INTRODUCTION TO SERVICE ORIENTED ARCHITECTURE & WEB SERVICES

Lesson Outline

1.1 Understanding web services.

1.2 Advantages and Disadvantages of web services.

1.3 Understanding service-oriented architecture.

1.4 Advantages and Disadvantages of service oriented architecture.

SERVICE ORIENTED ARCHITECTURE (SOA)

A Service Oriented Architecture (SOA) is an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network.

Why do you need a Web Service?

Modern day business applications use variety of programming platforms to develop web-based applications. Some applications may be developed in Java, others in .Net, while some other in Angular JS, Node.js, etc

These heterogeneous applications need some sort of communication to happen between them.

Web services provide a common platform that allows multiple applications built on various programming languages to have the ability to communicate

Source: https://www.guru99.com/web-service-architecture.html

WEB SERVICE

A Web service is similar in that it is accessed via a URL. The difference lies in the content of what is sent in the request from the client to the service. Web service clients send an XML document formatted in a special way in accordance with the rules of the *SOAP specification*.

A SOAP message can contain a call to a method along with any parameters that might be needed. In addition, the message can contain a number of header items that further specify the intent of the client.

These header items might designate what web services will get this method call after the current service finishes its work, or they might contain security information.

WEB SERVICE

Amazon Web Services or Google's Web service, Google Cloud Platform.

Addressing the need for application development

Offering modern web services possible

WEB PAGE VS WEB SERVICE

Feature	Web Page	Web Service
Purpose	To **display content** to users in a browser	To **exchange data** between systems or applications
Output Format	**HTML**, CSS, images, JavaScript	**XML**, **JSON**, or other machine-readable formats
Accessed by	**Humans** using web browsers	**Programs** or other systems
Example	`https://example.com/products.html`	`https://api.example.com/GetProducts`
Interaction	Users click, scroll, view images, fill forms	Applications send/receive data via requests/responses
Technology Used	HTML, CSS, JavaScript, possibly PHP/ASP.NET backend	REST, SOAP, WSDL, XML/JSON, APIs
Visibility	Viewable in any browser	Not directly viewable in browser unless formatted
User Interface	Yes – designed for people to interact with	No – designed for other software to use

RELATIONSHIP BETWEEN **SOA** AND **WEB SERVICES**

SOA is an architecture; Web Services are a technology.

SOA is an architectural style where software components (services) provide functionality to other components via well-defined interfaces and protocols.

Web Services are a specific implementation technology that allows applications to communicate over the web using standard protocols like HTTP, SOAP, REST, and XML/JSON.

RELATIONSHIP BETWEEN **SOA** AND **WEB SERVICES**

Relationship

Web Services are one way to implement SOA.SOA can be realized using different technologies, but Web Services are the most common and popular. Web Services provide the platform-independent mechanism SOA needs for components to interact.

CHARACTERISTICS OF WEB SERVICES

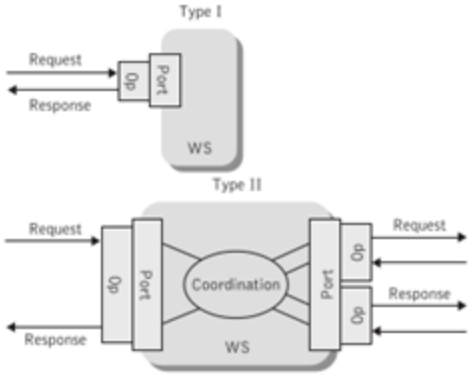
Types of Web services

Web services can come in two flavors, Informational, or type I,

Simple (Type I)

Web services, which support only simple request/response operations and always wait for a request; they process it and respond.

Handle one-way communication—like a client sending a request and getting a response. For example, a weather API that returns the forecast when you sen location.

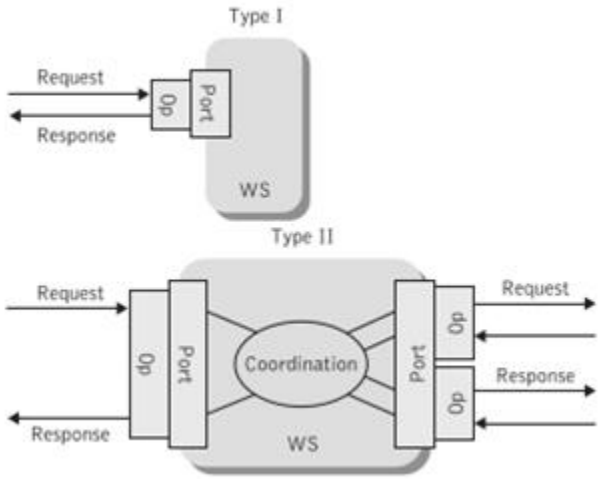


CHARACTERISTICS OF WEB SERVICES

Types of Web services

Complex, or type II Web services implement some form of coordination between inbound and outbound operations.

These involve coordinated interactions—not just simple request-response.



CHARACTERISTICS OF WEB SERVICES

Functional and non-functional properties

Services are described in terms of a description language.

A service description has **two major interrelated components**: its functional and non-functional characteristics.

The functional description details the operational characteristics that define the **overall behavior** of the service, i.e., defines details of how the service is invoked, the location where it is invoked, and so on. This description focuses on details regarding the syntax of messages and how to configure the network protocols to deliver messages.

The non-functional description concentrates on **service quality attributes**, such as service metering and cost, performance metrics, e.g., response time or accuracy, security attributes, authorization, authentication, (transactional) integrity, reliability, scalability, and availability.

CHARACTERISTICS OF WEB SERVICES

State properties

Services could be stateless or stateful.

If services can be invoked repeatedly without having to maintain context or state, they are called stateless,

While services that may require their context to be preserved from one invocation to the next are called stateful.

The services access protocol is always connectionless. A connectionless protocol means that the protocol has no concept of a job or a session and does not make any assumptions about eventual delivery.

CHARACTERISTICS OF WEB SERVICES

Loose coupling

Web services interact with one another dynamically and use Internet standard technologies, making it possible to build bridges between systems that otherwise would require extensive development efforts. The term coupling indicates the degree of dependency any two systems have on each other.

Type II Web Services are like conductors of an orchestra. Not just handle one-off requests—they manage a sequence of actions, coordinating between what they receive and what they send out.

e-commerce, banking, logistics, and enterprise systems where multiple systems must work together to complete a business task.

CHARACTERISTICS OF WEB SERVICES

Service granularity

Web services may vary in function from simple requests to complex systems that access and combine information from multiple sources. Even simple service requests may have complicated realizations. Simple services are discrete in nature, exhibit normally a request/reply mode of operation, and are of fine granularity, i.e., they are atomic in nature. In contrast, complex services are coarse grained.

A coarse-grained description of a system regards large subcomponents.

A fine-grained description regards smaller components of which the larger ones are composed.

ADVANTAGES OF WEB SERVICES

Interoperability (Connecting Different Applications)

Web Services allow different applications to talk to each other and share data and services among themselves. Other applications can also use the services of the web services. For example, VB or .NET applications can talk to java web services and vice versa. So, Web services are used to make the application platform and technology independent.

ADVANTAGES OF WEB SERVICES

Versatility

Web services are also versatile by design. They can be accessed by humans via a Web-based client interface, or they can be accessed by other applications and other Web services. A client can even combine data from multiple Web services.

E.g. Application to update sales, shipping, and ERP systems from one unified interface – even if the systems themselves are incompatible. Because the systems exchange information via Web services, a change to the sales database, for example, will not affect the service itself.

Enterprise Resource Planning (ERP)

ADVANTAGES OF WEB SERVICES

Re-usability

One service might be utilized by several clients, all of which employ the operations provided to fulfill different business objectives. Instead of having to create a custom service for each unique requirement, portions of a service are simply re-used as necessary.

Standardized Protocol

Web Services uses a standardized industry-standard protocol for communication. All four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use the well-defined protocol in the Web Services protocol stack. This standardization of the protocol stack gives the business many advantages like a wide range of choices, reduction in the cost due to competition, and increase in quality.

ADVANTAGES OF WEB SERVICES

Cost Saving

All these benefits add up to significant cost savings. Based on interoperability, possible to create highly customized applications for integrating data, in an inexpensive mode.

Existing investments in systems development and infrastructure can be utilized easily and combined to add additional value.

Since Web services are based on open standards their cost is low and the associated learning curve is smaller than that of many proprietary solutions.

DISADVANTAGES OF WEB SERVICES

Although the simplicity of Web services is an advantage in some respects, it can also be an obstruction.

Web services use plain text protocols that use a fairly verbose method to identify data. This means that Web service requests are larger than requests encoded with a binary protocol. The extra size is really only an issue over low-speed connections, or over extremely busy connections.

DISADVANTAGES OF WEB SERVICES

Security and Privacy—Anything sent over the Internet can be viewed by others.

At present, the only approved option for sending sensitive information to Web services is to use a Secure Socket Layer (SSL) program. SSL over HTTP is sometimes called HTTPS.

This technology is wonderful in that it does a good job of safeguarding information such as credit card numbers, and it is easy to set up compared to other encryption techniques. The disadvantage is that SSL is slower than unencrypted HTTP transactions and is therefore unsuited to large data transfers.

RELATION OF SERVICE ORIENTED ARCHITECTURE AND WEB SERVICES

Most enterprises have made extensive investments in system resources over the course of many years. Such enterprises have an enormous amount of data stored in legacy enterprise information systems (EIS), so it's not practical to discard existing systems. It's more cost-effective to evolve and enhance EIS. But how can this be done? Service Oriented Architecture (SOA) provides a cost-effective solution.

SOA is not a new concept. Sun defined SOA in the late 1990's to describe Jini, which is an environment for dynamic discovery and use of services over a network. Web services have taken the concept of services introduced by Jini technology and implemented it as services delivered over the web using technologies such as XML, Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery, and Integration(UDDI).

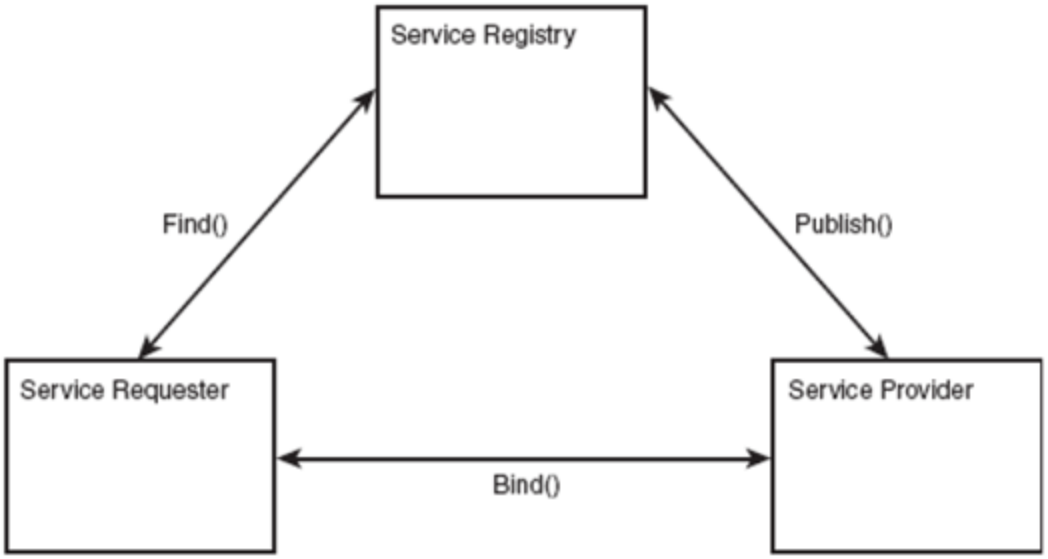
https://www.oracle.com/technical-resources/articles/javase/soa.html

SERVICE ORIENTED ARCHITECTURE (SOA)

SOA is a logical way of designing a software system to provide services to either end-user applications or to other services distributed in a network, via published and discoverable interfaces.

The main building blocks of an SOA

The service provider, the service registry, and the service requester (client).



SERVICE ORIENTED ARCHITECTURE (SOA)

The Service Requester

This is any consumer of the web service. The requester utilizes an existing web service by opening a network connection and sending an XML request. It doesn't know where the web service is or how to locate it, so it turns to the Service Registry and requests a Find() operation.

The Service Registry

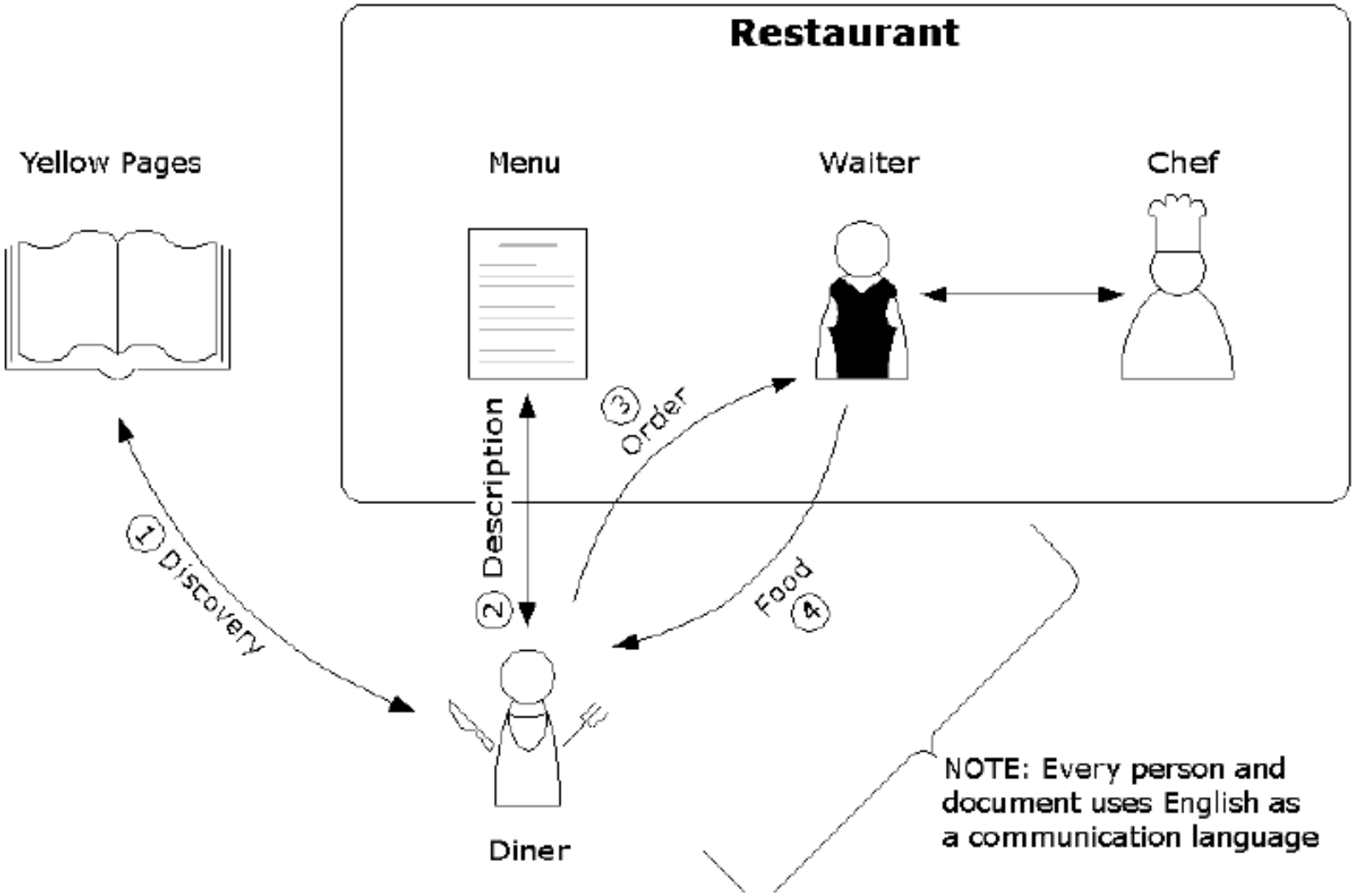
This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It, therefore, serves as a centralized clearinghouse for companies and their services. It responds to search criteria that has been submitted by a Service Requester in the Find() operation. This information returns the contact information for Web services based on their classifications that match the search criteria. It also includes information about how to find out the connection details.

SERVICE ORIENTED ARCHITECTURE (SOA)

The Service Provider

This is the provider of the web service. The Service Providers Publish() these classification details as well as the connection details to the Service Registry. The Service Requester uses the connection details to Bind() to the Service Provider. Once bound, they can send messages and receive responses

WEB SERVICES COMMUNICATION FUNDAMENTALS



INTRODUCTION TO MAJOR COMPONENTS OF THE ARCHITECTURE

Hypertext Transport Protocol (HTTP)

The purpose of HTTP is to provide a protocol to move requests and responses between clients and servers. It will carry any information that is placed in it from point A to point B without regard for its data type.

HTTP State Management Mechanism

The HTTP state management mechanism allows clients and servers that wish to exchange state information to place HTTP requests and responses within a larger context, which is called a session. This mechanism enables HTTP servers to store state (called cookies) at HTTP user agents, letting the servers maintain a stateful session over the mostly stateless HTTP protocol.

INTRODUCTION TO MAJOR COMPONENTS OF THE ARCHITECTURE

Simple Object Access Protocol (SOAP)

SOAP is an XML-based messaging protocol. It defines a set of rules for structuring messages that can be used for simple one-way messaging but it is particularly useful for performing RPC-style (Remote Procedure Call) request-response dialogues.

Universal Description, Discovery, and Integration (UDDI)

The Universal Description, Discovery, and Integration (UDDI) is an XML-based standard for describing, publishing, and finding Web services. It uses special classification schemes called taxonomies that categorize a Web service in ways that are meaningful to potential clients.

INTRODUCTION TO MAJOR COMPONENTS OF THE ARCHITECTURE

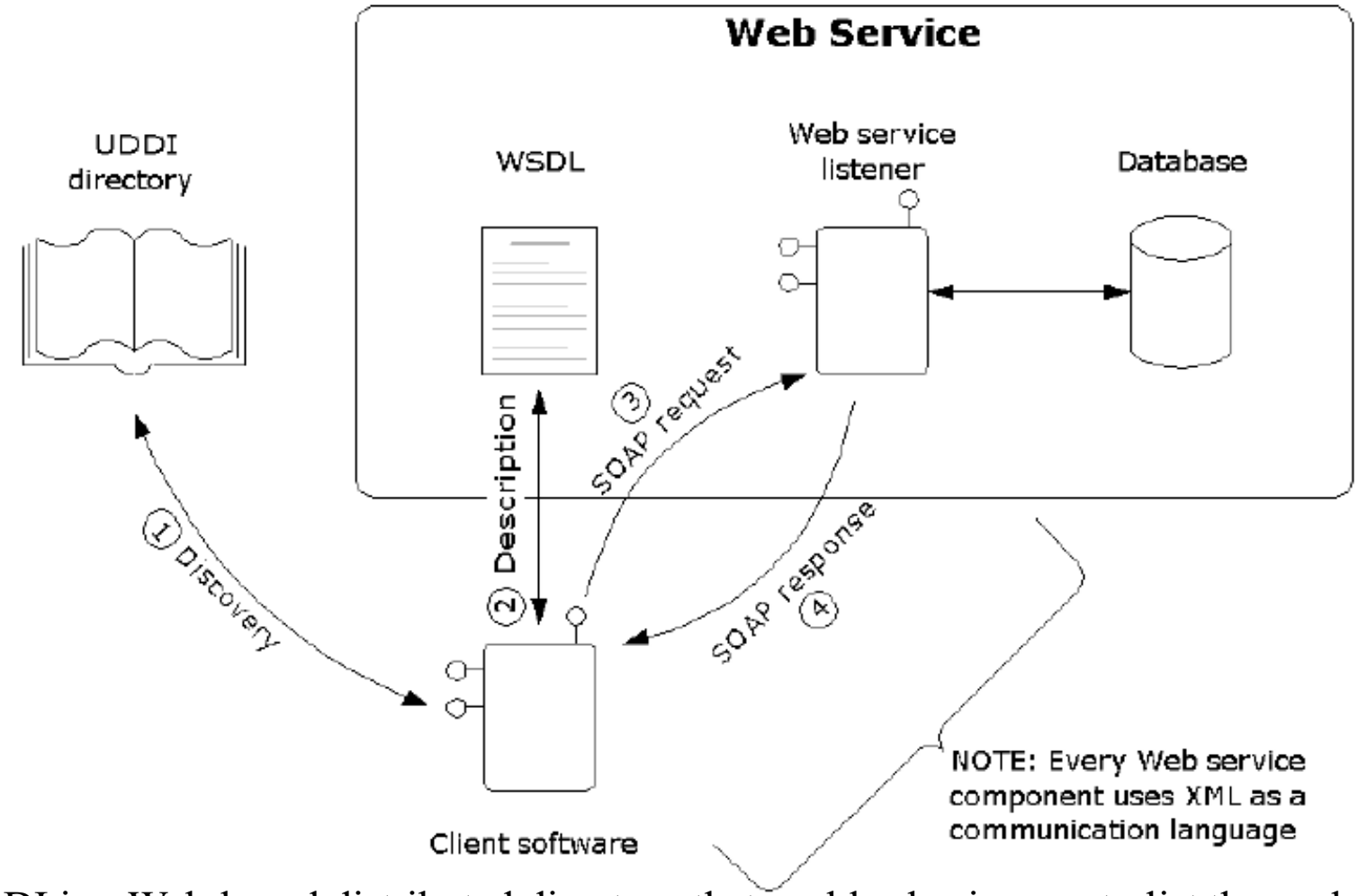
Web Services Description Language (WSDL)

Web Services Description Language (WSDL) is a standard specification for describing networked, XML-based services. It provides a simple way for service providers to describe the basic format of requests to their systems regardless of the underlying run-time implementation.

Extensible Markup Language (XML)

XML is a simple, very flexible text format derived from SGML. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

WEB SERVICES COMMUNICATION FUNDAMENTALS

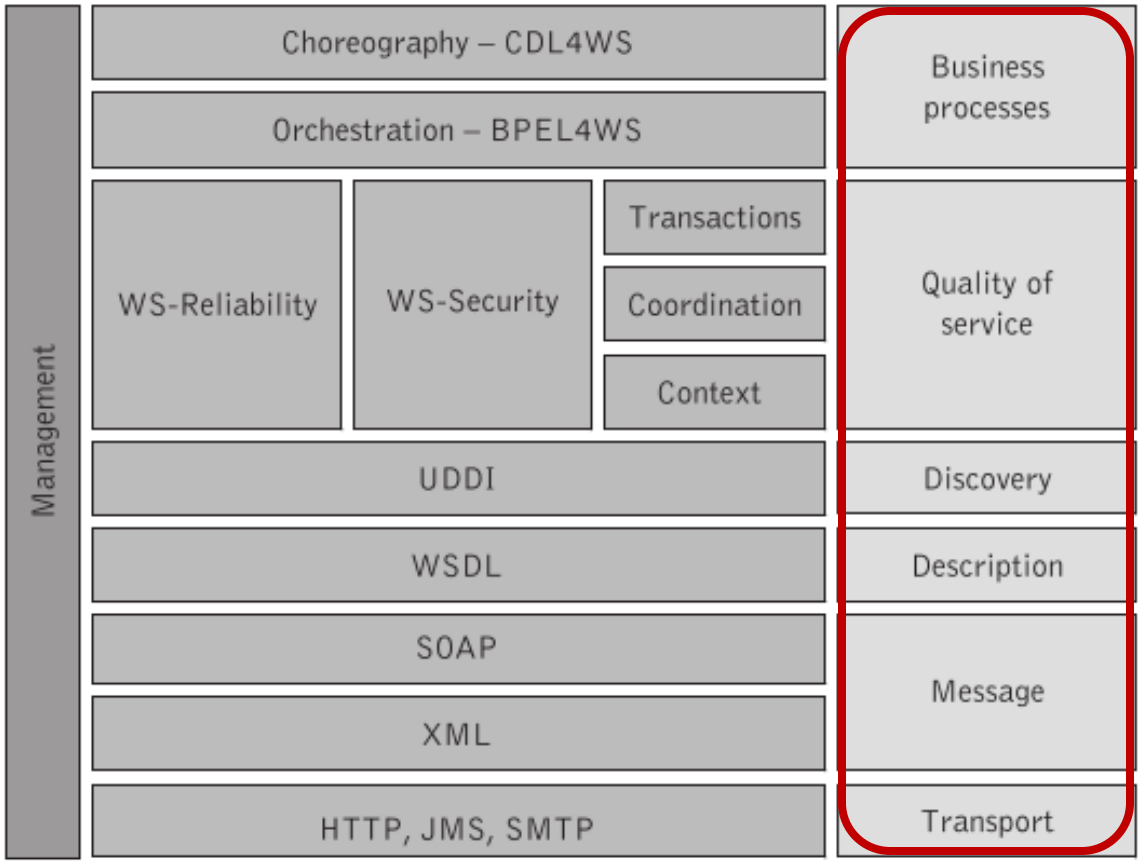


UDDI is a Web-based distributed directory that enables businesses to list themselves on the Internet (or Intranet) and discover each other

THE WEB SERVICES TECHNOLOGY STACK

The goal of Web services technology is to allow applications to work together over standard Internet protocols, without direct human intervention. By doing so, we can automate many business operations, creating new functional efficiencies and new, more effective ways of doing business.

THE WEB SERVICES TECHNOLOGY STACK



The WS Choreography Definition Language (WS-CDL) is an XML-based language that describes peer-to-peer collaborations of Web Services participants by defining their common and complementary observable behavior.

ADVANTAGES OF SERVICE ORIENTED ARCHITECTURE

Platform independence:

Services communicate with other applications through common language which means it is independent of the platform on which the application is running. Services can provide API in different languages e.g. PHP, Java etc.

Maintenance is easy:

Editing and updating any service is easy. You don't need to update your system. Service is maintained by a 3rd party and any change in that service will not affect your system

ADVANTAGES OF SERVICE ORIENTED ARCHITECTURE

Prevent reinventing the wheel:

The company can use pre-build service and doesn't have to rebuild the same functionality from scratch. This will also increase the productivity of the company.

Scalable:

If any service gets many users then it can be easily scalable by attaching more servers. This will make the service available all time to the users.

Reliable:

Services are usually small in size as compared to full-fledged applications. So it is easier to debug and test the independent services.

ADVANTAGES OF SERVICE ORIENTED ARCHITECTURE

Independent of other services

Services are independent of each other. So services can be used by multiple applications at the same time.

DISADVANTAGES OF SERVICE ORIENTED ARCHITECTURE

Extra overload:

In SOA, all inputs are validated before it is sent to the service. If you are using multiple services then it will overload your system with extra computation.

High cost:

SOA is costly in terms of human resources, development, and technology.

High bandwidth server:

As some web service sends and receives messages and information frequently so it easily reaches a million requests per day. So it involves a high-speed server with a lot of data bandwidth to run a web service.

THE END