Service-Oriented Architecture

A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

Service-Oriented Architecture (SOA) is an architectural approach in which applications make use of services available in the network. In this architecture, services are provided to form applications, through a communication call over the internet.

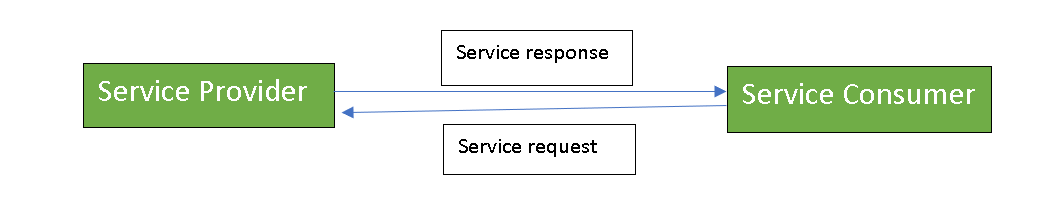
* SOA allows users to combine a large number of facilities from existing services to form applications.
* SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
* SOA based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

**Service**

A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

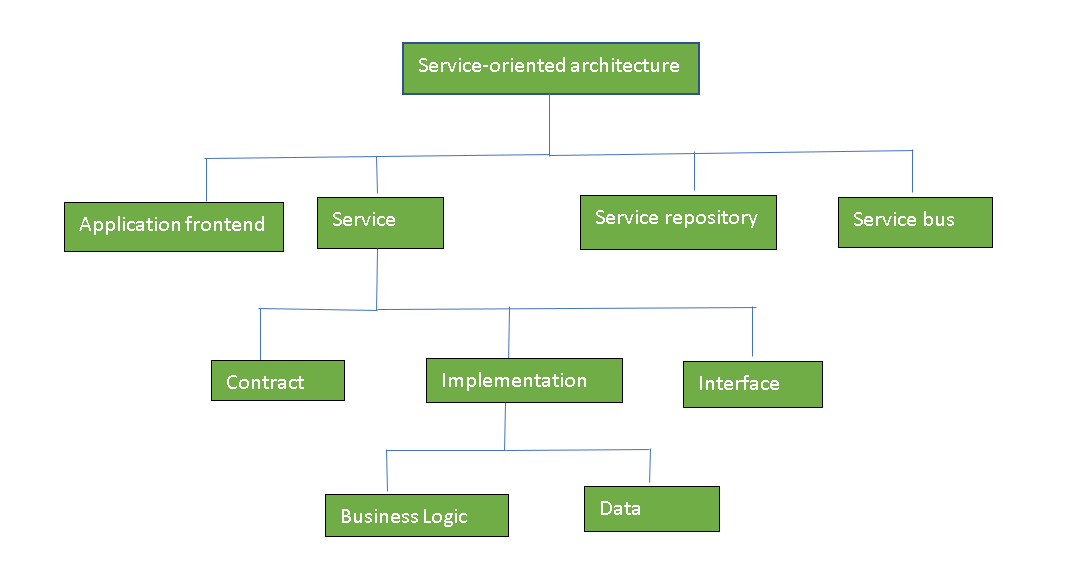
There are two major roles within Service-oriented Architecture:

1. **Service provider:** The service provider is the maintainer of the service and the organization that makes available one or more services for others to use. To advertise services, the provider can publish them in a registry, together with a service contract that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.
2. **Service consumer:** The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.



Services might aggregate information and data retrieved from other services or create workflows of services to satisfy the request of a given service consumer. This practice is known as service orchestration Another important interaction pattern is service choreography, which is the coordinated interaction of services without a single point of control.

**Components of SOA:**



**Guiding Principles of SOA:**

1. **Standardized service contract:**Specified through one or more service description documents.
2. **Loose coupling:** Services are designed as self-contained components, maintain relationships that minimize dependencies on other services.
3. **Abstraction:** A service is completely defined by service contracts and description documents. They hide their logic, which is encapsulated within their implementation.
4. **Reusability:** Designed as components, services can be reused more effectively, thus reducing development time and the associated costs.
5. **Autonomy:** Services have control over the logic they encapsulate and, from a service consumer point of view, there is no need to know about their implementation.
6. **Discoverability:** Services are defined by description documents that constitute supplemental metadata through which they can be effectively discovered. Service discovery provides an effective means for utilizing third-party resources.
7. **Composability:** Using services as building blocks, sophisticated and complex operations can be implemented. Service orchestration and choreography provide a solid support for composing services and achieving business goals.

**Advantages of SOA:**

* **Service reusability:** In SOA, applications are made from existing services.Thus, services can be reused to make many applications.
* **Easy maintenance:** As services are independent of each other they can be updated and modified easily without affecting other services.
* **Platform independant:** SOA allows making a complex application by combining services picked from different sources, independent of the platform.
* **Availability:** SOA facilities are easily available to anyone on request.
* **Reliability:** SOA applications are more reliable because it is easy to debug small services rather than huge codes
* **Scalability:**Services can run on different servers within an environment, this increases scalability

**Disadvantages of SOA:**

* **High overhead:** A validation of input parameters of services is done whenever services interact this decreases performance as it increases load and response time.
* **High investment:** A huge initial investment is required for SOA.
* **Complex service management:** When services interact they exchange messages to tasks. the number of messages may go in millions. It becomes a cumbersome task to handle a large number of messages.

**Practical applications of SOA:** SOA is used in many ways around us whether it is mentioned or not.

1. SOA infrastructure is used by many armies and air force to deploy situational awareness systems.
2. SOA is used to improve the healthcare delivery.
3. Nowadays many apps are games and they use inbuilt functions to run. For example, an app might need GPS so it uses inbuilt GPS functions of the device. This is SOA in mobile solutions.
4. SOA helps maintain museums a virtualized storage pool for their information and content.

**Implementation of RESTful**

HTTP GET

HTTP POST

HTTP PUT

HTTP DELETE

HTTP PATCH

Summary

Glossary

## **HTTP GET**

Use GET requests **to retrieve resource representation/information only** – and not to modify it in any way. As GET requests do not change the state of the resource, these are said to be **safe methods**.

#### Example request URIs

* HTTP GET http://www.appdomain.com/users
* HTTP GET http://www.appdomain.com/users?size=20&page=5

## **HTTP POST**

Use POST APIs **to create new subordinate resources or insert new record/update**. Talking strictly in terms of REST, POST methods are used to create a new resource into the collection of resources.

#### Example request URIs

* HTTP POST http://www.appdomain.com/users
* HTTP POST http://www.appdomain.com/users/123/accounts

## **HTTP PUT**

Use PUT APIs primarily **to update existing resource** (if the resource does not exist then API may decide to create a new resource or not).

#### Example request URIs

* HTTP PUT http://www.appdomain.com/users/123
* HTTP PUT http://www.appdomain.com/users/123/accounts/456

## **HTTP DELETE**

As the name applies, DELETE APIs are used **to delete resources** (identified by the Request-URI).

A successful response of DELETE requests SHOULD be HTTP response code 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has been queued, or 204 (No Content) if the action has been performed but the response does not include an entity.

#### Example request URIs

* HTTP DELETE http://www.appdomain.com/users/123
* HTTP DELETE http://www.appdomain.com/users/123/accounts/456

## **HTTP PATCH**

* HTTP PATCH requests are **to make partial update on a resource**. Or command to replace an entity.

Simple Patch request

HTTP PATCH /users/1

[  
{ “op”: “replace”, “path”: “/email”, “value”: “new.email@example.org” }  
]