Service Oriented Architecture
Web Services

Sergio López Bernal

2022-2023

Objectives

- SOA Concepts and advantages
- Web Services description
- WS-* technologies:
  - XSD
  - SOAP
  - WSDL

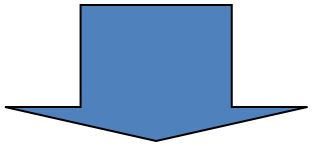
Interoperability

- Need for the interoperability among different applications and resources.
  - Applications programmed in different programming languages.
  - Example: Try to communicate a C application and a Java application.
  - Could we send an “Enumerate” object from the Java application to the C one?
- Need for:
  - A common data structure (based on XML)
  - A common transport.

- Emergence of SOA (Services Orientation Architecture)

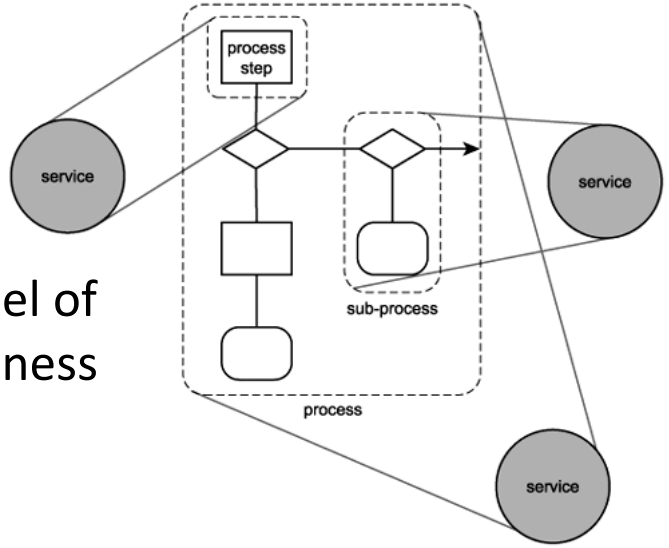
The Roadmap to SOA

- **Technology-dependent solutions**: with new technology is neccesary to re-program.
- **Problems with the data structure**. If we exchange objects from a specific languages it is difficult to make interoperation with other application based on a different language.
- Good for integration but not for the cooperation of services.

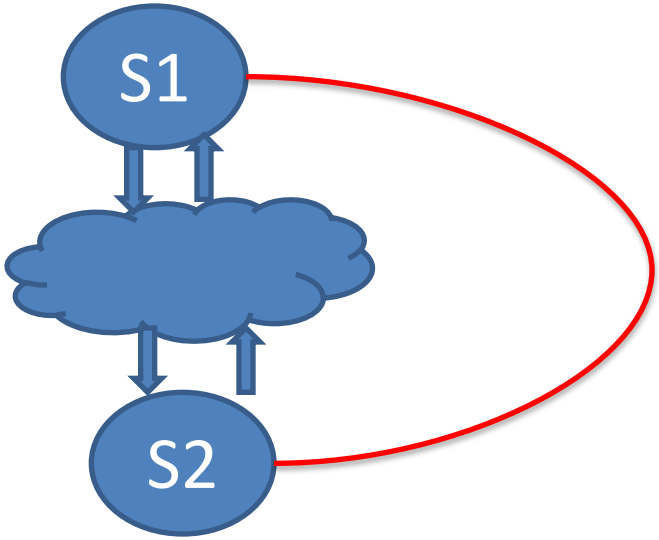


- XML is the standard most used for interoperability.
- However, each organization offers different specifications and standards.

SOA Model



- Software architecture which defines a decouple model of services to support the requirements of modern business processes and software
- Present a model in which the business logic is composed of some “small units”, each of them maintaining its own logic
- Moreover the small units can be distributed
- Examples: Authentication Service, Repository Service, Users Management Service, Registration Process
- The computing based on services is named as Service Oriented Computing (SOC)



Advantages of SOA

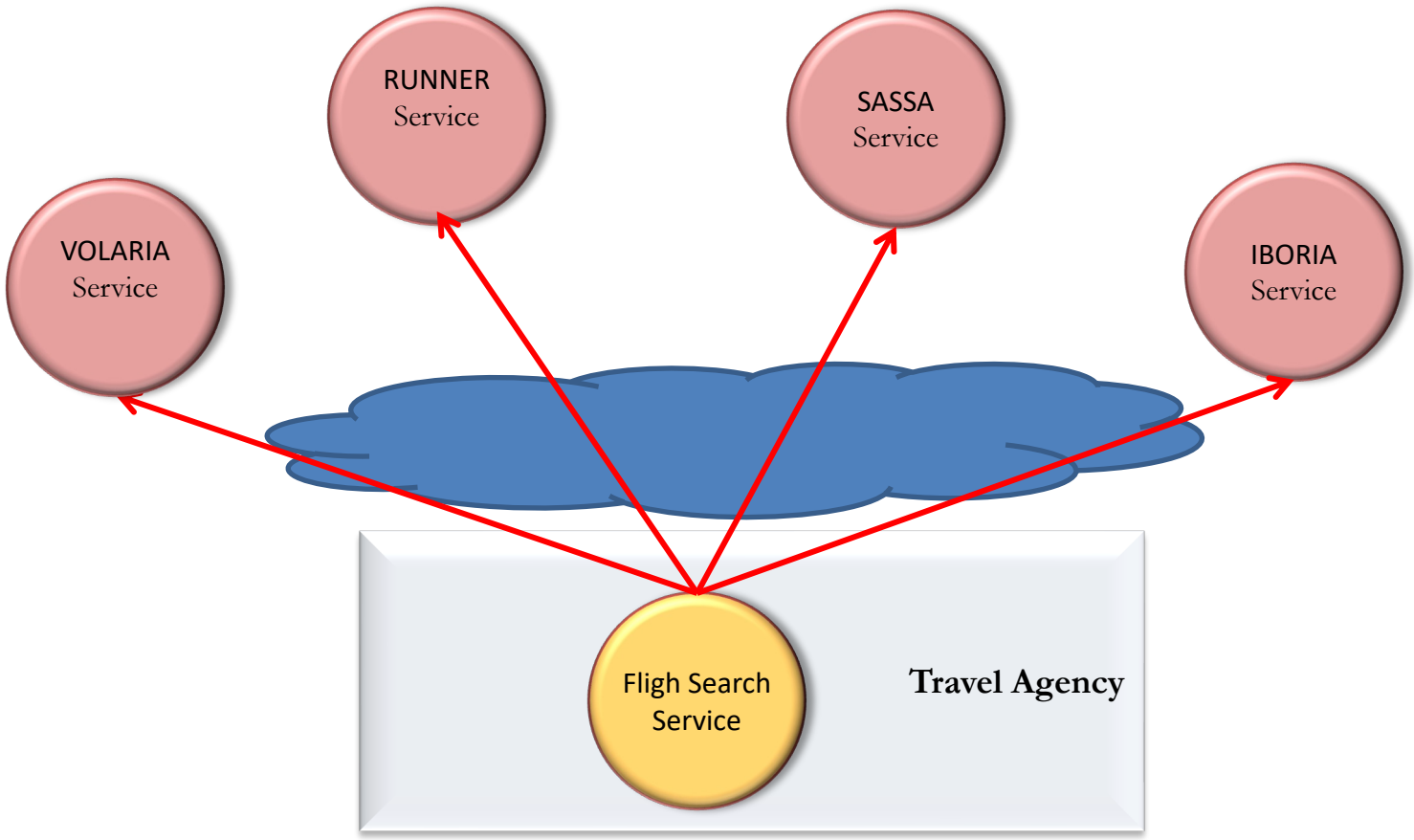
- **Improve integration**:
  - Following the same protocol and data structure we can communicate and link new services to existing applications.

- **Improve Reusability**:
  - The same service can be used in the building of several applications (e.g., Authentication Service).

- **Organizational Agility**:
  - Technology evolves over time, thus enterprises following SOA paradigm can integrate new software in a simpler way.
  - Therefore, the cost of the integration is lower.

SOA Example

- Travel Agency which search flights in different companies:

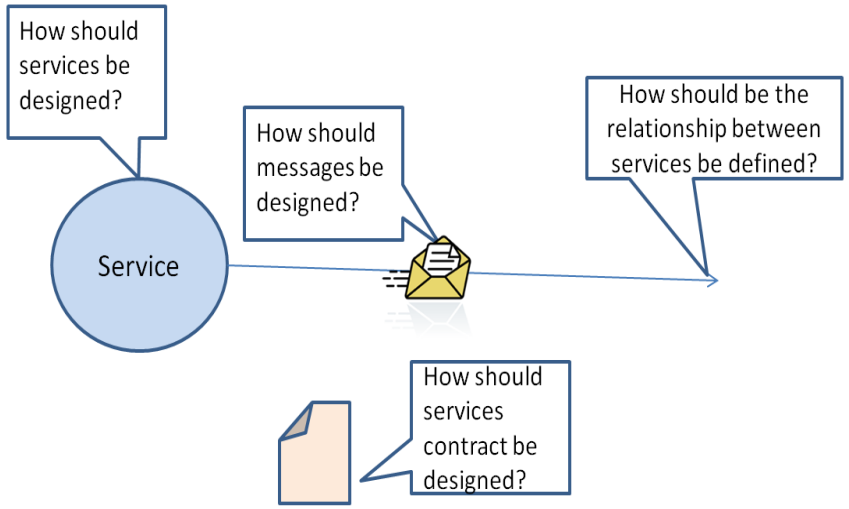


Principles of Service Orientation

- Loose coupling Services maintain a relationship that minimizes dependencies and only requires that they retain an awareness of each other.
- Service contract Services adhere to a communications agreement, as defined collectively by one or more service descriptions and related documents.
- Autonomy Services have control over the logic they encapsulate.
- Abstraction Beyond what is described in the service contract, services hide logic from the outside world.
- Reusability Logic is divided into services with the intention of promoting reuse.
- Composability Collections of services can be coordinated and assembled to form composite services.
- Statelessness Services minimize retaining information specific to an activity.
- Discoverability Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms

Services Communication

- **Some fundamentals of services**
  - How services relate
    - With other services ➔ **Services Composition**
    - With Programs➔ **Client invokation**
  - How services are designed
    - **Service description** (“agreement”)
      - Service name
      - Input data
      - Output data
  - How services communicate
    - **Messages**
      - Independent units of communication



Web Services: Definition

- Web.- Flexible Human-Machine Interaction.
- According to the W3C, “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network”.
- They are based on open standards (SOAP, REST, WSDL)
- Growing interest in the research area, business companies, etc.
- Impact in several areas such as e-commerce, e-Grid, Business Process Management (BPM), health systems, govern application, education, …
- New trends emerging from Web Services are: Cloud Computing and Services Computing

Web Services

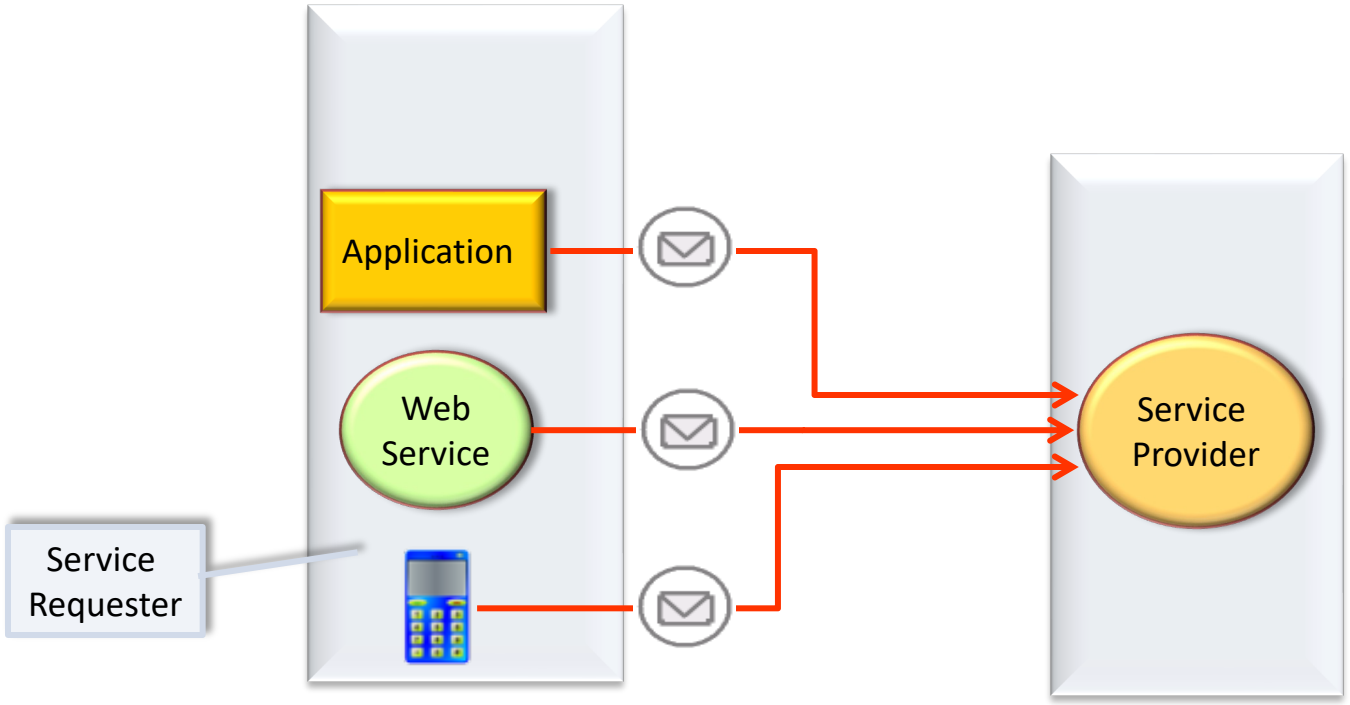
- **Two important trends in web services**
  - **Web Services based on SOAP or WS-* services**
    - Simple Object Access Protocol (SOAP): XML standard protocol based on the exchange of messages
    - Several technologies and standards have been devoloped over it: WSDL , BPEL and UDDI
  - **RESTful Web Services** (Representational Style Transfer)
    - Based on the Web architecture
    - Use URI to identify resources
    - Uniform interface for all the resources. Methods: GET, POST, PUT, DELETE

SOAP vs REST

Dimension	SOAP	REST
Design	Standard	Loose guidelines and recommendations
Approach	Function-driven	Data-driven
Statefulness	Stateless by default, but can be stateful	Stateless
Security	WS-Security + communications	Communications
Performance	Higher resources	Fewer resourcers
Messages	Only XML	Plain text, HTML, XML, JSON, YAML, etc
Transfer protocols	HTTP, SMTP, UDP, etc	Only HTTP
Advantages	Secure, standard, extensible	Scalable, high performance, browser-friendly, flexible
Disadvantages	Lower performance, higher complexity, lower flexibility	Less secure
Common use	Enterprise, high security apps, finances, payment gateways, telecommunications	Public APIs for web services, mobile services, social networks

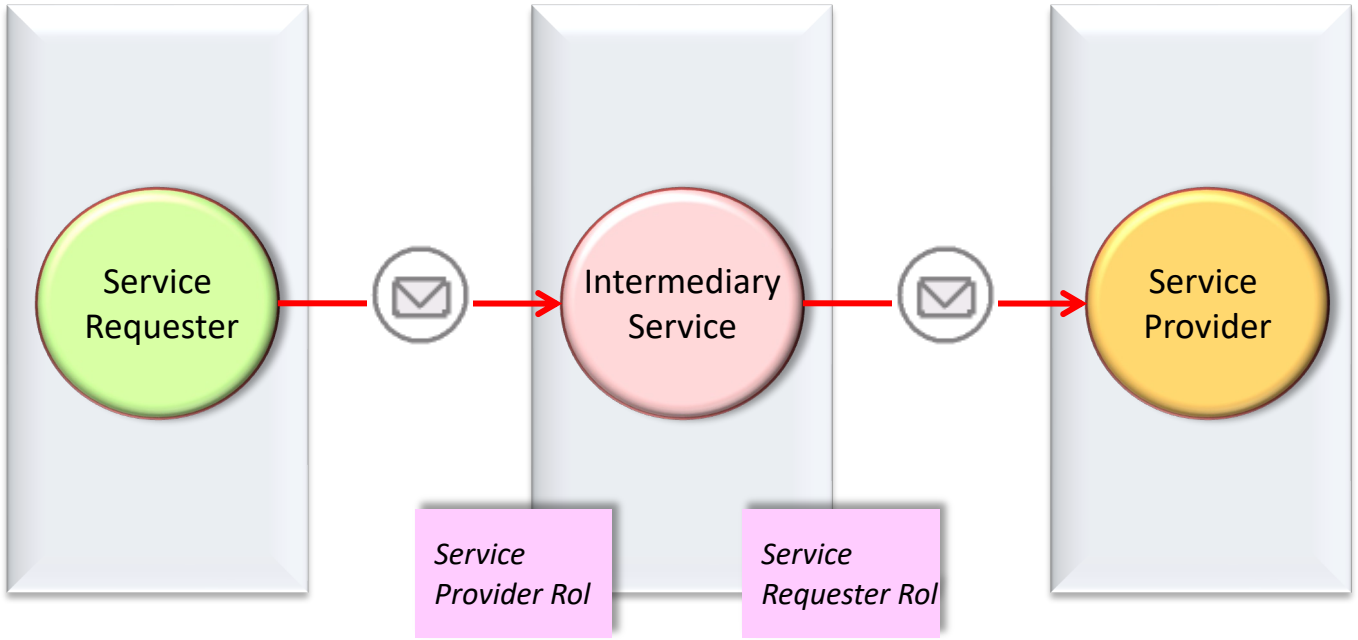
Web Services Roles

- Service Provider: The service which provides functionalities.
- Service Requester: The service or program asking for some operation in the service.



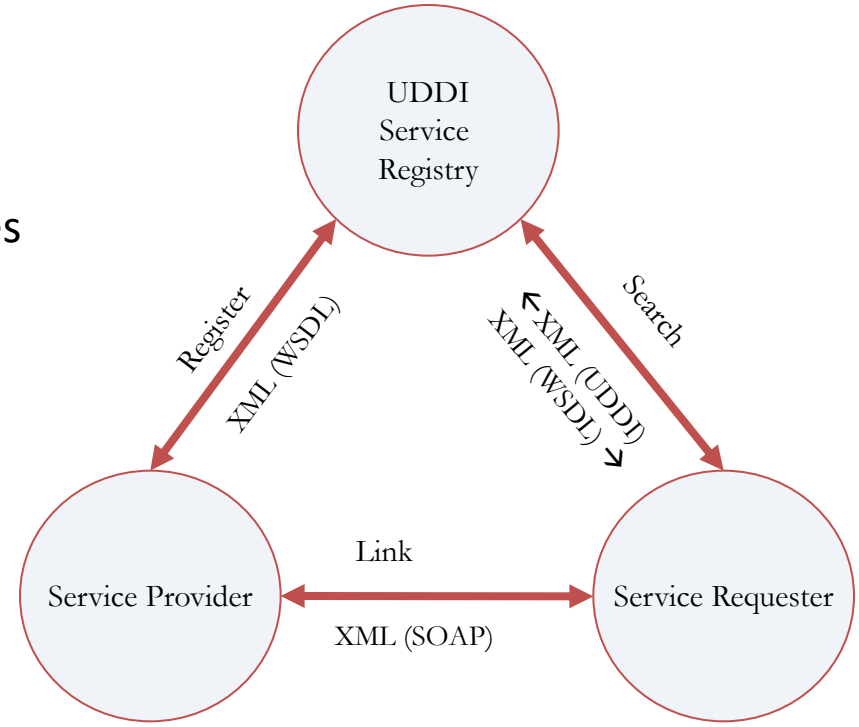
Web Services Roles

- Intermediary Service:
  - Passive: typically responsible for routing messages to a subsequent location. It does not modify the message.
  - Active: also route messages to a forwarding destination. However, these services actively process and alter the message contents.



Early Approach to SOA (WS-*)

- Based on XML
  - SOAP: Provide a mechanism for sending messages for the exchange of data between the Service Requester and the Service Provider.
  - WSDL: Language for describing the services (operations, inputs, outputs)
  - UDDI: Specification to store and search services.
- Majority of development use WS-* technologies without using UDDI
- However, UDDI has taken quite relevance in the research field



Searching new services

- Problems when using syntactic information:
  - Different ways of expressing the same word (name, first_name, FirstName…)
  - Different ways of writting the XML files

- Use of semantic ontologies for annotating the services

Web Services Technologies

- XML Schema: Standard structure of XML files
- SOAP: Protocol for the exchange of messages
- WSDL: Language for describing the services

XML Schema

- **.XSD**
- Meta-language to describe the structure of XML documents (parameters, variables, …)
- Allow to describe the structure of:
  - XML documents of the service
  - Structure of the messages to exchange (inputs and outputs)
- **Languages Elements:**
  - Simple Types
  - Complex Types
  - Elements and Atributtes

Simple Types

- Primitive Types: string, boolean, integer, float, date …

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

- Restrictions: Restrictions are used to define acceptable values for XML elements

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Complex Types

- Mechanism for creating complex data
- Can contain:
  - Simple Types
  - Other Complex Types
  - “Aggregation” models
    - Sequence.- ordered list of elements
    - Choice.- for choosing one element of the set
    - All.- a sequence but not ordered.

Complex Types

Example:

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

XML-Schema

- Definition of a book (libro.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="Libro">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Titulo" type="xsd:string"/>
                <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
                <xsd:element name="Editorial" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="precio" type="xsd:double"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

XML Schema

- Elements (libros.xml)

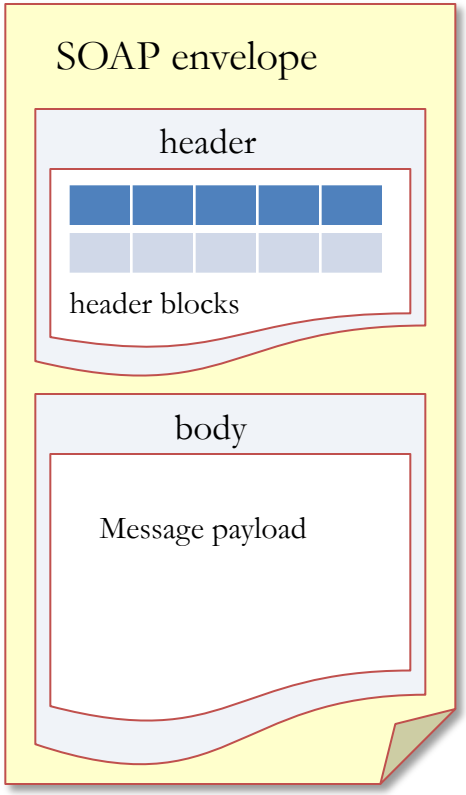
```
<?xml version="1.0" encoding="UTF-8"?>
<Libro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="libro.xsd" precio="20">
    <Titulo>Fundamentos de XML Schema</Titulo>
    <Autores>Allen Wyke</Autores>
    <Autores>Andrew Watt</Autores>
    <Editorial>Wiley</Editorial>
</Libro>
```

SOAP

- Protocol based on XML for the exchange of messages between several services
- Frequently, it uses HTTP to transport the messages between services
- Described in XML, thus, it is technology independent and can be transported by any protocol
- Implementation over JMS, SMTP, …
- Each SOAP messages is described in a container called *envelope*

Structure of SOAP message

- SOAP:Header
  - Special rules or information to indicate some actions to deal with such as security, trust or context
  - Some examples of the use of header:
    - processing instructions that may be executed by service intermediaries or the ultimate receiver
    - routing or workflow information associated with the message
    - security measures implemented in the message
    - reliability rules related to the delivery of the message
- SOAP:Body
  - Content of the message (XML payload)



- Two invokation styles: RPC Style, Document Style (Recommended)
  - Document Style has to follow an XML schema

Example of SOAP Message

Converring currency, 23 dolars to euro/Document Style

```
<soapenv:Envelope xmlns:q0="urn:umu:masterNTI" … needed xml
    schemas…>
<soapenv:Header>   </soapenv:Header>
<soapenv:Body>
       <q0:convertCurrency>
        <quantity>23</quantity>
        <currencyIn>dolar</currencyIn>
        <currencyOut>euro</currencyOut>
      </q0:convertCurrency>
</soapenv:Body>
</soapenv:Envelope>
```

Example SOA Message with security header

….

```
<soapenv:Header>
  <wsse:UsernameToken>
    <wsse:Username>scott</wsse:Username> <wsse:Password
    Type="wsse:PasswordDigest">
    KE6QugOpkPyT3Eo0SEgT30W4Keg=</wsse:Password>
    <wsse:Nonce>5uW4ABku/m6/S5rnE+L7vg==</wsse:Nonce>
    <wsu:Created xmlns:wsu=
    "http://schemas.xmlsoap.org/ws/2002/07/utility"> 2002-08-
    19T00:44:02Z </wsu:Created> </wsse:UsernameToken>
</soapenv:Header>
```

…

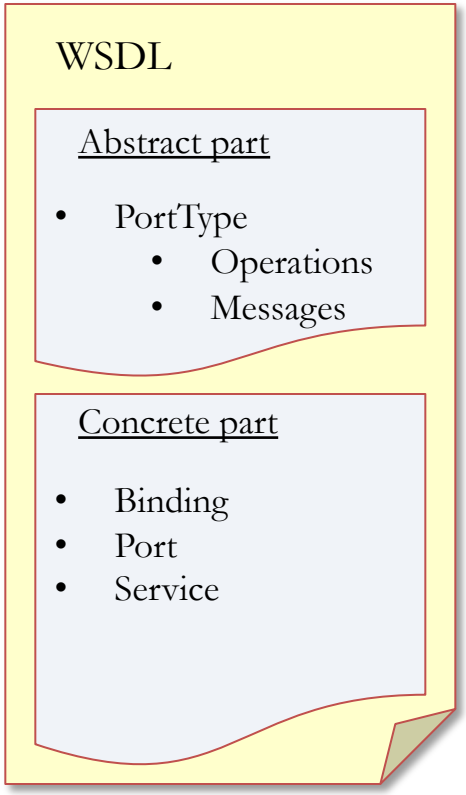
WSDL (Web Service Description Language)

- Describe:
  - Operations (services) offered by the service
  - Where to find the service
  - How to invoke, what services are available and what types of data is needed
- Represent an agreement between the service requester and the service provider
- Technology independent (C, Java, Perl …)
- Transport-protocol independent (http, smtp, jms, soap, xml-rpc…)

WSDL Structure

- WSDL 1.0 (used in Business Process Execution Language (BPEL)), WSDL 2.0

- Contains two parts:
  - Abstract part:
    - Types, interfaces and Message Exchange Patterns
  - Concrete part:
    - Connection with the concrete technology



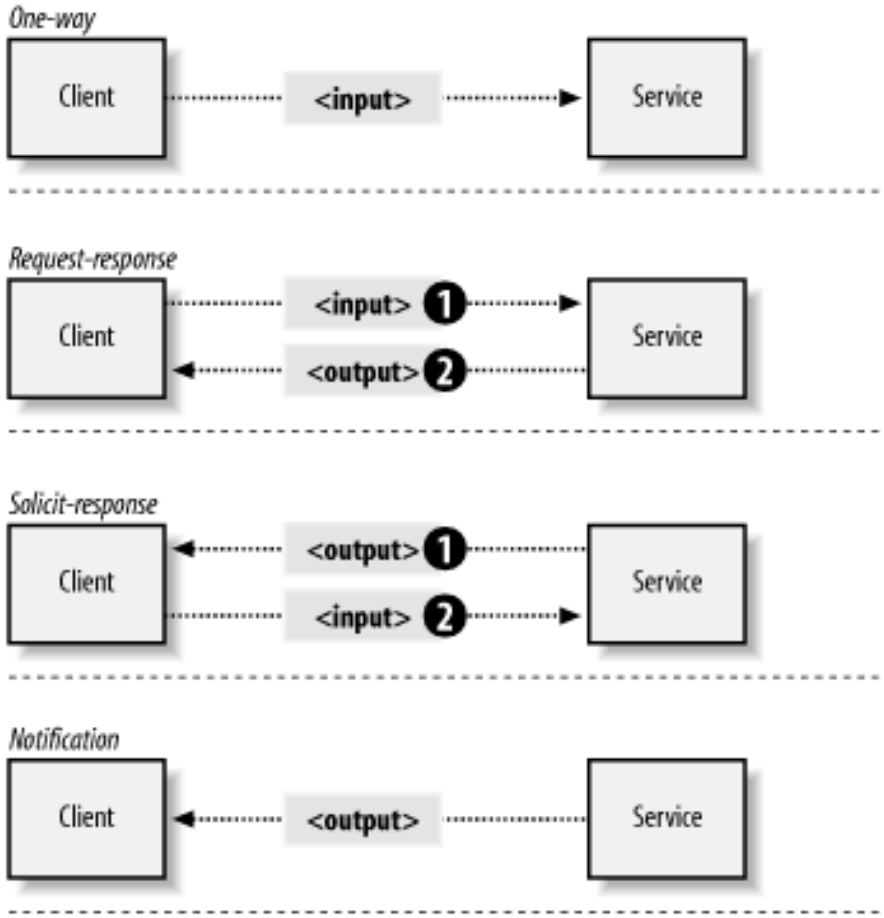
WSDL

- Abstract part
  - **portType**
    - High-level view of the service interface by sorting the messages a service can process into groups of functions known as operations
    - Analogy with clasess or libraries, …
  - **Operation**
    - Specific actions performed by the service
    - Similar to methods in a class
    - Consists of input and/or output messages
  - **Message**
    - Data structures represented in XML Schema
    - They are linked to the input and/or output of the operations (paramenters)

WSDL

Message Exchange Patterns (MEP)

- MEP represents the way in which the services are related.
- 4 patterns WSDL 1.0
  - **One-way**
  - **Request- response**
  - **Solicit-response**
  - **Notification**
- WSDL 2.0 includes 4 more patterns, but they are based on the previous one including exceptions



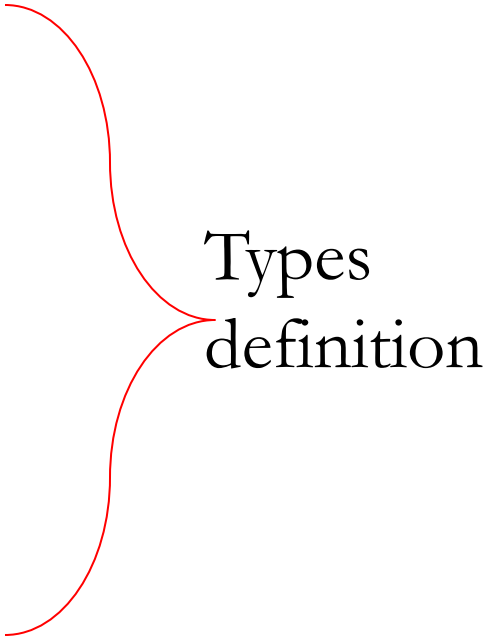
WSDL

- Concrete part
  - Indicate aspects such as the concrete transport protocol
  - **Binding**
    - Specify a transport technology that the service can use
      - REST, HTTP or SOAP (the most common)
    - Can be applied to all the interface or a specific operation
  - **Service** and **port**
    - Defines the address or connection point to a Web service. It is typically represented by a simple URL string.

Example: Defining Abstract part

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/ ..>
 <wsdl:types>
..
    <xsd:simpleType name="currencyType">
     <xsd:restriction base="xsd:string">
            <xsd:enumeration value="dolar"></xsd:enumeration>
            <xsd:enumeration value="euro"></xsd:enumeration>
            <xsd:enumeration value="yen"></xsd:enumeration>

     </xsd:restriction>
    </xsd:simpleType>
  </xsd:schema>
 </wsdl:types>
```



Example: Defining Abstract part

```
<wsdl:message name="convertCurrencyRequest">
  <wsdl:part element="tns:convertCurrency" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="convertCurrencyResponse">
  <wsdl:part element="tns:convertCurrencyResponse" name="parameters"/>
 </wsdl:message>

 <wsdl:portType name="WSCurrencyConverter">
  <wsdl:operation name="convertCurrency">
   <wsdl:input message="tns:convertCurrencyRequest"/>
   <wsdl:output message="tns:convertCurrencyResponse"/>
  </wsdl:operation>
 </wsdl:portType>
```

Messages Definition

Operations Definition

Example: Defining Concrete Part

```
<wsdl:binding name="WSCurrencyConverterSOAP”     type="tns:WSCurrencyConverter">
    <soap:binding style="document”
            transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="convertCurrency">
            <soap:operation
                    soapAction="urn:umu:masterNTI/convertCurrency" />
            <wsdl:input>
                    <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                    <soap:body use="literal" />
            </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="WSCurrencyConverter">
  <wsdl:port binding="tns:WSCurrencyConverterSOAP" name="WSCurrencyConverterSOAP">
    <soap:address location="http://www.example.org/"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

References

1. “RESTful Web Services vs. “Big” Web Services:Making the Right Architectural Decision”**.** Cesare Pautasso, Olaf Zimmermann and Frank Leymann (2008)
2. “SOA: Principles of Service Design”. Thomas Erl. Prentice Hall. ISBN: 0-13-234482-3 (2007)
3. “Service-Oriented Architecture: Concepts, Technology, and Design”. Thomas Erl. Prentice Hall (2005)

Links

- W3 Schools. Full web building tutorials. http://www.w3schools.com/
- XML Schema specification. http://www.w3.org/XML/Schema
- World Wide Web Consortium. http://www.w3c.es