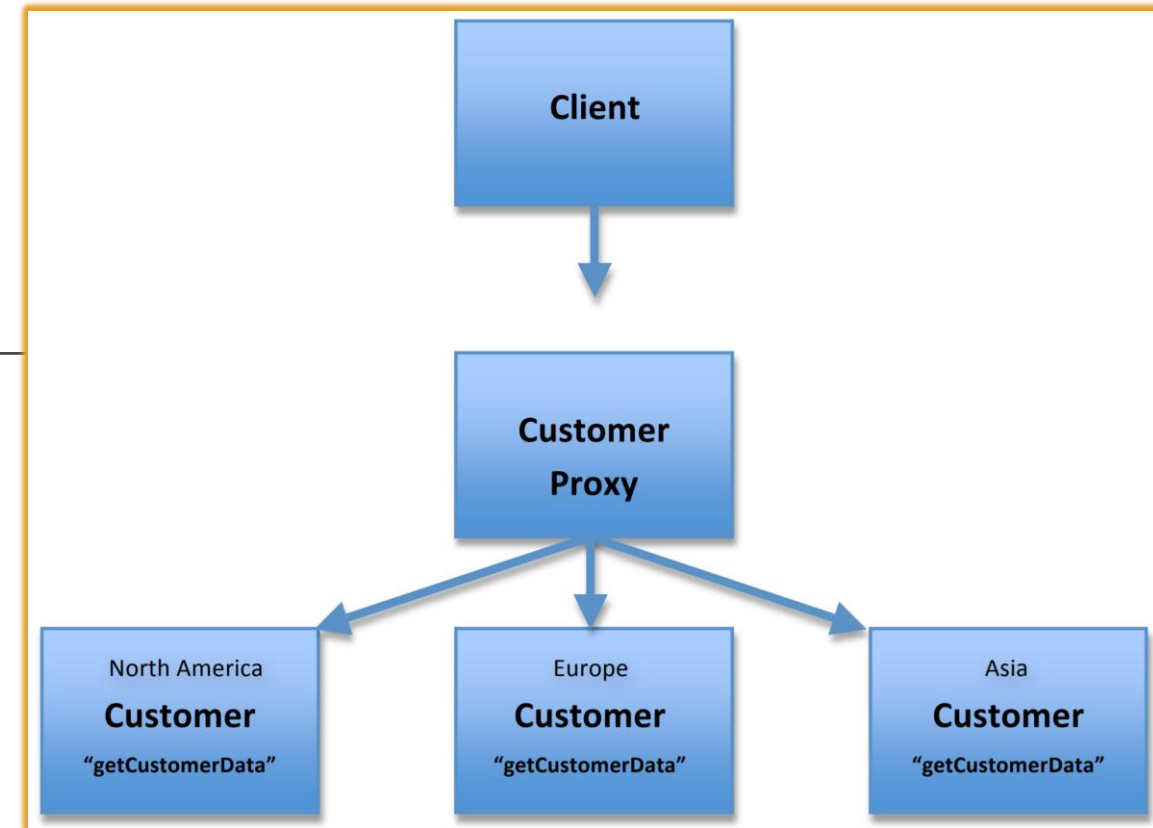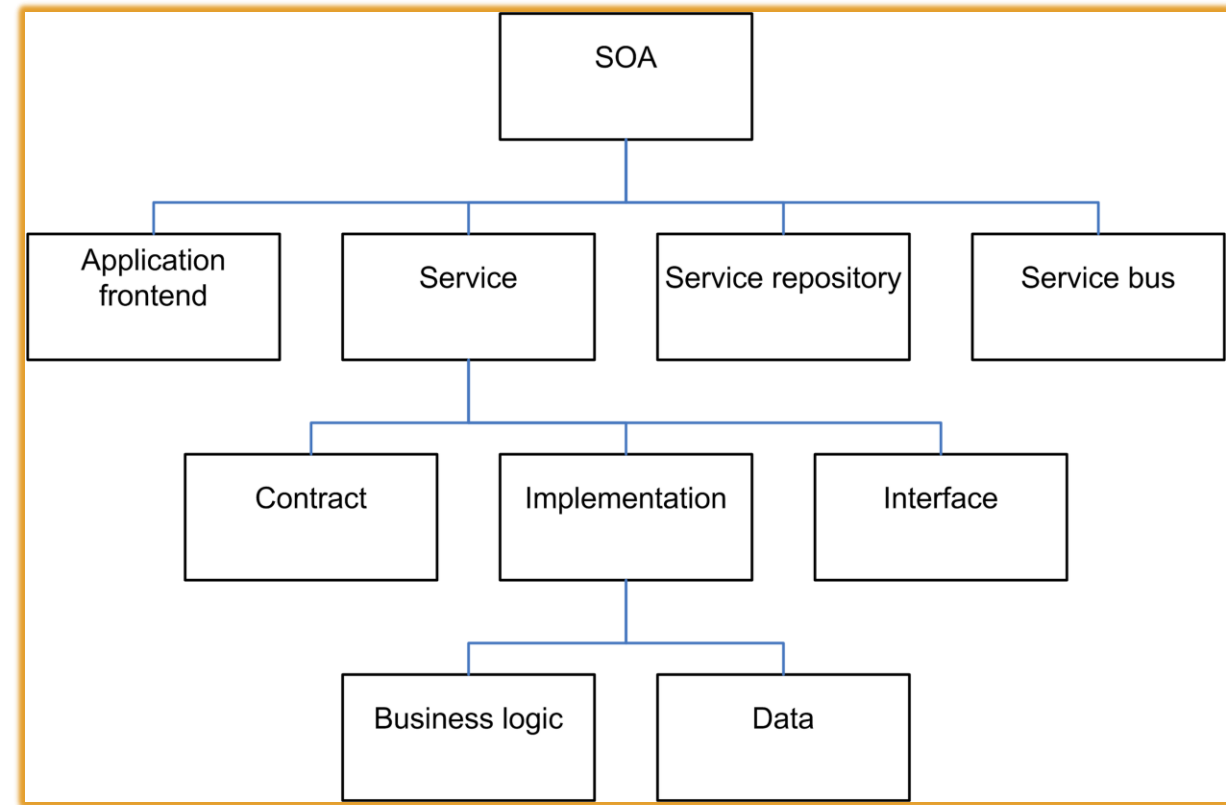
SOA Implementations

SOA's are often implemented using Web Services. The most common Web Service SOA implementations use *SOAP* and *REST* technologies but there are many others.

SOA architectures can implement many different technologies due to their *loosely coupled* nature. The various services comprising a SOA must decide on a protocol for communication. This is often done with a formal document called a *Web Services Description Language (WSDL)* Document.

When using a WSDL, no responding service needs to know anything about the calling service. Each service is a black box

SOA – Benefits

With Service-Oriented Architecture, you can integrate legacy code to with new, mixed technologies to reuse existing applications. This reduces development costs.

Decoupling a service from its presentation reduces expenses and decreases development time. SOA makes applications more dynamic by exposing information and data sharing across the organization and focusing development on the best ways to improve overall operations.

SOA – Challenges

SOA services, being typically *loosely coupled*, exhibit more latency than more *tightly coupled* implementations. This can present challenges with the implementation of real-time, dynamic requirements.

SOA is meant to create an environment where legacy systems can work together, but new business practices like the standardization of naming, definitions, and identification can present implementation challenges. To overcome these challenges, you must implement identity and naming *Services*.