



Service Oriented Architecture

.NET

Service-oriented architecture (SOA) is a software design pattern where functionalities are provided by one API to another API through a predetermined communication protocol (HTTP) over a network.

[HTTPS://EN.WIKIPEDIA.ORG/WIKI/SERVICE-ORIENTED_ARCHITECTURE](https://en.wikipedia.org/wiki/Service-oriented_architecture)

SOA – Overview

https://www.ibm.com/support/knowledgecenter/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/pegl_serv_overview.html#pegl_serv_overview_introsoa

https://en.wikipedia.org/wiki/Service-oriented_architecture#Patterns

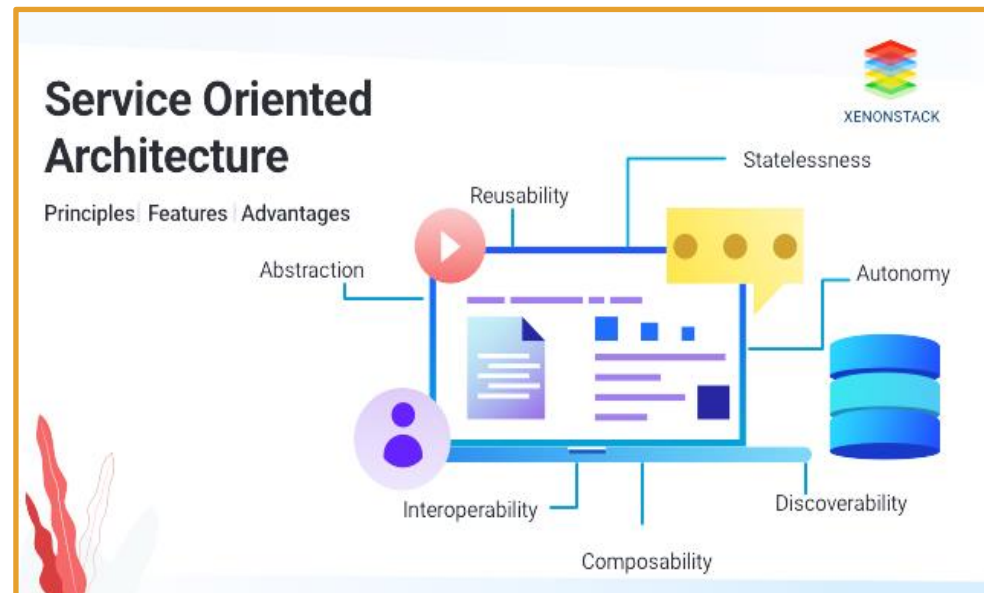
<https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf#features-of-wcf>

Service-oriented architecture (SOA) is defined by a reliance on “services” to send and receive data. **SOA** involves the deployment of services that run in a network. A service has the following characteristics:

- It handles a business process with a specified outcome.
- It may handle a technical task such as accessing a database.
- It is self-contained and independent.
- It can access other services (other APIs).
- It is a black box for users.

The independence of the service from other software is called **loose coupling**.

Usually, the services are deployed as **Web Services** so they are independent of platforms and programming languages.



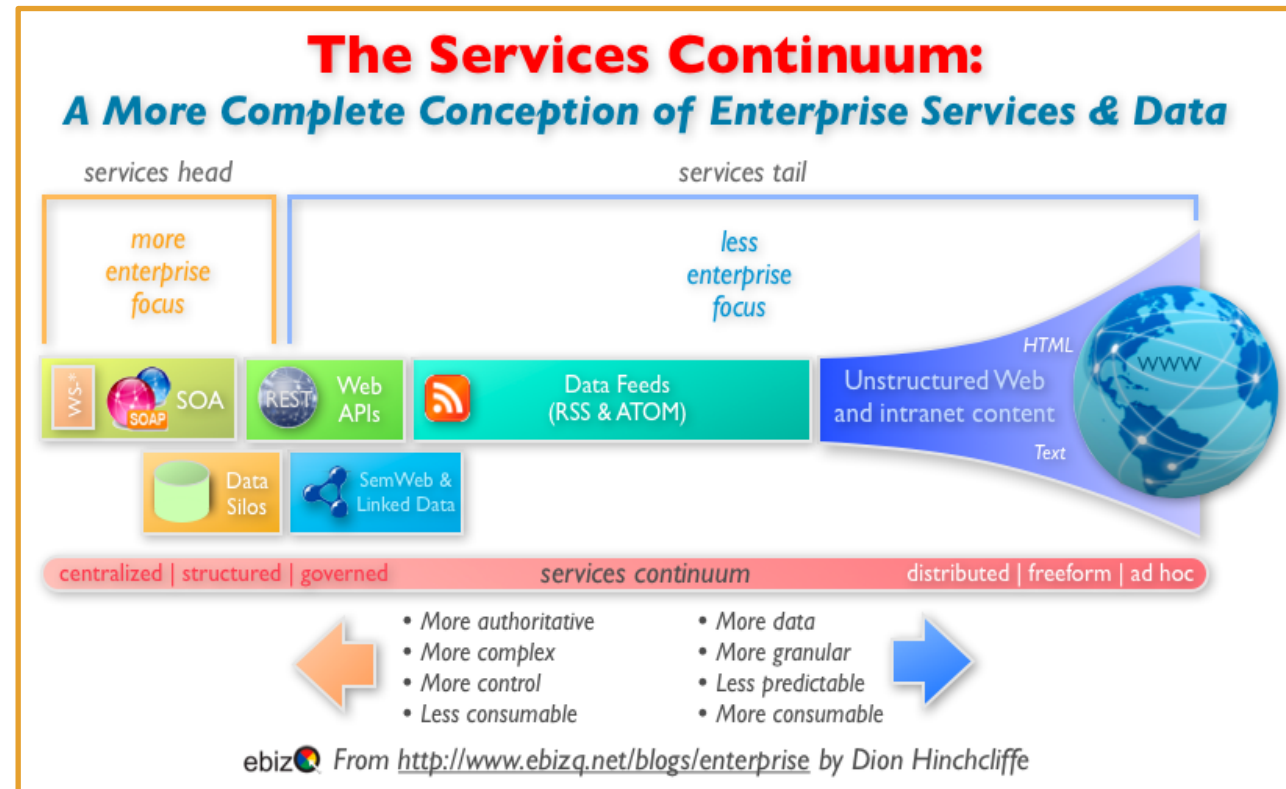
SOA – Overview

https://en.wikipedia.org/wiki/Service-oriented_architecture#Patterns

SOA is part of an organizational continuum.

This continuum:

1. Begins at the (older) concepts of distributed computing and modular programming.
2. Then moves to SOA.
3. Then moves on to the offspring of SOA, such as:
 - Mashups,
 - SaaS, and
 - cloud computing.



SOA – Principles

https://en.wikipedia.org/wiki/Service-oriented_architecture#Patterns

<https://www.modusoperantic.com/en/the-soa-manifesto/>

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.
 1. “Perfect” is the enemy of “DONE”.



Client/Server Principle

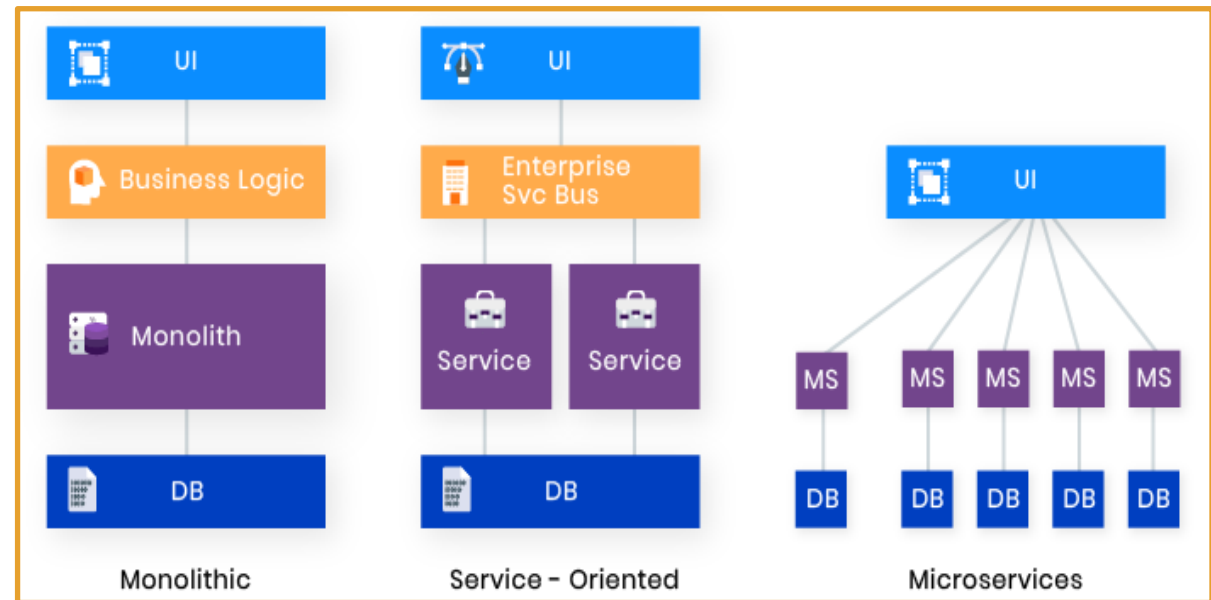
TODO

SOA – Uses

https://www.ibm.com/support/knowledgecenter/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/pegl_serv_overview.html#pegl_serv_overview_introsoa

SOA is mainly implemented from the ground up on new services. Justifications for when old code might be converted to SOA architecture are if the application:

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



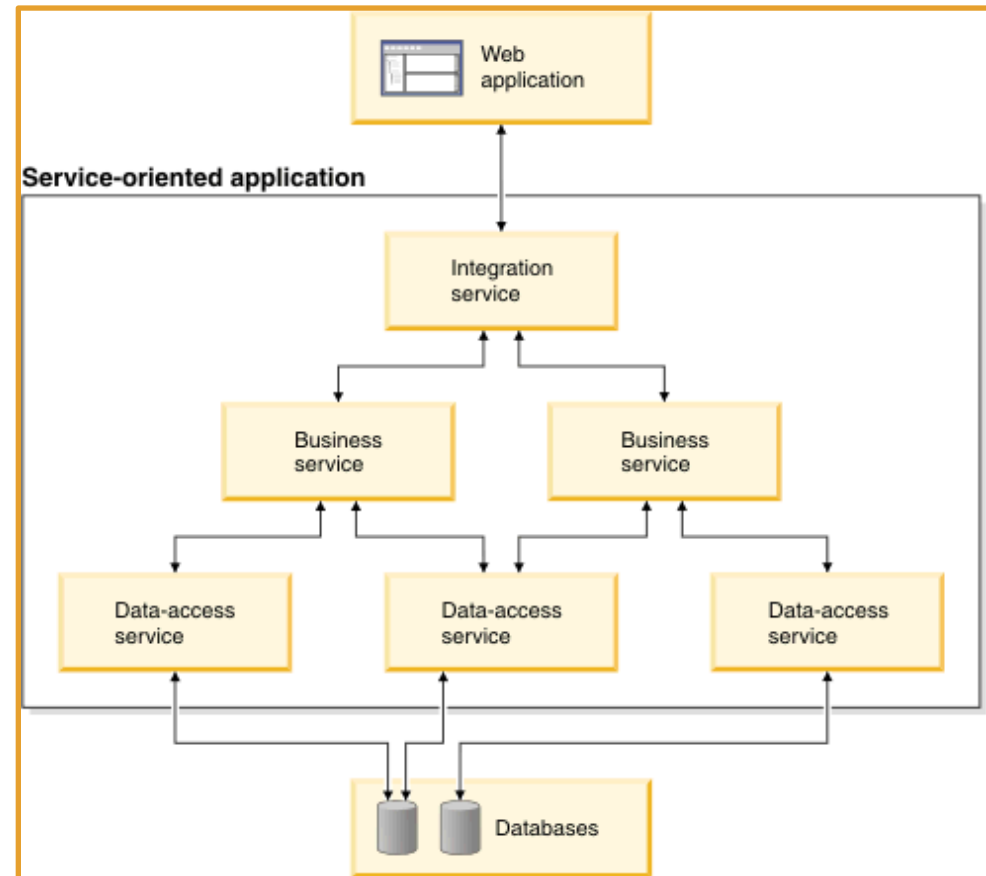
SOA – Design Patterns

ibm.com/support/knowledgecenter/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/egl_serv_overview.html#egl_serv_overview_introsoa

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 (the user) and sends requests via HTTP to the appropriate Business service API.

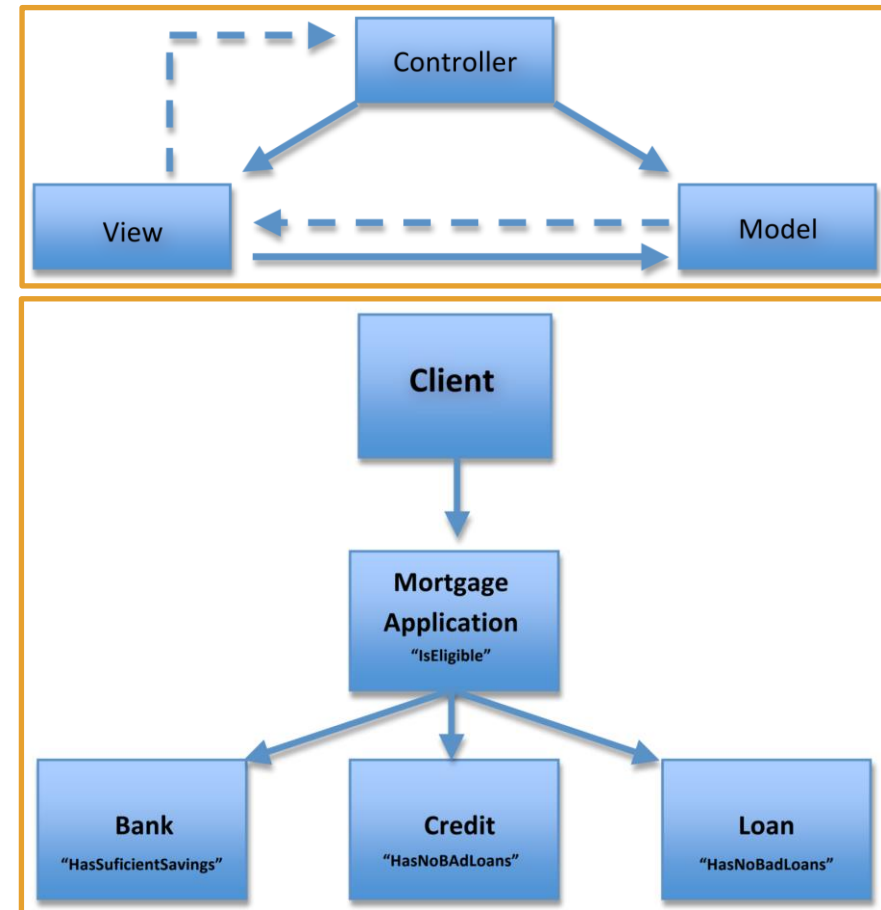
The Business Service may manipulate the data or send another request to a Data Access Services, or access a Database, depending on what is needed.



SOA – Web Service Design Patterns

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

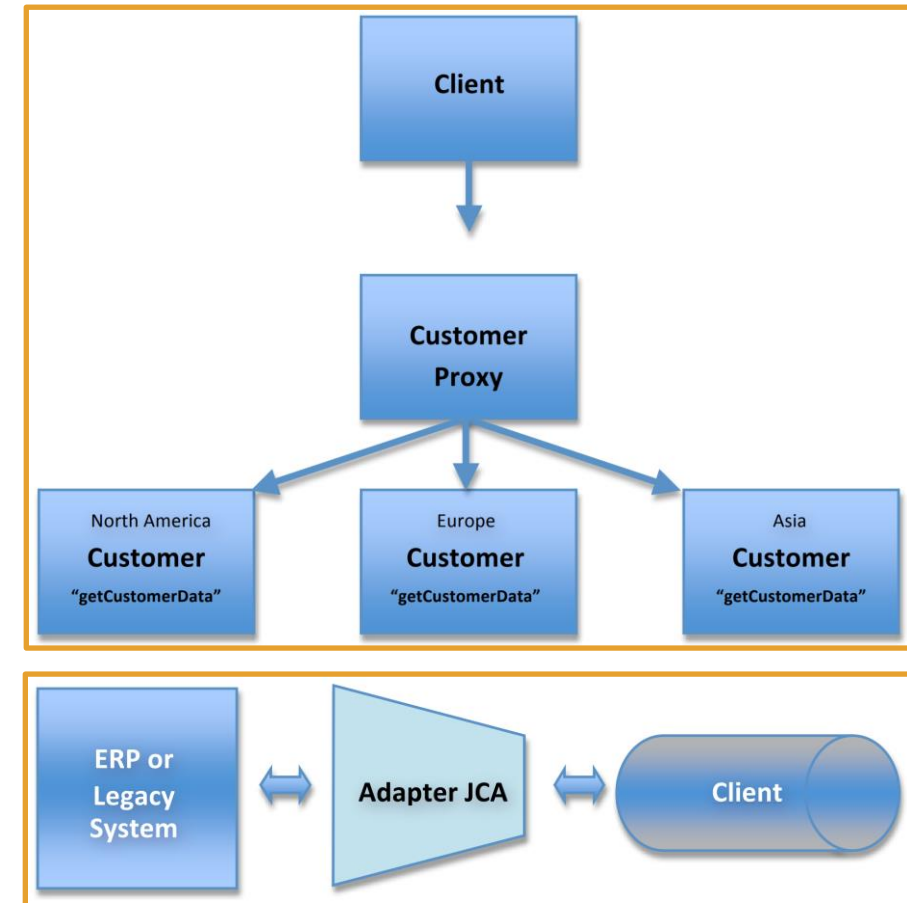
- Controller - A Controller can be used in SOA architectures to encapsulate the **Business Logic** of the target service. **Controller** is a key component of “MVC” architecture.
- Façade: Placed between the client and the services used. It Loosens the coupling between client and server components.



SOA – Web Service Design Patterns

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

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



SOA - Implementations

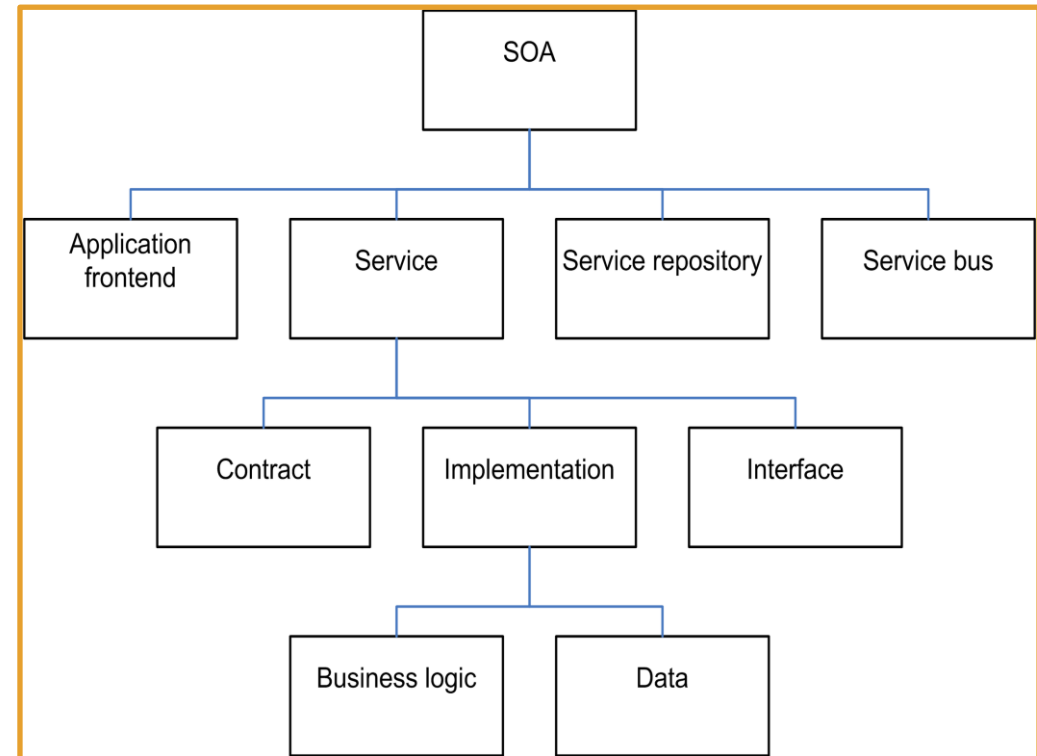
https://en.wikipedia.org/wiki/Service-oriented_architecture#Patterns

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's can make use of many different technologies due to their **loosely coupled** nature. The various services comprising a SOA must only decide on a protocol for communication.

This is often done (especially with SOAP) with a formal document called a **Web Services Description Language (WSDL)** Document.

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



SOA – Benefits

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

With Service-Oriented Architecture, you can integrate legacy code with new 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 strategies to improve overall operations.



SOA – Difficulties

https://www.mercurymagazines.com/pdf/Bst_Prctcs_Dsgn_Ptrns.pdf
<https://martinfowler.com/bliki/ServiceOrientedAmbiguity.html>

- **SOA services** are typically *loosely coupled*. Due to this loose coupling, they have more latency than *tightly coupled* implementations. This can present challenges when implementing dynamic requirements.
- **SOA** is meant to create an environment where legacy systems can work with new systems, but new business practices like the standardization of naming, definitions, and identification can present implementation challenges.
- To overcome the challenge of blending legacy applications with new standards, **services** can be implemented to handle certain tasks for the legacy application.