UNIT 1

Syllabus:

- ❑ **What Are Web Services?**
- ❑ **Types of Web Services**
- ❑ **Distributed computing infrastructure**
- ❑ **overview of XML**
- ❑ **SOAP**
- ❑ **Building Web Services with JAX-WS**
- ❑ **Registering and Discovering Web Services**
- ❑ **Service Oriented Architecture**
- ❑ **Web Services Development Life Cycle**
- ❑ **Developing and consuming simple Web Services across platform**

1. Write about significance and benefits of SOA. List some benefits of an SOA.

ANS 1:

→ Service-oriented architecture (SOA) is a logical view 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.

→ This architectural approach particularly applicable when multiple applications running on varied technologies and platforms need to communicate with each other.

→ The Essential Goal of an SOA is to enable general purpose interoperability among existing technologies and extensibility to future purposes and architectures.

→ Service-oriented architectures have the following key characteristics:

+ SOA services have self-describing interfaces in platform-independent XML documents. Web Services Description Language (WSDL) is the standard used to describe the services.
+ SOA services communicate with messages formally defined via XML Schema (also called XSD). Communication among consumers and providers or services typically happens in heterogeneous environments, with little or no knowledge about the provider. Messages between services can be viewed as key business documents processed in an enterprise.
+ SOA services are maintained in the enterprise by a registry that acts as a directory listing. Applications can look up the services in the registry and invoke the service. Universal Description, Definition, and Integration (UDDI) is the standard used for service registry.
+ Each SOA service has a quality of service (QoS) associated with it. Some of the key QoS elements are security requirements, such as authentication and authorization, reliable messaging, and policies regarding who can invoke services.

→ Benefits of SOA

While the SOA concept is fundamentally not new, SOA differs from existing distributed technologies in that most vendors accept it and have an application or platform suite that enables SOA.

+ SOA, with a ubiquitous set of standards, brings better reusability of existing assets or investments in the enterprise and lets you create applications that can be built on top of new and existing applications.
+ SOA enables changes to applications while keeping clients or service consumers isolated from evolutionary changes that happen in the service implementation.
+ SOA enables upgrading individual services or services consumers; it is not necessary to completely rewrite an application or keep an existing system that no longer addresses the new business requirements.
+ Finally, SOA provides enterprises better flexibility in building applications and business processes in an agile manner by leveraging existing application infrastructure to compose new services.

2. What are the main layers in an SOA and what is their purpose?

ANS 2:

An SOA is shown to comprise six distinct layers: domains, business processes,

- business services, infrastructure services, service realizations, and operational systems.
- Each of these describes a logical separation of concerns by defining a set of common enterprise elements.
- *Layer 1:*
  + The topmost layer (layer 1) is formed based on the observation that all business process constellations in an enterprise target a particular business domain.
  + A *business domain* is a functional domain comprising a set of current and future business processes that share common capabilities and functionality and can collaborate with each other to accomplish a higher-level business objective, such as loans, insurance, banking, finance, manufacturing, marketing, human resources, etc.
- *Layer 2:*
  + Layer 2 in the SOA model, the business process layer, is formed by subdividing a business domain, such as distribution, into a small number of core business processes, such as purchasing, order management, and inventory, which are made entirely standard for use throughout the enterprise.
  + In an SOA we may think of such a business process as consisting of people, business services, and the interfaces between the business services.
- *Layer 3:*
  + In the layered SOA model, one approach to specifying the right business services (which comprise layer 3) for a process like order management is to decompose it into increasingly smaller subprocesses until the process cannot decomposed any further.

+ The resulting subprocesses then become candidate indivisible (singular) business services for implementation.
+ *Business services* automate generic business tasks that provide value to an enterprise and are part of standard business process.

- *Layer 4:*
  + *Technical services* in layer 4 are coarse-grained services that provide the technical infrastructure enabling the development, delivery, maintenance, and provisioning of singular business services (in layer 3) and their integration into processes (in layer 2) as well as capabilities that maintain QoS such as security, performance, and availability.
  + *Access services* are dedicated to transforming data and integrating legacy applications and functions into the SOA environment.
  + Infrastructure services also provide *management and monitoring services*.
  + Management services manage resources both within and across system boundaries, gather information about managed systems and managed resource status and performance.
  + *Interaction services* support the interaction between applications and end users.
  + All types of infrastructure services are considered as integral parts of the Enterprise Service Bus, which is the technical infrastructure that enables standards-based integration in an SOA environment.
- *Layer 5:*
  + Layer 5 is the component realization layer that is used for implementing services out of pre-existing applications and systems found in the operational systems layer (layer 6).
  + This layer uses components to implement (realize) the required service functionality.

+ Components comprise autonomous units of software that may provide a useful service or a set of functionalities to a client (business service) and have meaning in isolation from other components with which they interoperate.

- Layer 6:
  + The operational systems in layer 6 are used by components to implement business services and processes.
  + Layer 6 is shown to contain existing enterprise systems or applications, including customer relationship management (CRM) and ERP systems and applications, legacy applications, database systems and applications, other packaged applications, and so on.
  + These systems are usually known as enterprise information systems.

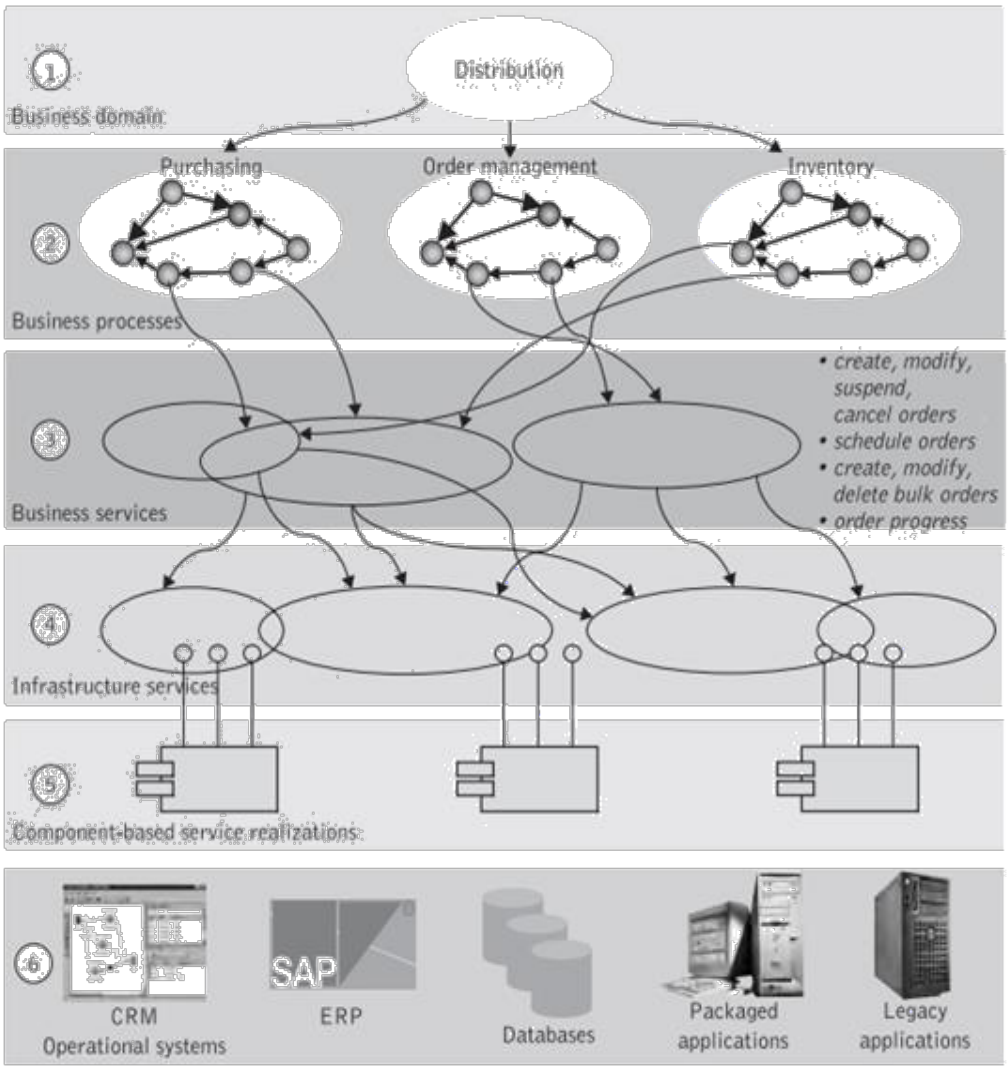


Figure 1.6 Layers in an SOA

3. What is the Web services technology stack?

ANS 3:

→ 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.
→ In order to simply things we provide a classification scheme for the most important standards
in the Web services technology stack, which we introduce briefly below.

- Enabling technology standards : Although not specifically tied to any specific transport protocol, Web services build on ubiquitous Internet connectivity and infrastructure to ensure nearly universal reach and support.
- Extensible Markup Language (XML). XML is a widely accepted format for all exchanging data and its corresponding semantics.
- Core services standards.
- The core Web services standards comprise the baseline standards SOAP, WSDL, and UDDI:
    - ❖ Simple Object Access Protocol: SOAP is a simple XML-based messaging protocol on which Web services rely to exchange information among themselves. It is based on XML and uses common Internet transport protocols like HTTP to carry its data.
    - ❖ Service description: Web services can be used effectively when a Web service and its client rely on standard ways to specify data and operations, to represent Web service contracts, and to understand the capabilities that a Web service provides. Web service are first described by means of a Web Services Description Language.
    - ❖ Service publication: Web service publication is achieved by UDDI, which is a public directory that provides publication of on-line services and facilitates eventual discovery of Web services. Companies can publish WSDL specifications for services they provide and other enterprises can access those services using the description in WSDL.
- Service composition and collaboration standards
    - ❖ Service composition: Describes the execution logic of Web-services-based applications by defining their control flows (such as conditional, sequential, parallel, and exceptional execution) and prescribing the rules for consistently managing their unobservable business data.
    - ❖ Service collaboration: Describes cross-enterprise collaborations of Web service participants by defining their common observable behavior, where synchronized information exchanges occur through their shared contact points, when commonly defined ordering rules are satisfied.
    - ❖ Coordination/transaction standards: Solving the problems associated with service discovery and service description retrieval is the key to success of Web services.
    - ❖ Value-added standards: Value-added services standards include mechanisms for security and authentication, authorization, trust, privacy, secure conversations, contract management, and so on.

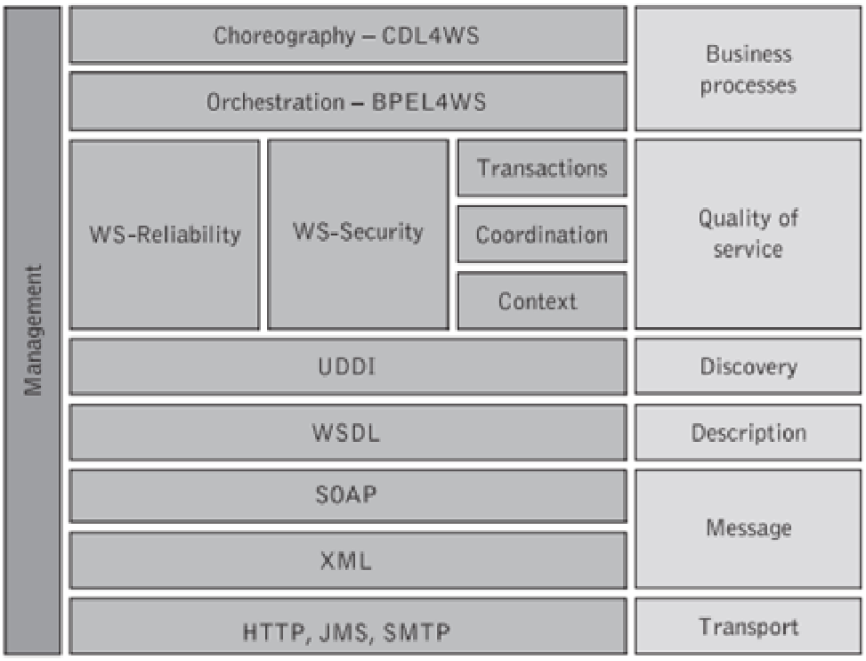


Figure 1.7 The Web services technology stack

4. Define web service. Explain the two types of web service.

ANS 4:

★ A Web service is a self-describing, self-contained software module available via a network,
such as the Internet, which completes tasks, solves problems, or conducts transactions on behalf of a user or application.

★ Web services constitute a distributed computer infrastructure made up of many different
interacting application modules trying to communicate over private or public networks (including the Internet and Web) to virtually form a single logical system.

★ Types of Web Services:

- *Informational, or type I*

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

➔ Informational services are services of relatively simple nature.

➔ They either provide access to content interacting with an end user by means of simple request/response sequences, or alternatively may expose back-end business applications to other applications.

➔ Web services that typically expose the business functionality of the applications and components that underlie them are known as programmatic services.

➔ Three subcategories:

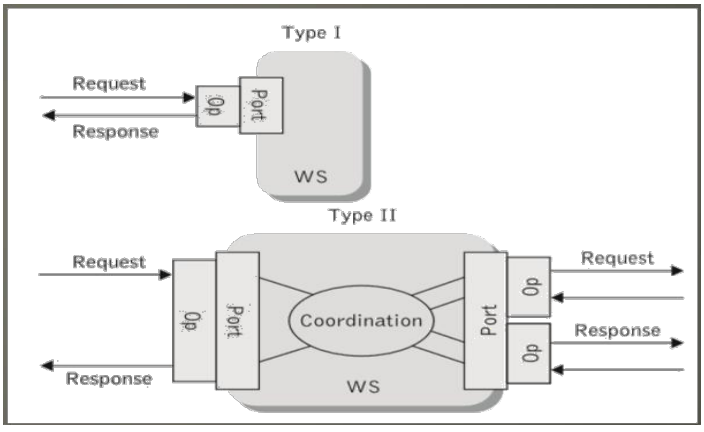
- Pure content services:
  - Which give programmatic access to content such as weather report information, simple financial information, stock quote information, design information, news items and so on.
- Simple trading services:
  - Provide a seamless aggregation of information across disparate
    systems and information sources, including back-end systems, giving programmatic access to a business information system so that the requestor can make informed decisions.

5 | WEB SERVICES

- Such service requests may have complicated realizations.

- Information syndication services:
  - Which are value-added information Web services that purport to “plug into” commerce sites of various types, such as e-marketplaces, or sell-sites.
  - Generally speaking, these services are offered by a third party and run the whole range from commerce-enabling services, such as logistics, payment, fulfilment, and tracking services, to other value-added commerce services, such as rating services.

- *Complex services:*
  - ➔ Complex (or composite) services typically involve the assembly and invocation of many pre-existing services possibly found in diverse enterprises to complete a multi-step business interaction.
  - ➔ 2 subcategories:
    - Complex services that compose interactive Web services:
      - These services expose the functionality of a Web application’s presentation (browser) layer.
      - They frequently expose a multi-step Web application behaviour that combines a Web server, an application server, and underlying database systems and typically deliver the application directly to a browser and eventually to a human user for interaction.
      - Clients of these Web services can incorporate interactive business processes into their Web applications, presenting integrated (aggregated) applications from external service providers.
    - Complex services exhibit coarse-grained functionality and are stateful.
    - A stateful Web service maintains some state between different operation invocations issued by the same or different Web service clients.



5.	**What are the two types of information? Explain.**
ANS 5:	• Informational services are services of relatively simple nature. • They either provide access to content interacting with an end user by means of simple request/response sequences, or alternatively may expose back-end business applications to other applications. • Informational services can be subdivided into three subcategories according to the business problems they solve: • ***Pure content services***, which give programmatic access to content such as weather report information, simple financial information, stock quote information, design information, news items, and so on. • ***Simple trading services***, which are more complicated forms of informational services that can provide a seamless aggregation of information across disparate systems and information sources, including back-end systems, giving programmatic access to a business information system so that the requestor can make informed decisions. Such service requests may have complicated realizations. Consider, for example, "pure" business services, such as logistic services, where automated services are the actual front-ends to complex physical organizational information systems. • ***Information syndication services***, which are value-added information Web services that purport to "plug into" commerce sites of various types, such as e-marketplaces, or sell sites. These services are offered by a third party and run the whole range from commerce enabling services, such as logistics, payment, fulfilment, and tracking services, to other value-added commerce services, such as rating services. Typical examples of syndicated services might include reservation services on a travel site or rate quote services on an insurance site. • Informational services are singular in nature in that they perform a complete unit of work that leaves its underlying datastores in a consistent state. • However, they are not transactional in nature (although their back-end realizations may be).
	• An informational service does not keep any memory of what happens to it between requests. • In that respect this type of service is known as a stateless Web service
6.	**Explain in detail about Synchronicity.**
ANS 6:	We may distinguish between two programming styles for services: synchronous or • remote procedure call (RPC) style versus asynchronous or message (document) style • *Synchronous services:* ➔ Clients of synchronous services express their request as a method call with a set of arguments, which returns a response containing a return value. ➔ This implies that when a client sends a request message, it expects a response message before continuing with its computation. ➔ This makes the whole invocation an all-or-nothing proposition. ➔ If one operation is unable to complete for any reason, all other dependent operations will fail. ➔ Because of this type of bilateral communication between the client and service, RPC-style services require a tightly coupled model of communication between the client and service provider. ➔ RPC style Web services are normally used when an application exhibits the following characteristics:

★ The client invoking the service requires an immediate response.
★ The client and service work in a back-and-forth conversational way.

- *Asynchronous services:*
  ➔ Asynchronous services are document-style or message driven services.
  ➔ When a client invokes a message-style service, the client typically sends it an entire
  document, such as a purchase order, rather than a discrete set of parameters.

➔ The service accepts the entire document, it processes it and may or may not return a result message.

➔ A client that invokes an asynchronous service does not need to wait for a response before it continues with the remainder of its application.

➔ The response from the service, if any, can appear hours or even days later.

➔ Asynchronous interactions (messaging) are a key design pattern in loosely coupled environments.

➔ Messaging enables a loosely coupled environment in which an application does not need to know the intimate details of how to reach and interface with other applications.

➔ Document style Web services are normally used when an application exhibits the following characteristics:

★ The client does not require (or expect) an immediate response.

★ The service is document oriented (the client typically sends an entire document, e.g., a purchase order, rather discrete parameters).

7. ~~Explain the Client Service Model with respect to Web Service.~~

ANS 7:

➔ A client-server architecture is a computational architecture in which processing and storage
tasks are divided between two classes of network members, clients and servers.

➔ Client-server involves client processes (service consumers) requesting service from server
processes (service providers). Servers may in turn be clients of other servers

➔ The same device may function as both client and server.

➔ In a client–server architecture the client machine runs software and applications that are
stored locally. Some of the applications may be stored and executed on the server, but most
of them are on the client.

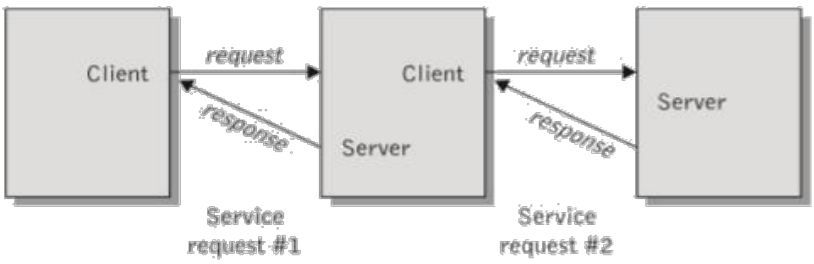
➔ The server also provides the data for the application.

➔ In a client–server architecture, the client actually has two tasks. The client makes requests to
servers and is also responsible for the user interface

➔ The client–server model is the most prevalent structure for Internet applications. The Web,
e-mail, file transfer, Telnet applications, newsgroups, and many other popular Internet based
applications adopt the client–server model.

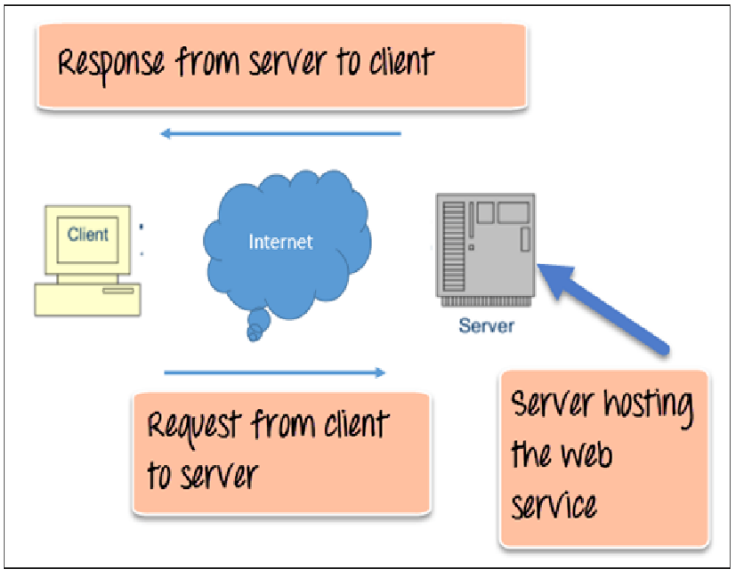
➔ Since a client program typically runs on one computer and the server program runs on another
computer, client–server Internet applications are, by definition, distributed applications.

➔ The client program and the server program interact with each other by sending each other
messages over the Internet.



Client–server architecture

8 | WEB SERVICES



Web Service Architecture Diagram

- The above diagram shows a very simplistic view of how a web service would work.
- The client would invoke a series of web service calls via requests to a server which would host the actual web service.
- These requests are made through what is known as remote procedure calls.
- Remote Procedure Calls (RPC) are calls made to methods which are hosted by the relevant web service.
- As an example, Amazon provides a web service that provides prices for products sold online via amazon.com.
- The front end or presentation layer can be in .NET or Java but either programming language would have the ability to communicate with the web service.
- The main component of a web service is the data which is transferred between the client and the server, and that is XML.
- XML (Extensible mark-up language) is a counterpart to HTML and easy to understand the intermediate language that is understood by many programming languages.
- So, when applications talk to each other, they talk in XML. This provides a common platform for application developed in various programming languages to talk to each other.

8. What is SOAP? Explain in detail OR Discuss why SOAP is called as a lightweight process.

ANS 8: *Simple Object Access Protocol*

- ➔ SOAP codifies the use of XML as an encoding scheme for request response typically using HTTP as a transport protocol to reach any destination in the Internet without needing any additional wrapping or encoding.
- ➔ SOAP is an open standard, enabling web services to be developed and supported across a range of platforms and environments.
- ➔ Since, SOAP possesses only two fundamental properties. It can send and receive HTTP (or other) transport protocol packets, and process XML messages, hence it is lightweight protocol.
- ➔ Moreover, SOAP does not require that a specific object to be tied to a given endpoint.

9 | WEB SERVICES

➔ The SOAP specification describes a standard, XML-based way to encode requests and responses, including:
  - Requests to invoke a method on a service, including *in parameters*
  - Responses from a service method, including return value and *out parameters*
  - Errors from a service

➔ SOAP describes the structure and data types of message payloads by using the emerging W3C XML Schema standard issued by the World Wide Web Consortium (W3C).

➔ SOAP is a transport-agnostic messaging system; SOAP requests and responses *travel* using HTTP, HTTPS, or some other transport mechanism.

➔ In general, a SOAP service remote procedure call (RPC) request/response sequence includes the following steps:
  - A *SOAP client* formulates a request for a service. This involves creating a conforming XML document, either explicitly or using Oracle SOAP client API.
  - A SOAP client sends the XML document to a *SOAP server*. This SOAP request is posted using HTTP or HTTPS to a SOAP Request Handler running as a servlet on a Web server.
  - The Web server receives the SOAP message, an XML document, using the SOAP Request Handler Servlet. The server then dispatches the message as a service invocation to an appropriate server-side application providing the requested service.
  - A response from the service is returned to the SOAP Request Handler Servlet and then to the caller using the standard SOAP XML payload format.

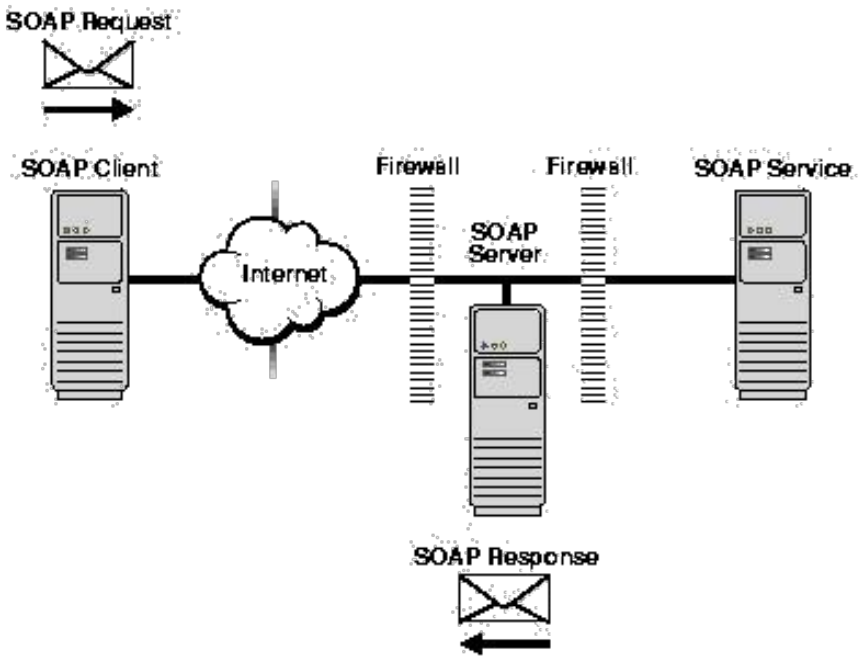
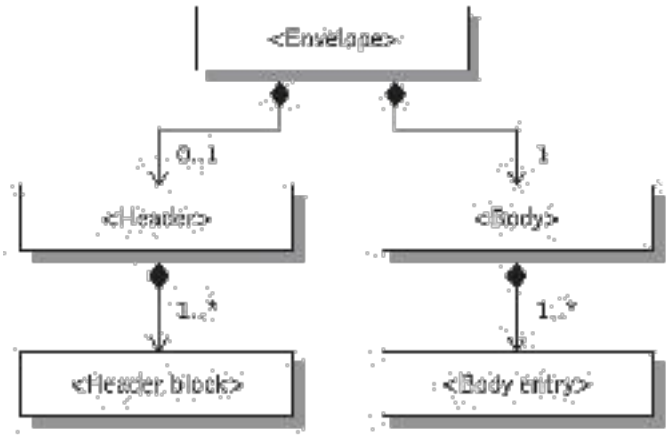


Figure 1-1 Components of the SOAP Architecture

→ The SOAP specification does not describe how the SOAP server should handle the content of the SOAP message body.

→ The content of the body may be handed to a SOAP service, depending on the SOAP server implementation.

9.	**Explain in detail the structure of a SOAP message.**
ANS 9:	 • A SOAP message consists of an <Envelope> element containing an optional <Header> and a mandatory <Body> element • The contents of these elements are application defined and not a part of the SOAP specifications, although the latter do have something to say about how such elements must be handled. • A SOAP <Header> element contains blocks of information relevant to how the message is to be processed. • This provides a way to pass information in SOAP messages that is not part of the application payload. • Such “control” information includes, for example, passing directives or contextual information related to the processing of the message, e.g., routing and delivery settings, authentication or authorization assertions, and transaction contexts. • This allows a SOAP message to be extended in an application-specific manner. • The immediate child elements of the <Header> element are called header blocks, and represent a logical grouping of data, which can individually be targeted at SOAP nodes that might be encountered in the path of a message from a sender to an ultimate receiver. • The SOAP <Body> element is the mandatory element within the SOAP <Envelope>, which implies that this is where the main end-to-end information conveyed in a SOAP message must be carried.
10.	**List the advantages and disadvantages of SOAP.**
ANS 10:	*Advantages-* • **Simplicity:** SOAP is simple as it based on XML, which is highly structured and easy to parse. • **Portability**: SOAP is portable without any dependencies on the underlying platform like byte-ordering issues or machine-word widths. Today, XML parsers exist for virtually any platform from mainframes to write-watch-size devices. • **Firewall-friendly**: Posting data over HTTP means not only that the delivery mechanism is widely available but also that SOAP is able to get past firewalls that pose problems for other methods. • **Use of open standards:** SOAP uses the open standard of XML to format the data, which makes it easily extendable and well supported. • **Interoperability**: SOAP is built on open, rather than vendor-specific, technologies and facilitates true distributed interoperability and loosely coupled applications. Because SOAP is a wire protocol based on XML and HTTP, it is possibly the most widely interoperable protocol to date and can be used to describe message exchanges.

- **Universal acceptance**: SOAP is the most widely accepted standard in the message communication domain.
- **Resilience to changes**: Changes to the SOAP infrastructure will likely not affect applications using the protocol, unless significant serialization changes are made to the SOAP specification.

Disadvantages-

- SOAP is stateless, it does not keep track of orienting information to link multiple communications together.
- Because of the verbose XML format, SOAP can be considerably slower than competing middleware technologies.
- SOAP serializes by value and does not support serialization by reference. Serialization by value requires that multiple copies of an object will, over time, contain state information that is not synchronized with other dislocated copies of the same object. This means that currently it is not possible for SOAP to refer or point to some external data source (in the form of an object reference).

11. Explain in brief the following styles of SOAP services

a. RPC

b. Document

OR

Describe the two SOAP communication models

ANS 11: *RPC - Style*

- An RPC-style Web service appears as a remote object to a client application.
- The interaction between a client and an RPC-style Web service centres around a service-specific interface.
- Clients express their request as a method call with a set of arguments, which returns a response containing a return value.
- RPC style supports automatic serialization/deserialization of messages, permitting developers to express a request as a method call with a set of parameters, which returns a response containing a return value.
- Because of this type of bilateral communication between the client and Web service, RPC-style Web services require a tightly coupled (synchronous) model of communication between the client and service provider.
- Example:
  - *WSDL file:*
    - In WSDL file, it doesn't specify the types details.

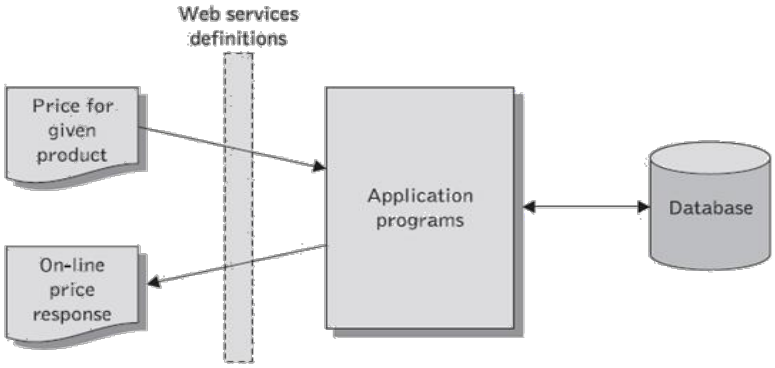
```
<types/>

<message name="getHelloWorldAsString">
    <part name="arg0" type="xsd:string"/>
</message>
<message name="getHelloWorldAsStringResponse">
    <part name="return" type="xsd:string"/>
</message>
```

  - For soap:body, it defines use and namespace attributes.

```
<binding name="HelloWorldImplPortBinding"
type="tns:HelloWorld">
<soap:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="rpc"/>
    <operation name="getHelloWorldAsString">
        <soap:operation soapAction=""/>
        <input>
```

```
                    <soap:body
7.                  use="literal"
        namespace="http://javatpoint.com/"
        />
8.              </input>
9.              <output>
                    <soap:body
10.                 use="literal"
    namespace="http://javatpoint.com/"/>
11.             </output>
12.     </operation>
13. </binding>
```



4.8 RPC-style Web service for calculating the price of a given product

Document-style SOAP

- Typically, sending non-coded XML content in the body is known as document-style SOAP, since it focuses on the message as an XML document rather than an abstract data model that happens to be encoded into XML.
- With document-style SOAP applications, there are no restrictions on the contents of the SOAP Element.
- Document-style Web services are message driven. When a client invokes a message-style Web service, the client typically sends it an entire document, such as a purchase order, rather than a discrete set of parameters.
- Example:
  - In WSDL file, it specifies types details having namespace and schemaLocation.

```
1. <types>
2.          <xsd:schema>
3.          <xsd:import namespace="http://javatpoint.com/"
            schemaLocation="http://localhost:7779/ws/hello?xsd=1
            "/>
4.          </xsd:schema>
5. </types>
```

- For message part, it defines name and element attributes.

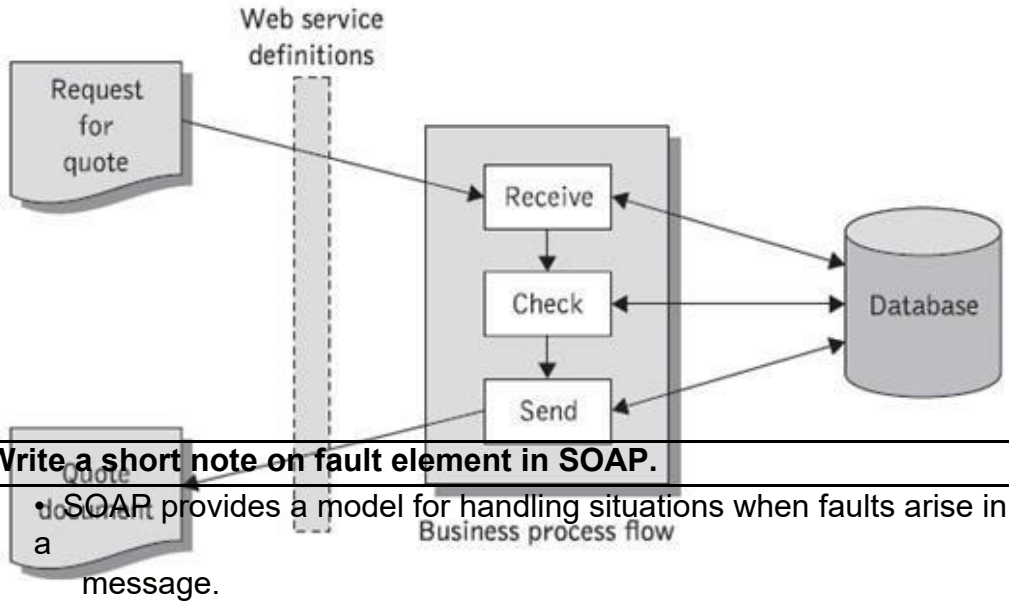
```
1. <message name="getHelloWorldAsString">
2.          <part name="parameters" element="tns:getHelloWorldAsString"/>
3. </message>
4. <message name="getHelloWorldAsStringResponse">
5.          <part name="parameters" element="tns:getHelloWorldAsStringResponse"/>
6. </message>
```

- For soap:body, it defines use attribute only not namespace.

```
1.  <binding name="HelloWorldImplPortBinding" type="tns:HelloWorld">
2.          <soap:binding
            transport="http://schemas.xmlsoap.org/soap/http"
            style="document"/>
3.          <operation name="getHelloWorldAsString">
```

4. <soap:operation soapAction=""/>

```
5.              <input>
6.                  <soap:body use="literal"/>
7.              </input>
8.              <output>
9.                  <soap:body use="literal"/>
10.             </output>
11.       </operation>
12. </binding>
```



12. Write a short note on fault element in SOAP.

ANS 12:

- SOAP provides a model for handling situations when faults arise in the processing of a message.
- SOAP distinguishes between the conditions that result in a fault and the ability to signal that fault to the originator of the faulty message or another node.
- The SOAP fault model requires that all SOAP-specific and application-specific faults be reported using a special-purpose element called env:Fault.
- The env:Fault element is a reserved element predefined by the SOAP specification whose purpose is to provide an extensible mechanism for transporting structured and unstructured information about problems that have arisen during the processing of a SOAP message.
- A SOAP message can carry only one fault block.
- It is optional part of a SOAP message.
- The SOAP Fault element has the following sub elements:

Sub Element	Description
<faultcode >	A code for identifying the fault
<faultstring>	A human readable explanation of the fault

<faultactor>	Information about who caused the fault to happen
<detail>	Holds application specific error information related to the Body element

```
<env:Body>
   <env:Fault>
      <env:Code>
         <env:Value>env:Sender</env:Value>
         <env:Subcode>
            <env:Value> m:InvalidPurchaseOrder </env:Value>
         </env:Subcode>
      </env:Code>
      <env:Reason>
         <env:Text xml:lang="en-UK"> Specified product
          did not exist </env:Text>
      </env:Reason>
      <env:Detail>
        <err:myFaultDetails
           xmlns:err="http://www.plastics_supply.com/
           faults">
          <err:message> Product number contains invalid
           characters
          </err:message>
          <err:errorcode> 129 </err:errorcode>
        </err:myFaultDetails>
      </env:Detail>
   </env:Fault>
</env:Body>
```

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode xsi:type = "xsd:string">SOAP-ENV:Client</faultcode>
         <faultstring xsi:type = "xsd:string">
            Failed to locate method (ValidateCreditCard) in class (examplesCreditCar
               /usr/local/ActivePerl-5.6/lib/site_perl/5.6.0/SOAP/Lite.pm line 1555.
         </faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

13.	**Write a short note on payload in the SOAP message.**
ANS 13:	● A message typically comprises three basic elements: a header, its properties, and a message payload or body. ● The message body carries the actual “payload” of the message. ● The format of the message payload can vary across messaging implementations. ● Most common formats are plain text, a raw stream of bytes for holding any type of binary data, or a special XML message type that allows the message payload to be accessed using any number of common XML parsing technologies. ● The payload is the actual message content that you want to send to the other application.

- For example, here's a typical "hello world" SOAP message (using document/literal encoding):

```
<e:Envelope
    xmlns:e="http://schemas.xmlsoap.org/soap/envelope/">
    <e:Body
        xmlns:t="http://my.example.com/xsd/types/"
        xmlns:i="https://www.w3.org/2001/XMLSchema-
        instance"
        xmlns:d="https://www.w3.org/2001/XMLSchema"> <t:In
        i:type="d:string">
            Hello world!
        </t:In>
    </e:Body>
</e:Envelope>
```

- Another example for payload:

```
<SOAP-
ENV:Envelope
  xmlns:xsi="http://www.w3.org/1999/XMLSchema/instance"
  xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope"
  xsi:schemaLocation=
      "http://www.northwindtraders.com/schemas/NPOSchema.xs
  d"> <SOAP-ENV:Body xsi:type="NorthwindBody">
    <UpdatePO>
      <orderID>0</orderID>
      <customerNumber>999</customerNumber>
      <item>89</item>
      <quantity>3000</quantity>
      <return>0</return>
    </UpdatePO>
  </SOAP-
  ENV:Body>
</SOAP-ENV:Envelope>
```

The above example is for updating a customer information.

14. Describe with a diagram an RPC call using SOAP over HTTP.

ANS 14:

- SOAP codifies the use of XML as an encoding scheme for request and response parameters typically using HTTP as a transport protocol to reach any destination in the Internet without needing any additional wrapping or encoding.
- A SOAP method is simply an HTTP request and response that complies with the SOAP encoding rules, while a SOAP endpoint is simply an HTTP-based URL that identifies a target for method invocation.
- SOAP does not require that a specific object be tied to a given endpoint.
- Rather, it is up to the implementer to decide how to map the object endpoint identifier onto a provider-side object.
- In the following we examine briefly the concept of HTTP binding with the SOAP request/response message exchange pattern using the HTTP POST method.
- Note that the use of this message exchange pattern in the SOAP HTTP binding is available to all applications, whether they involve the exchange of general XML data or RPCs

encapsulated in SOAP messages.

- SOAP requests are transported in the body of an HTTP POST method, which transmits the request content in the body of the HTTP request message.
- With POST, the SOAP envelope becomes the data part of an HTTP request message.
- The SOAP response is returned in the HTTP response (see Figure 4.10).
- The code snippet in Listing 4.9 illustrates the use of the purchase order SOAP RPC request message depicted in Listing 4.5 within an HTTP POST operation, which means the message is being posted to some service provider.

- When RPC-style SOAP is used to format the SOAP message, for instance the message in Listing 4.9, a procedure is being invoked and results are generated which are returned in a SOAP response message.
- The SOAP response (see Listing 4.10) is carried in the data part of the HTTP response.

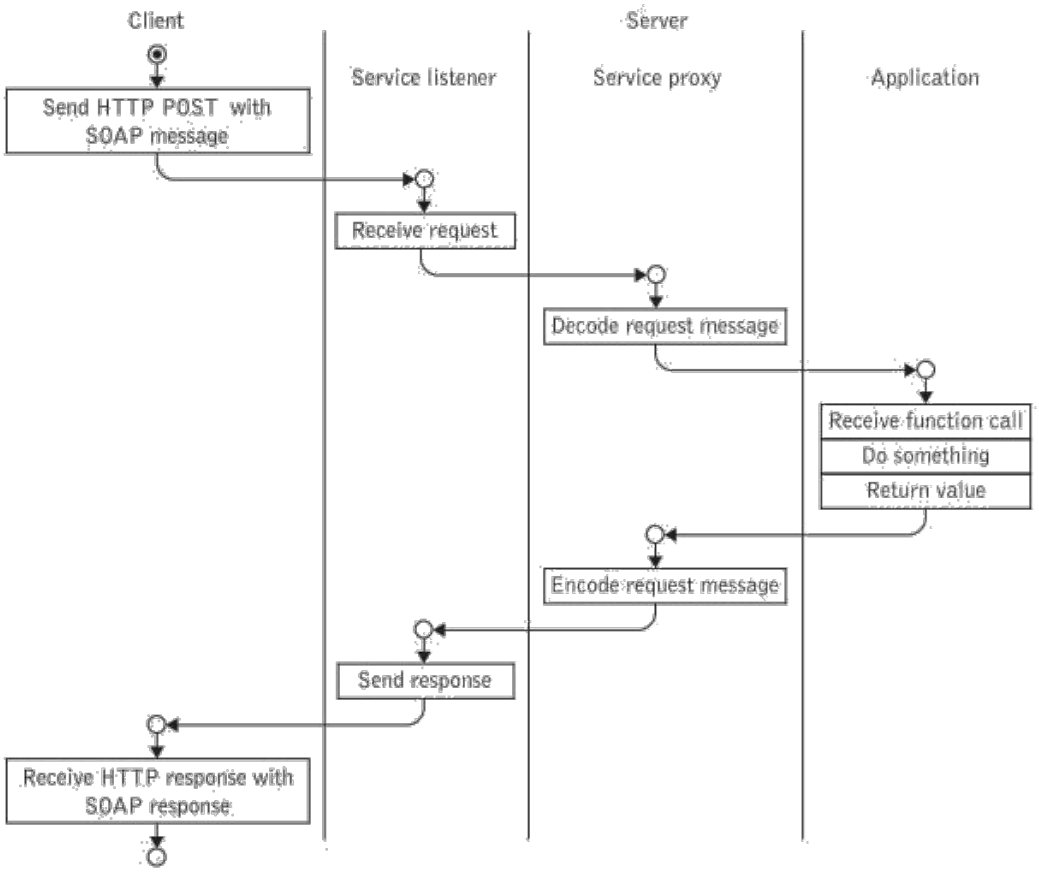


Figure 4.10 Usage of the SOAP HTTP POST method

-

```
POST /Purchase Order HTTP/1.1
Host: http://www.plastics_supply.com  <! - Service provider -- >
Content-Type:application/soap+xml;
charset = "utf-8"
Content-Length: nnnn

<?xml version="1.0" ?>
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
       <tx:Transaction-id
           xmlns:t="http://www.transaction.com/transactions"
           env:mustUnderstand='1'>
               512
       </tx:Transaction-id>
    </env:Header>
    <env:Body>
       <m:GetProductPrice>
          <product-id>  450R6OP  </product-id >
       </m:GetProductPrice >
    </env:Body>
</env:Envelope>
```

Listing 4.9 Sample HTTP/SOAP enveloped request

```
HTTP/1.1 200 OK
Content-Type:application/soap+xml;
charset = "utf-8"
Content-Length: nnnn

<?xml version="1.0" ?>
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
       <--! - Optional context information -->
    </env:Header>
    <env:Body>
       <m:GetProductPriceResponse>
          <product-price> 134.32 </product-price>
       </m:GetProductPriceResponse>
    </env:Body>
</env:Envelope>
```

Listing 4.10 RPC return sent by the purchase order service application

- The RPC return sent by the purchase order service application in the corresponding HTTP response to the request from Listing 4.6. SOAP, using HTTP transport, follows the semantics of the HTTP status codes for communicating status information in HTTP.
- For example, the 200 series of HTTP status codes indicate that the client's request was successfully received, understood, and accepted.
- While it is important to understand the SOAP foundation for services, most Web service developers will not have to deal with this infrastructure directly.
- Most Web services use optimized SOAP bindings generated from WSDL. In this way, SOAP implementations can self-configure exchanges between Web services while masking most of the technical details.

15.	Why is a service description needed? Explain in detail about WSDL.

ANS 15:

• To develop service-based applications and business processes, which comprise service assemblies, Web services need to be described in a consistent manner.

- In this way Web services can be published by service providers, discovered by service clients and developers, and assembled in a manageable hierarchy of composite services that are orchestrated to deliver value-added service solutions and composite application assemblies.
- However, to accomplish this, consumers must determine the precise XML interface of a Web service along with other miscellaneous message details a priori.
- In the Web services world, XML Schema can partially fill this need as it allows developers to describe the structure of XML messages understood by Web services.
- However, XML Schema alone cannot describe important additional details involved in communicating with a Web service such as service functional and non-functional characteristics or service policies.
- Service description is a key to making the SOA loosely coupled and reducing the amount of required common understanding, custom programming, and integration between the service provider and the service requestor's applications.
- Service description is a machine-understandable specification describing the structure, operational characteristics, and non-functional properties of a Web service.
- It also specifies the wire format and transport protocol that the Web service uses to expose this functionality.
- It can also describe the payload data using a type system.
- The service description combined with the underlying SOAP infrastructure sufficiently isolates all technical details, e.g., machine- and implementation-language-specific elements, from the service requestor's application and the service provider's Web service.
- It does not mandate any specific implementation on the service requestor side provided that the contract specified in a standard service description language is abided by.
- Web Services Description Language (WSDL) is a format for describing a Web Services interface.
- It is a way to describe services and how they should be bound to specific network addresses. WSDL has three parts:
    + Definitions
    + Operations
    + Service bindings

Definitions

- Definitions are generally expressed in XML and include both data type definitions and message definitions that use the data type definitions.
- These definitions are usually based upon some agreed upon XML vocabulary.
- This agreement could be within an organization or between organizations.
- Vocabularies within an organization could be designed specifically for that organization.
- They may or may not be based on some industry-wide vocabulary.
- If data type and message definitions need to be used between organizations, then most likely an industry-wide vocabulary will be used.
- XML, however, is not necessary required for definitions.
- The OMG Interface Definition Language (IDL), for example, could be used instead of XML.
- If a different definitional format were used, senders and receivers would need to agree on the format as well as the vocabulary.
- Nevertheless, over time, XML-based vocabularies and messages are likely to dominate.
- XML Namespaces are used to ensure uniqueness of the XML element names in the definitions,
  operations, and service bindings.

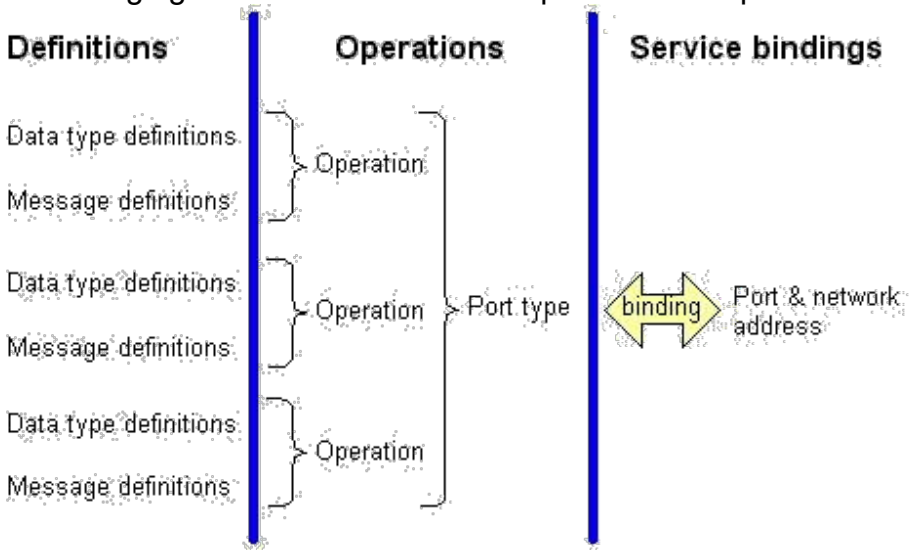
Operations

- Operations describe actions for the messages supported by a Web service.

- There are four types of operations:
    - + One-way: Messages sent without a reply required
    - + Request/response: The sender sends a message and the received sends a reply.
    - + Solicit response: A request for a response. (The specific definition for this action is pending.)
    - + Notification: Messages sent to multiple receivers. (The specific definition for this action is pending.)
- Operations are grouped into port types.
- Port types define a set of operations supported by the Web service.

- Service bindings connect port types to a port.
- A port is defined by associating a network address with a port type.
- A collection of ports defines a service.
- This binding is commonly created using SOAP.

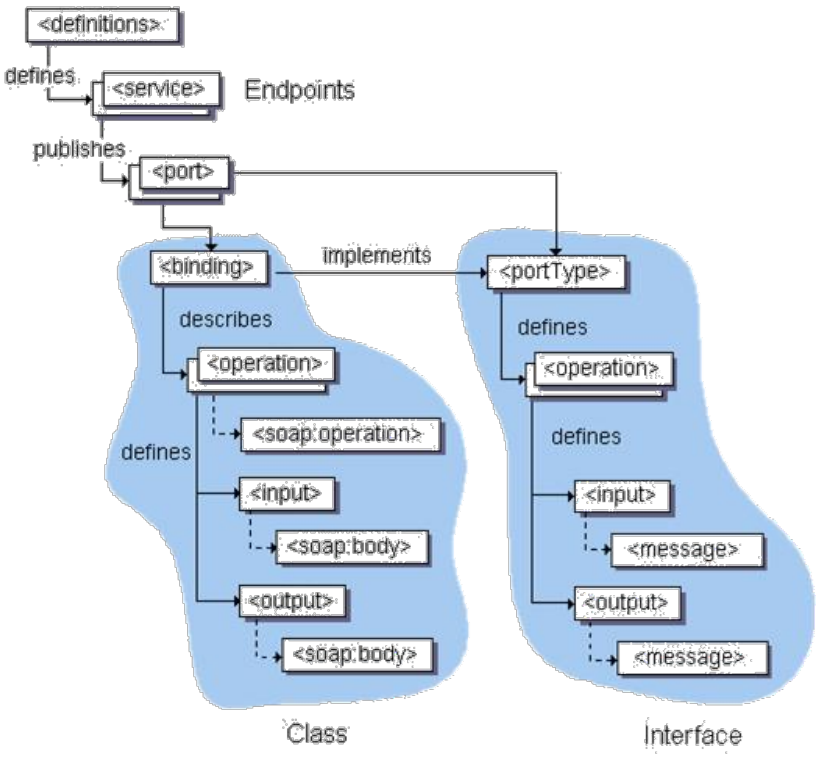
- The following figure shows the relationship of the basic parts of WSDL:



16.	Describe the structure of WSDL document

ANS 16:

- Web Services Description Language (WSDL) is an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages.
- The diagram below illustrates the elements that are present in a WSDL document and indicates their relationships.



➔ **WSDL Document Elements**

- A WSDL document has a definitions element that contains the other five elements, types, message, portType, binding and service.
- WSDL supports the XML Schemas specification (XSD) as its type system.

➔ **Definitions**

- Contains the definition of one or more services.
- JDeveloper generates the following attribute declarations for this section:
  - ➢ name is optional.
  - ➢ targetNamespace is the logical namespace for information about this service. WSDL documents can import other WSDL documents and setting targetNamespace to a unique value ensures that the namespaces do not clash.
  - ➢ xmlns is the default namespace of the WSDL document, and it is set to http://schemas.xmlsoap.org/wsdl/.
  - ➢ All the WSDL elements, such as <definitions>, <types> and <message> reside in this namespace.
  - ➢ xmlns:xsd and xmlns:soap are standard namespace definitions that are used for specifying SOAP-specific information as well as data types.
  - ➢ xmlns:tns stands for this namespace.
  - ➢ xmlns:ns1 is set to the value of the schema targetNamespace, in the <types> section.
- Notice that the default of http://tempuri.org in namespaces to ensure that the namespaces are unique.
  - ★ types

- Provides information about any complex data types used in the WSDL document.
- When simple types are used the document does not need to have a types section.

★ message
- An abstract definition of the data being communicated. In the example, the message contains just one part, response, which is of type string, where string is defined by the XML Schema.

★ operation
- An abstract description of the action supported by the service.

★ portType
- An abstract set of operations supported by one or more endpoints.

★ binding
- Describes how the operation is invoked by specifying concrete protocol and data format specifications for the operations and messages.

★ port
- Specifies a single endpoint as an address for the binding, thus defining a single communication endpoint.

★ service
- Specifies the port address(es) of the binding. The service is a collection of network endpoints or ports.

17. Explain in brief the various tags in WSDL document.

ANS 17:

- A WSDL document is used to describe a web service.
- This description is required, so that client applications are able to understand what the web service actually does.
  - The WSDL file contains the location of the web service and
  - The methods which are exposed by the web service.
- The WSDL file itself can look very complex to any user, but it contains all the necessary information that any client application would require to use the relevant web service.
- Services are defined using six major elements:
  - **types**, which provides data type definitions used to describe the messages exchanged.
  - **message**, which represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.
  - **portType**, which is a set of abstract operations. Each operation refers to an input message and output messages.
  - **binding**, which specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.
  - **port**, which specifies an address for a binding, thus defining a single communication endpoint.
  - **service**, which is used to aggregate a set of related ports.

Types

1. :

- The types element encloses data type definitions that are relevant for the exchanged messages.
- For maximum interoperability and platform neutrality, WSDL prefers the use of XSD as

the canonical type system and treats it as the intrinsic type system.

```
<definitions .... >
   <types>
      <xsd:schema .... />*
```

</types>
</definitions>

2. Messages:

- Messages consist of one or more logical parts.
- Each part is associated with a type from some type system using a message-typing attribute.
- The set of message-typing attributes is extensible.
- WSDL defines several such message-typing attributes for use with XSD:
  - element. Refers to an XSD element using a QName.
  - type. Refers to an XSD simpleType or complexType using a QName.
- Other message-typing attributes may be defined as long as they use a namespace different from that of WSDL.
- Binding extensibility elements may also use message-typing attributes. The syntax for defining a message is as follows.
- The message-typing attributes (which may vary depending on the type system used) are shown in bold.

<definitions .... >
*<message name="nmtoken"> **
*<part name="nmtoken" **element="qname"?***
***type="qname"?**/> * </message>*
</definitions>

- The message name attribute provides a unique name among all messages defined within the enclosing WSDL document.
- The part name attribute provides a unique name among all the parts of the enclosing message.

3. Port type

- A port type is a named set of abstract operations and the abstract messages involved.

<wsdl:definitions .... >
<wsdl:portType
name="nmtoken">
<wsdl:operation name="nmtoken" ....
*/> * </wsdl:portType>*
</wsdl:definitions>

- The port type name attribute provides a unique name among all port types defined within in the enclosing WSDL document.
- An operation is named via the name attribute.
- WSDL has four transmission primitives that an endpoint can support:
  - One-way. The endpoint receives a message.
  - Request-response. The endpoint receives a message and sends a correlated message.
  - Solicit-response. The endpoint sends a message and receives a correlated message.
  - Notification. The endpoint sends a message. WSDL refers to these primitives as operations.

4. Binding

- A binding defines message format and protocol details for operations and messages defined by a portType.
- There may be any number of bindings for a given portType.
- The grammar for a binding is as follows: *<wsdl:definitions .... >*

<wsdl:binding name="nmtoken"
*type="qname"> * <-- extensibility element*
*(1) --> **

<wsdl:operation name="nmtoken"> *
<-- extensibility element (2) --> *
<wsdl:input name="nmtoken"? >?
<-- extensibility element (3) -->
</wsdl:input>
<wsdl:output name="nmtoken"? >?
<-- extensibility element (4) --> *
</wsdl:output>
<wsdl:fault name="nmtoken"> *
<-- extensibility element (5) --> *
</wsdl:fault>
</wsdl:operation>
</wsdl:binding>
</wsdl:definitions>

- The name attribute provides a unique name among all bindings defined within in the enclosing WSDL document.
- A binding references the portType that it binds using the type attribute.

5. Port
   - A port defines an individual endpoint by specifying a single address for a binding. <wsdl:definitions .... >
     <wsdl:service .... > *
     <wsdl:port name="nmtoken"
     binding="qname"> * <-- extensibility element
     (1) -->
     </wsdl:port>
     </wsdl:service>
     </wsdl:definitions>
   - The name attribute provides a unique name among all ports defined within in the enclosing WSDL document.
   - The binding attribute (of type QName) refers to the binding using the linking rules defined by WSDL.
   - Binding extensibility elements (1) are used to specify the address information for the port.
     - A port MUST NOT specify more than one address.
     - A port MUST NOT specify any binding information other than address information.
6. Service
   - A service groups a set of related ports together: <wsdl:definitions .... >
     <wsdl:service
     name="nmtoken"> *
     <wsdl:port .... />*
     </wsdl:service>
     </wsdl:definitions>
   - The name attribute provides a unique name among all services defined within in the enclosing WSDL document.

18. **What is Web Service Interface Definition? Explain in brief.**

ANS 18:

- Service clients interact with a Web service by means of invoking its operations.
- Clients must know not only the interfaces of a Web service and the operations it contains, but also what communication protocol to use for sending messages to the service, along with the specific mechanics involved in using the given protocol, such as the use of commands, headers, and error codes.

- The Web service interface definition describes messages, operations, and port types in a platform- and language-independent manner, parts of messages and bindings.
- 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 runtime implementation.
- The WSDL document specifies a web service interface definition, which includes the elements shown in the following table:
  - Port type component contains Operation signature, –
  Messages component contains Parameter definitions,
  - Types component contains Complex type definitions and
  - Bindings component contains Transport protocol and preload format.

19. Write a short note on WSDL message exchange patterns.

ANS 19: *WSDL message exchange patterns:*

- WSDL is the service representation language used to describe the details of the complete interfaces exposed by Web services and thus is the means to accessing a Web service.
- WSDL interfaces support four types of operations.
- The WSDL operations correspond to the incoming and outgoing versions of two basic operation types:
  - An incoming single message passing operation and its outgoing counterpart ("one-way" and "notification" operations), and
  - The incoming and outgoing versions of a synchronous two-way message exchange
  ("request/response" and "solicit/response").

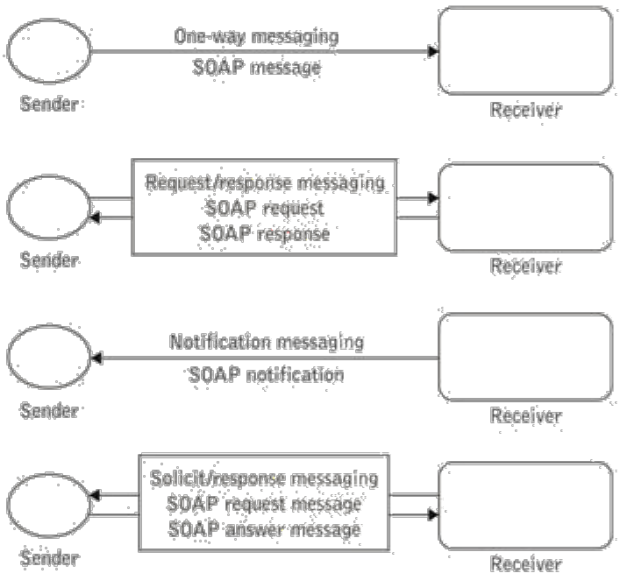


Figure 5.9 WSDL messaging patterns

- One-way operation.
  - A one-way operation is an operation in which the service endpoint receives a message but does not send a response.
  - An example of a one-way operation might be an operation representing the submission of an order to a purchasing system. Once the order is sent, no immediate response is expected.
  - This message exchange pattern is typically thought of as asynchronous messaging.

- If an <operation> element is declared with a single <input> element but no <output> element, it defines a one-way operation.
- By listing only an <input> message, the <operation> element indicates that clients will send messages to the Web service without expecting any responses.
- Eg. SubmitPurchaseOrder <portType> that defines a one-way operation.

```
<!-- portType element describes the abstract interface of a Web
 service -->
<wsdl:portType name="SubmitPurchaseOrder_PortType">
     <wsdl:operation name="SubmitPurchaseOrder">
          <wsdl:input name="order"
           message="tns:SubmitPurchaseOrder_Message"/>
     </wsdl:operation>
</wsdl:portType>
```

- Request/response operation.
  - A request/response operation is an operation in which the service endpoint receives a message and returns a message in response.
  - If an <operation> element is declared with a single <input> element followed by a single <output> element, it defines a request/response operation.
  - By listing the <input> tag first, the <operation> indicates that the Web service receives a message that is sent by the client.
  - Listing the <output> tag second indicates that the Web service should respond to the message.
  - Eg. SendPurchase operation as an example of the request/response messaging pattern.
  - The SendPurchase operation receives as input a message containing a purchase order (order number and date) and customer details and responds with a message containing an invoice.

```
<!-- portType element describes the abstract interface of a Web
 service -->
<wsdl:portType name="PurchaseOrder_PortType">
     <wsdl:operation name="SendPurchase">
          <wsdl:input message="tns:POMessage"/>
          <wsdl:output message="tns:InvMessage"/>
     </wsdl:operation>
</wsdl:portType>
```

- Notification operation.
  - A notification operation is an operation in which the service endpoint sends a message to a client, but it does not expect to receive a response.
  - This type of messaging is used by services that need to notify clients of events.
  - A Web service that uses the notification messaging pattern follows the push model of distributed computing.
  - The assumption is that the client (subscriber) has registered with the Web service to receive messages (notifications) about an event.
  - Notification is when a <portType> element contains an <output> tag but no <input> message definitions.

− An example of this could be a service model in which events are reported to the service and where the endpoint periodically reports its status.
− No response is required in this case, as most likely the status data is assembled and logged and not acted upon immediately.

• Solicit/response operation.
− A solicit/response operation is an operation in which the service endpoint sends a message and expects to receive an answering message in response.
− This is the opposite of the request/response operation since the service endpoint is initiating the operation (soliciting the client), rather than responding to a request.
− Solicit/response is similar to notification messaging, except that the client is expected to respond to the Web service.
− As with notification messaging, clients of the solicit/response Web services must subscribe to the service in order to receive messages.
− With this type of messaging the <portType> element first declares an <output> tag and then an <input> message definition – exactly the reverse of a request/response operation.
− An example of this operation might be a service that sends out order status to a client and receives back a receipt.

20. List and explain different types of service discoveries.

ANS 20:

• Service discovery is an important element of an SOA.
• Service discovery is the process of locating Web service providers, and retrieving Web service descriptions that have been previously published.
• Web service discovery entails locating and interrogating Web service definitions, which is a preliminary step for accessing a Web service.
• It is through this discovery process that Web services clients learn that a particular Web service exists, what its capabilities are, and how to properly interact with it.
• Interrogating services involves querying the service registry for Web services matching the needs of a service requestor.
• A query consists of search criteria such as the type of the desired service, preferred price, and maximum number of returned results and is executed against service information published by the service provider.
• After the discovery process is complete, the service developer or client application should know the exact location of a Web service (a URI for the selected service), its capabilities, and how to interface with it.
• Service selection involves deciding on what Web service to invoke from the set of Web services the discovery process returned.
• There are two basic types of service discovery:
  o static and
  o dynamic
• Static service discovery generally occurs at design time, whereas dynamic discovery occurs at run-time.
• **With static discovery**, the service implementation details such as the network location and network protocol to use are bound at design time and service retrieval is performed on a

service registry.

- A human designer usually examines the results of the retrieval operation and the service description returned by the retrieval operation is incorporated into the application logic.
- **With dynamic discovery**, the service implementation details such as the network location and network protocol to use are left unbound at design time so that they can be determined at run-time.
- In this case, the Web service requestor must specify preferences to enable the application to infer which Web service(s) the requestor is most likely to want to invoke.

- The application issues a retrieval operation at run-time against the service registry to locate one or more service implementation definitions that match the service interface definition used by the application.
- Based on application logic, QoS considerations such as best price, performance, security certificates, and so on, the application chooses the most appropriate service, binds to it, and invokes it.

21. Explain UDDI usage model.

ANS 21:

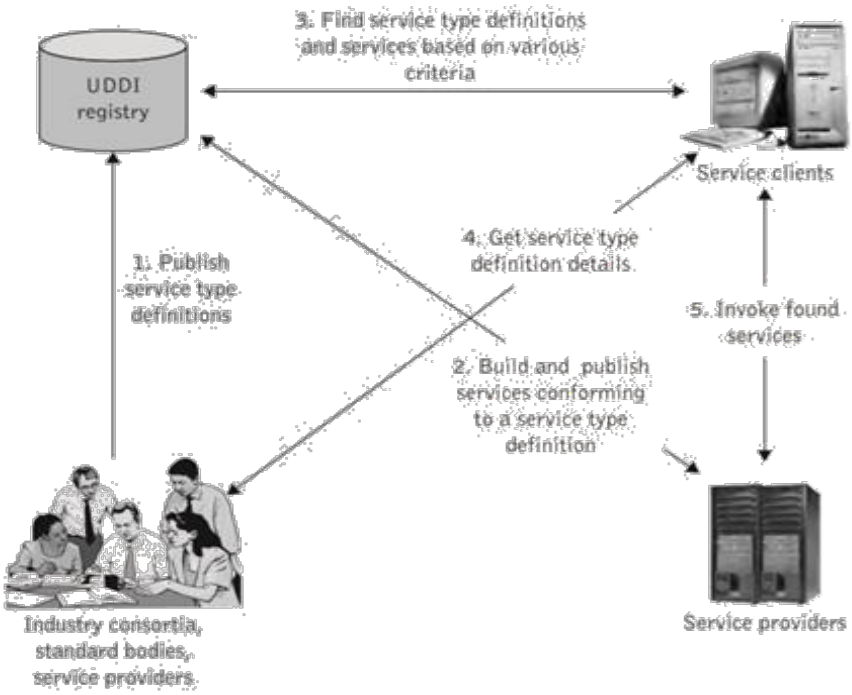


Figure 6.1 The UDDI usage model

- UDDI comes basically in two flavours: public and private UDDI registries.
- In fact, the UDDI usage model envisages different business information provider roles such as:

→ **Registry operators:**

- These refer to the organizations (referred to earlier as operator nodes) that host and operate the UBR.
- The operator nodes **UDDI** manage **BUSINESS** and maintain **REGISTRY** the directory information and cater for replication of business information and other directory-related functions.
- These operators provide a Web interface to the UDDI registry for browsing, publishing, and un-publishing business information.
- An enterprise does not need to register with each of these operators separately; it can register at any one of the operator companies.
- The registry works on a “register once, published everywhere” principle.
- This means that a client searching for a business or service can do so at any of the registry operators – they should get the same information.
- This happens since the operator nodes registries replicate their data with each other on a regular basis.

→ **Standards bodies and industry consortia:**

- These publish descriptions in the form of service type definitions (<tModel>s).
- These <tModel>s do not contain the actual service definitions, instead they have a URL that points to the location where the service descriptions are stored (definitions can be in any form; however, UDDI, recommends using WSDL).

→ **Service providers:**

- Commonly implement Web services conforming to service type definitions supported by UDDI.
- They publish information about their business and services in the UDDI.
- The published data also contains the endpoint of the Web services offered by these enterprises.

→ The structure of the UDDI allows the possibility of a variety of private UDDI nodes. → Currently, we may distinguish the following UDDI deployment possibilities:

➢ **The e-marketplace UDDI:**
- o e-marketplaces are local communities of service providers and requesters organized in vertical markets and gathering around portals.
- o An e-marketplace, a standards body, or a consortium of organizations that participate and compete in the industry can host this private UDDI node.
- o The entries in this private UDDI relate to a particular industry or narrow range of related industries.
- o This gives rise to the idea of a Web services discovery agency (or service broker) that is the organization (acting as a third trusted party) whose primary activities focus on hosting the registry, publishing, and promoting Web services.

➢ **The business partner UDDI registry:**
- o This variant of the above scheme is a private UDDI node hosted behind one of the business partner's firewalls and only trusted or vetted partners can access the registry.
- o It also contains Web service description metadata published by trusted business parties (i.e., those organizations with which the hosting organization has formal agreements/relationships)

➢ **The portal UDDI:**
- o This type of deployment is on an enterprise's firewall and is a private UDDI node that contains only metadata related to the enterprise's Web services.
- o External users of the portal would be allowed to invoke find operations on the registry; however, a publish operation would be restricted to services internal to the portal.
- o The portal UDDI gives a company ultimate control over how the metadata describing its Web services is used.
- o For example, an enterprise can restrict access.
- o It can also monitor and manage the number of lookups being made against its data and potentially get information about who the interested parties are.

➢ **The internal UDDI:**
- o This allows applications in different departments of the organization to publish and find services and is useful for large organizations.
- o The major distinction of this UDDI variant is the potential for a common administrative domain that can dictate standards (e.g., a fixed set of tModels can be used).

22.	**Explain in detail the data structures used in UDDI.**
ANS 22:	

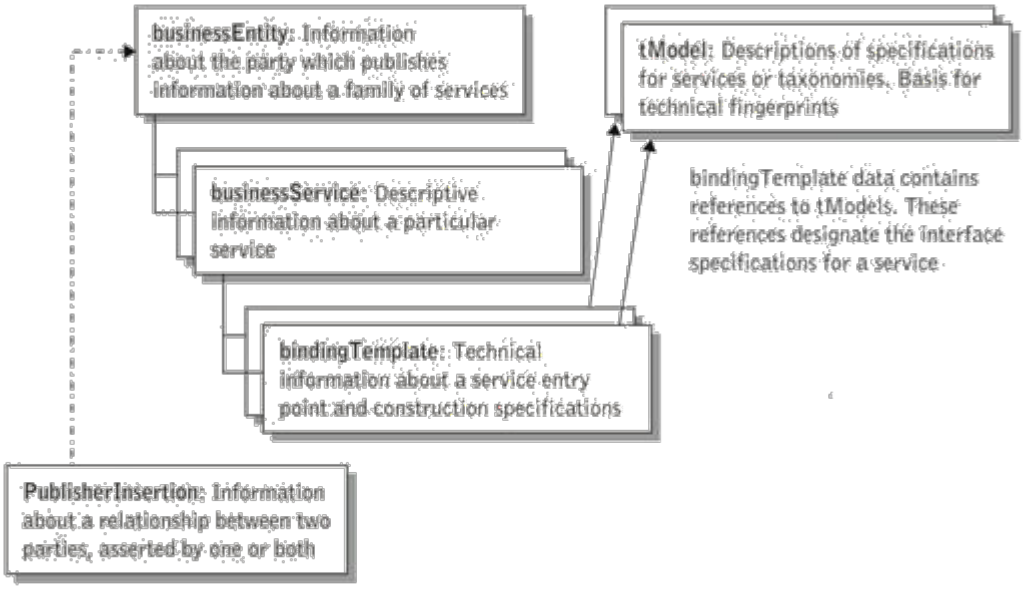


Figure 6.2 Overview of UDDI data structures

- UDDI defines a data structure standard for representing company and service description information.
- The data model used by the UDDI registries is defined in an XML schema. XML was chosen because it offers a platform-neutral view of data and because it allows hierarchical relationships to be described in a natural way.
- The UDDI XML schema defines four core types of information that provide the white/yellow/green page functions.
- These are: business entities, business services, binding templates; and information about specifications for services (technical or tModels)
- The UDDI XML schema specifies information about the business entity, e.g., a company, that offers the service (<businessEntity>), describes the services exposed by the business (<businessService>), and captures the binding information (<bindingTemplate>) required to use the service.
- The <bindingTemplate> captures the service endpoint address, and associates the service with the <tModel>s that represent its technical specifications.
- A service implementation registration represents a service offered by a specific service provider. Each <businessService> can be accessed in one or more ways.
- For example, a retailer might expose an order entry service accessible as a SOAP-based Web service, a regular Web form, or even a fax number. To convey all the ways a service is exposed each service is bound to one or more <tModel>s via a binding template.
- Each of these four core UDDI structures has for identification purposes a unique key called a universal unique identifier (UUID).
- Figure 6.3 also shows that there is a hierarchical relationship among the UDDI data structures.
-

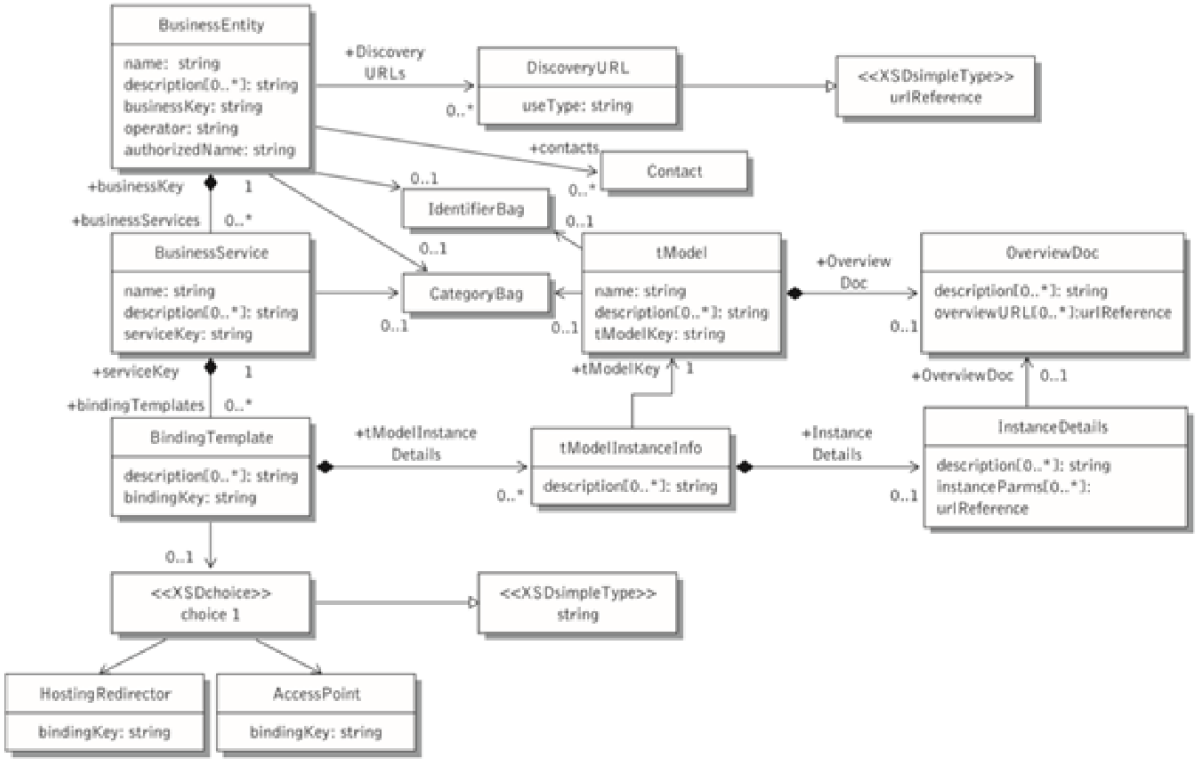


Figure 6.3 Relationships between the UDDI structures expressed in UML

- A business publishes a business entity containing, among other things, one or more business services.
- A business service has descriptive information about a service that a business provides, and can have one or more binding templates.
- The binding template has information on how to access a service entry point. It also has references to <tModel>s (using <tModel> keys) that point to the specification or interface definitions for a service.
- The interface definitions are usually in the form of WSDL definitions.
- The relationship between <tModel>s and <bindingTemplate>s is many-to-many. Accordingly, <tModel>s are not unique to <bindingTemplate>s.

OR

UDDI defines five primary data structures, which are used to represent an organization, its services, implementation technologies, and relationships to other businesses:

- A businessEntity represents the business or organization that provides the Web service.
- A businessService represents a Web service or some other electronic service.
- A bindingTemplate represents the technical binding of a Web service to its access point (its URL), and to tModels.
- A tModel represents a specific kind of technology, such as SOAP or WSDL, or a type of categorization system, such as U.S. Federal Tax ID numbers or D-U-N-S numbers. The tModels that a bindingTemplate refers to reveal the type of technologies used by a Web service. Most of the data structures also use tModels to indicate the categorization system of an identifier assigned to the structure.
- A publisherAssertion represents a relationship between two business entities.

23.	**How WSDL to USDI mapping model is done?**
ANS 23:	WSDL to UDDI mapping model ● Since both UDDI and WSDL schema have been architected to delineate clearly between interface and implementation, these two constructs will quite complementarily work together naturally. ● By decoupling a WSDL specification and registering it in UDDI, we can populate UDDI with standard interfaces that have multiple implementations, providing a landscape of business applications that share interfaces. ● The WSDL to UDDI mapping model is designed to help users find services that implement standard definitions. ● The mapping model describes how WSDL <portType> and <binding> element specifications can become <tModel>s; how the <port>s of WSDL become UDDI <bindingTemplate>s; and how each WSDL service is registered as a <businessService>. ● UDDI provides a method for publishing and finding service descriptions. ● The service information defined in WSDL documents is complementary to the information found in UDDI business and service entries. ● Since UDDI strives to accommodate many types of service descriptions it has no direct support for WSDL. ● However, since UDDI and WSDL distinguish clearly between interface and implementation, these two constructs work together quite naturally. ● The primary focus in this section is on how to map WSDL service description into a UDDI registry, which is required by existing Web service tools and run-time environments. ● In this section we use the term WSDL interface file to denote a WSDL document that contains the <types>, <message>, <portType> elements, and the term WSDL binding file to denote a WSDL document that contains the <binding> element. ● The term WSDL implementation file denotes a WSDL document that contains the <service> and <port> elements. ● The WSDL implementation file imports the interface and binding file, while the binding file imports the interface file. ● A complete WSDL service description is a combination of a service interface, service binding, and a service implementation document. ● Since the service interface and service binding represent a reusable definition of a service, they are published in a UDDI registry as a <tModel>. ● The service implementation describes instances of a service. Each instance is defined using a WSDL <service> element. ● Each <service> element in a service implementation document is used to publish a UDDI <businessService> and the service <port>s of WSDL become UDDI binding templates. ● When publishing a WSDL service description, a service interface must be published as a <tModel> before a service implementation is published as a <businessService>. ● By decoupling a WSDL specification and registering it in UDDI, we can populate UDDI with standard interfaces that have multiple implementations.

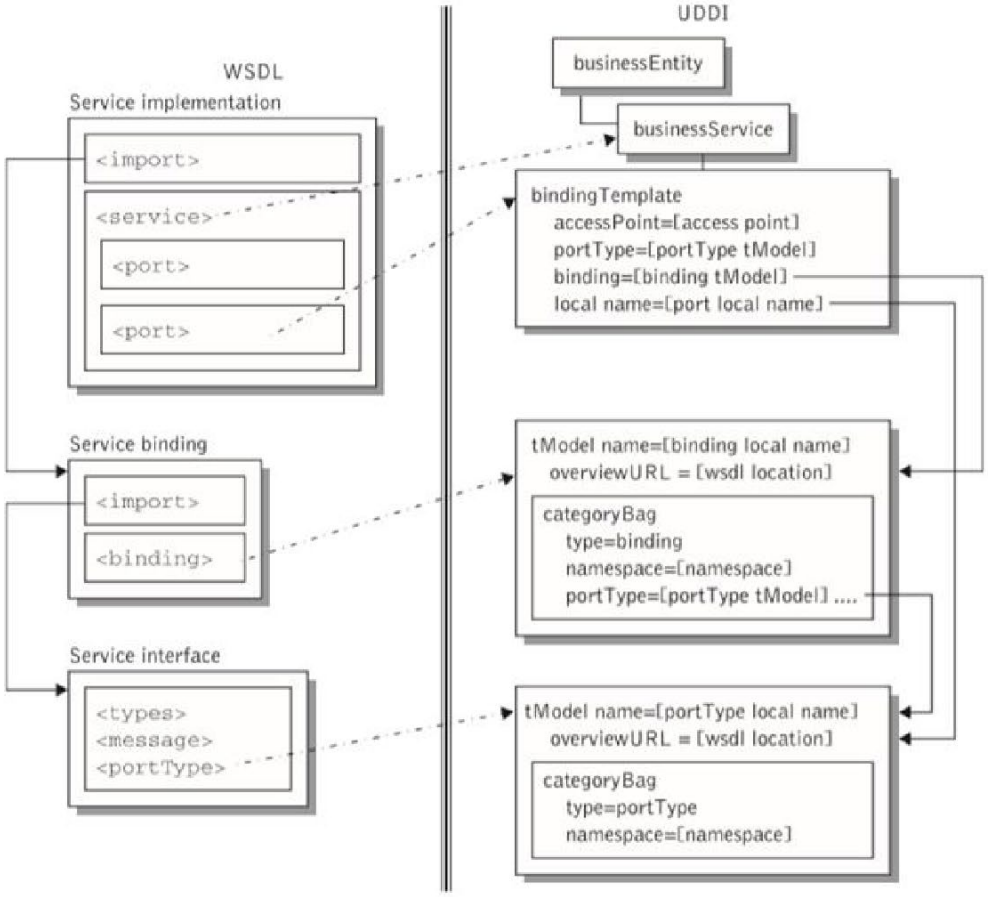


Figure 6.7 Mapping WSDL to UDDI schemas

6.3.2.3 Summary of WSDL to UDDI mapping model

This section summarizes the WSDL to UDDI mapping model. In particular, Tables 6.1 to 6.4 summarize the mapping of WSDL artifacts to the UDDI data structures `<tModel>`, `<businessService>`, and `<bindingTemplate>` according to the UDDI Technical Note [Colgrave 2004].

Table 6.1 shows that a `wsdl:portType` must be mapped to `uddi:tModel` categorized as a `<portType>`. The minimum information that must be captured about a `wsdl:portType` is its entity type, its local name, its namespace, and the location of the WSDL document that defines the `<portType>` [Colgrave 2004]. Capturing the entity

Table 6.1 Summary of mapping of `wsdl:portType` to `uddi:tModel`

WSDL	*UDDI*
`portType`	`tModel` (categorized as portType)
Local name of `portType`	`tModel` name
Namespace of `portType`	`keyedReference` in `categoryBag`
Location of WSDL document	`overviewURL`

24.	**Describe the various business information provider roles w.r.t UDDI.**
ANS 24:	• UDDI is a registry that contains relatively lightweight data. • The concept of the UDDI initiative is the UDDI business registration, an XML document used to describe a business entity and its web services.

- The information provided in a UDDI business registration consists of three interrelated components:
  - o "white pages" - includes address, contact, and other key points of contact.
  - o "yellow pages" - the classification of information according to industrial classifications based on standard industry taxonomies.
  - o "green pages" - the technical capabilities and information about services that are exposed by the business including references to specifications for Web services and pointers to various file and URL based discovery mechanisms.
- Business Partners and potential clients of an enterprise's services locate information about the services provided would normally have as a starting point a small set of facts about the service provider.
- For example, either its business name or perhaps some key identifiers, as well as optional categorization and contact information (white pages).
- The business entity element and business key attribute.
- The core XML elements for supporting publishing and discovering information about a business-the UDDI Business Registration - are contained in an element named <businessEntity>.
- This XML element serves as the top-level structure and contains white page information about a particular business unit (service provider).
- The <businessEntity> structure can be used to model any businesses and providers within UDDI.
- For instance, each <businessService> contained within a <businessEntity> structure describes a logical service offered by the business.
- Similarly, each <businessTemplate> contained within a given <businessEntity> provides the technical description of a Web service that belongs to the logical service that is described by the <businessService>.

25. Write a short note on Web Service Development Lifecycle.(from ppts)

ANS 25:

- Web services development lifecycle (SDLC), or service-oriented design and development, is a highly iterative and continuous approach for developing, implementing, deploying, and maintaining Web services in which feedback is continuous cycle of to and from phases in iterative steps of enhancement.
  - + Managing the entire services lifecycle – includes analyzing, identifying, designing, developing, deploying, finding, applying, evolving, and maintaining services.
  - + Establishing a platform and programming model, which includes connecting, deploying, and managing services within a specific run-time platform.
  - + Adopting best practices and tools for architecting services-oriented solutions in repetitive, predictable ways that deal with changing business needs. This includes mining existing applications to discover potential services, repurposing existing assets and functionality to extend their utility and make those capabilities accessible as services, creating new services, and “wiring” together services by connecting behaviour exposed through their interfaces.
  - + Delivering high-quality workable service-oriented solutions that respect QoS requirements. These solutions may be implemented as best practices, such as tried and tested methods for implementing security, ensuring performance, compliance with standards for interoperability, and designing for change.
- A fundamental goal to the above capabilities is that business goals and requirements should always drive downstream design, development, and testing to transform business processes into composite applications that automate and integrate enterprises.
- Thus, business requirements can be traced across the entire lifecycle from business goals, through software designs and code assets, to composite applications.

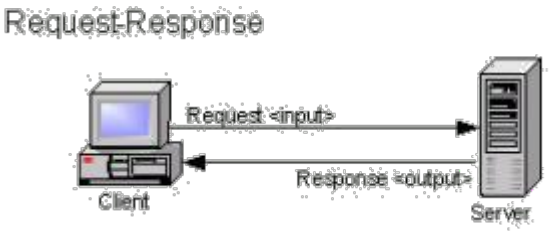
- The SDLC methodology encompass planning, analysis, design, construction may be built using a blend of forward and reverse-engineering techniques or other means to facilitate the needs of the business.

26. How does a one-way operation differ from a request/ response operation?

ANS 26:

- The Request-Response Pattern
  - + The request-response operation includes one input element, which is the client's request to the server, followed by one output element, which is the server's response back to the client.

FIGURE 1: THE REQUEST-RESPONSE MESSAGE PATTERN



  - + In WeatherSummary.wsdl, the request-response operation is defined by the getSummary operation:

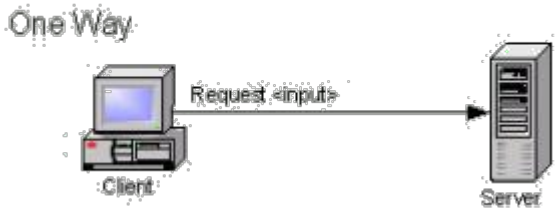
```
...
<operation name="getSummary">
        <input message="tns:getSummary"/>
        <output
message="tns:getSummaryResponse"/>
</operation>
...
```

  - + The operation **getSummary** contains an input message getSummary, which is the client request, and an output message getSummaryResponse, which is the server response. The contents of these messages are then defined in the WSDL's message element:

```
<message name="getSummary">
        <part name="zipcode"
type="xsd:string"/> </message>
<message name="getSummaryResponse">
        <part name="weatherData"
type="wsx:WeatherSummary"/> </message>
```

  - + The message getSummary contains one part, a zipcode of type xsd:string. The client request thus provides a zipcode to the server.
  - + The message getSummaryResponse contains one part, a weatherData object of type wsx:WeatherSummary. Based on the zipcode, the server return an XML WeatherSummary element for the location specified by the zipcode.
  - + A WeatherSummary element is a complex type defined by an embedded XML Schema in the types element of the WSDL file
  - + The implementation of this pattern occurs in the client-side WeatherSummaryClient.cpp, a sample implementation that uses the generated class in WeatherSummaryProxy.cpp, and in the server-side WeatherSummaryImp.cpp, the server implementation that receives the request and sends a response.
- The One Way Pattern
  - + The one-way operation includes one input element, which is the client's request to the server. No response is expected.

FIGURE 2: THE ONE-WAY MESSAGE PATTERN



+ In WeatherSummary.wsdl, the one-way pattern is defined by the operation updateWeather:

```
<!-- One-way -->
<operation name="updateWeather">
        <input
        message="tns:updateWeather"/>
</operation>
```

- The operation updateWeather consists of one input message containing several parts.

```
<message name="updateWeather">
        <part name="host" type="xsd:string"/>
        <part name="port" type="xsd:string"/>
        <part name="transportName" type="xsd:string"/>
        <part name="weatherData"
type="wsx:WeatherSummary"/> </message>
```

- The message contains a WeatherSummary element that provides updated weather conditions for the zip code.
- The implementation of this pattern occurs in the client-side WeatherSummaryClient.cpp, a sample implementation that uses the generated class in WeatherSummaryProxy.cpp, and in the server-side WeatherSummaryImp.cpp, the server implementation that receives the request.

OR

- *WSDL One-Way Operation*

A one-way operation example:

```
<message
name="newTermValues">
 <part name="term"
 type="xs:string"/> <part
 name="value" type="xs:string"/>
</message>
<portType
 name="glossaryTerms">
 <operation name="setTerm">
  <input name="newTerm"
 message="newTermValues"/> </operation>
</portType >
```

In the example above, the portType "glossaryTerms" defines a one-way operation called "setTerm". The "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value". However, no output is defined for the operation.

- *WSDL Request-Response Operation*

A request-response operation example:

```
<message
name="getTermRequest">
 <part name="term"
type="xs:string"/> </message>
```

```
<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

In the example above, the portType "glossaryTerms" defines a request-response operation called "getTerm".

The "getTerm" operation requires an input message called "getTermRequest" with a parameter called "term", and will return an output message called "getTermResponse" with a parameter called "value".