MONASH University

Information Technology

FIT5183: Mobile and Distributed Computing Systems (MDCS)

Lecture 3A
SOAP, WSDL & UDDI

Outline

- Role of SOAP, WSDL and UDDI in Web Services
- Web Services Description Language (WSDL)
- Universal Description, Discovery and Integration (UDDI)

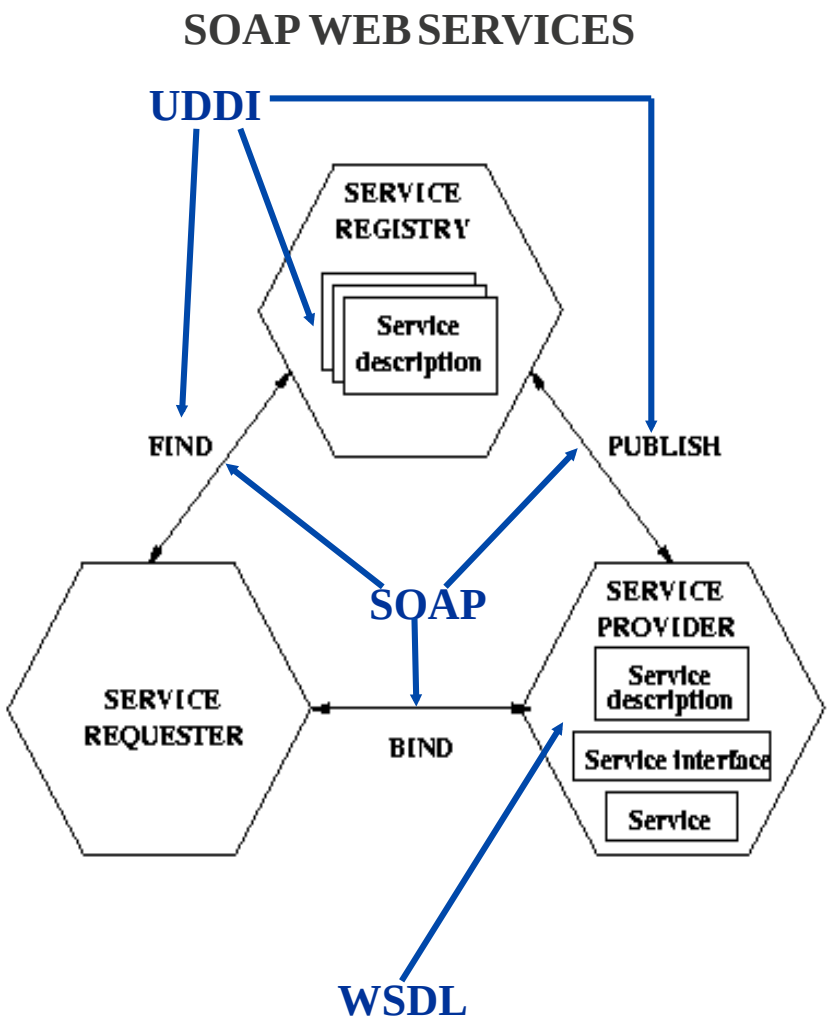
Role of SOAP, WSDL and UDDI in Web Services

Role of SOAP in Web Services

IBM's *Web service architecture*

- SOAP Web Service Definition by W3C:

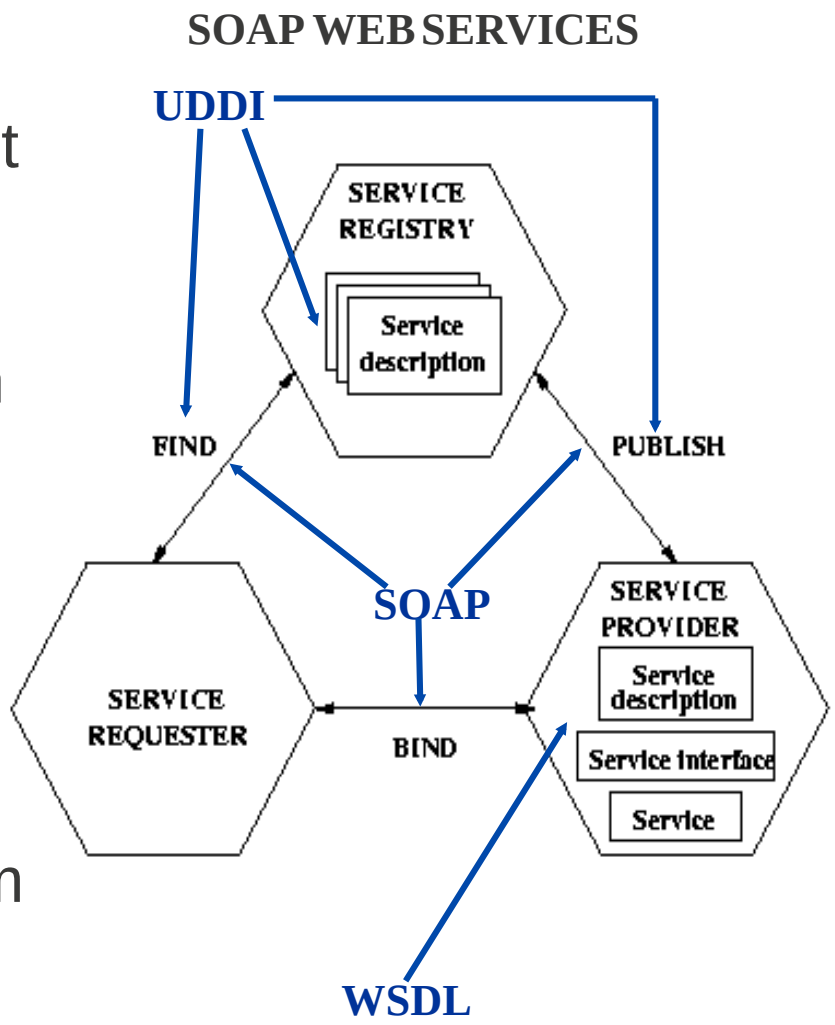
  "a software application **identified by a URI**, whose **interfaces and bindings** are capable of being defined, described, and discovered as **XML artifacts**."



Where does WSDL fit into Web Services?

IBM's *Web service architecture*

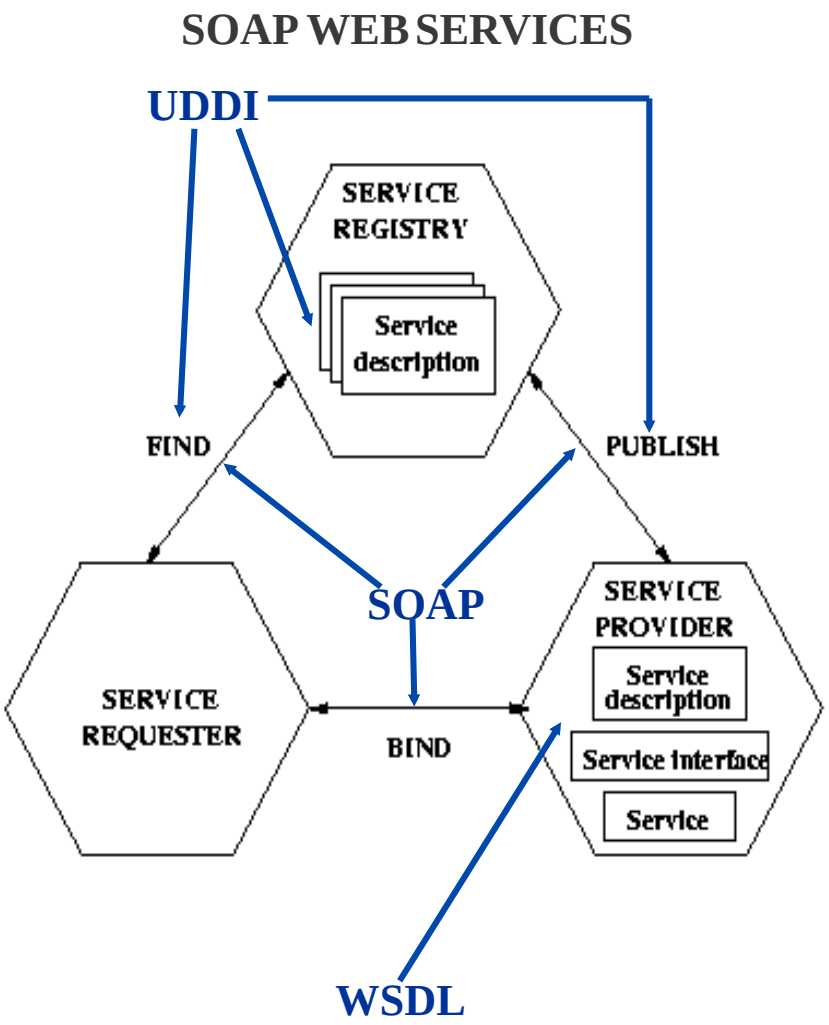
- Once it is possible to interact with any Service using the SOAP protocol we still need to **Describe the Services**, in particular their interfaces.
- WSDL is used to automatically generate the **client stubs** and **server skeleton** code, shielding details of the distribution from the developer.



Where does UDDI fit into Web Services?

IBM's *Web service architecture*

- Service **requester:** The potential user of a service
- Service **provider**: The entity that implements the service and offers to carry it out on behalf of the requester
- Service **registry**: A place where available services are listed and which allows providers to advertise their services and requesters to query for services



Web Services Description Language (WSDL)

Role of WSDL

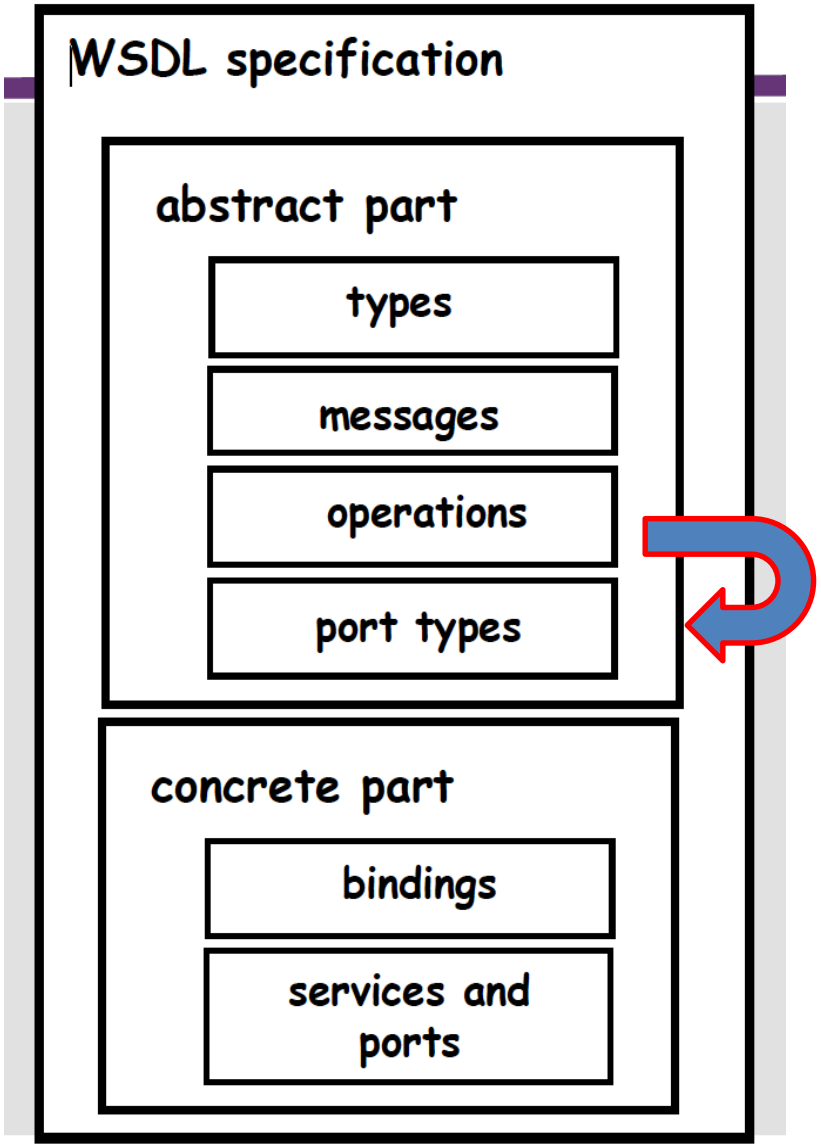
- Given that the client and service are different entities – we need to know:
  - where is the service?
  - what does it offer me?
  - what XML should I send/receive to interact with it?
  - how can I expose it to others?

What is WSDL ?

- WSDL: an **XML-based language for describing Web services and how to access them**.
- WSDL stands for **Web Services Description Language**
- WSDL is written in XML
- WSDL is an XML document
    - The document describes a Web service. It specifies the location of the service and the operations (or methods) the service exposes.
- WSDL is used to **describe** Web services
- WSDL is also used to **locate** Web services

WSDL Specification

- WSDL documents are written in XML
- WSDL documents conform to the XML Schemas defined in the WSDL standards http://www.w3.org/standards/techs/wsdl#w3c_all
- The WSDL Schemas introduce the six elements in the picture as `definitions`
- Abstract definitions have no concrete bindings, XML message encodings or services implementing port types/operations.



WSDL Document Elements

ABSTRACT PART:

<**types**> - The data types used by the web service

<**message**> - The messages used by the web service

<**portType**> - The operations performed by the web service

CONCRETE PART:

<**binding**> -The communication protocols used by the web service

<**service**> - Which binding to use and Location of the service

```
<definitions>
<types>
  definition of types…
</types>
<message>
  definition of a message…
</message>
<portType>
  definition of a port…
</portType>
<binding>
  definition of a binding…
</binding>
<service>
  definition of the
service....
</service>
</definitions>
```

WSDL Types

- The <**types**> element defines the data type that are used by the web service
- As with any Interface Description Language (IDL) this is required so that the data being exchanged can be correctly interpreted at both ends of the communication.
- Therefore the first step in defining a WSDL interface is to identify and define all the data structures that will be exchanged as parts of messages.
- WSDL uses the **XML Schema syntax** to define data types

WSDL Messages

- The messages let clients know about the input and output
- Each message is a series of name/type pairs
- The **<message>** element defines the **parts** of each message and the associated data **type**s

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

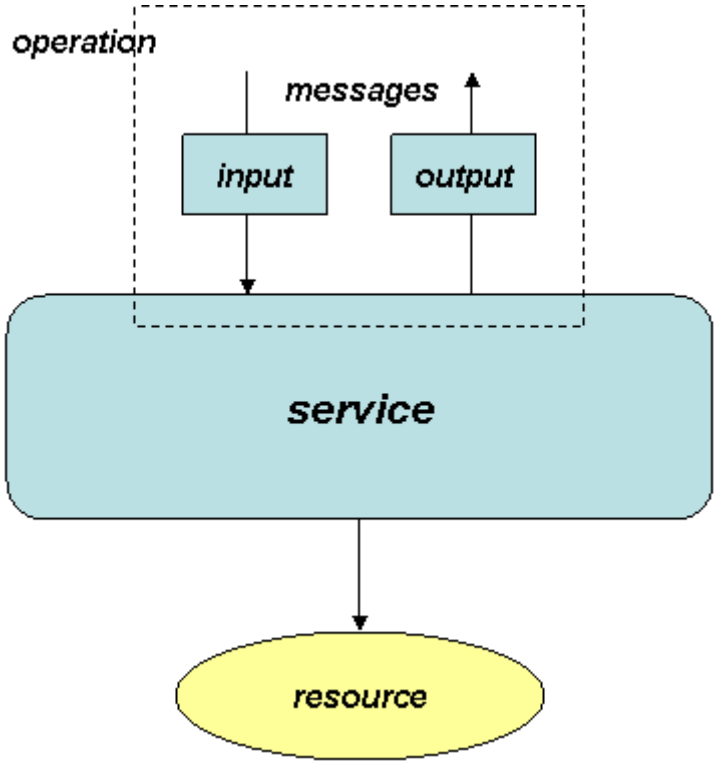
WSDL PortType

- The **<portType>** element describes the operations that can be performed, and the messages that are involved
- Each port type is a group of **operations**

```
<portType name="glossaryTerms">
    <operation name="getTerm">
        <input message="getTermRequest"/>
        <output message="getTermResponse"/>
    </operation>
</portType>
```

Message and Operations

- Messages are built on typed data structures, described by a set of names and types defined in XML schema.
- Operations define the interaction style for synchronous and asynchronous messaging between clients and servers



Operation Types

- The <portType> element defines the operations that can be performed, and the messages that are involved.
- WSDL defines the following four types of operations:

  The first pair are initiated by clients while the second are initiated by servers

Type	Definition
One-way	The operation can receive a message but will not return a response
Request-response	The operation can receive a request and will return a response
Solicit-response	The operation can send a request and will wait for a response
Notification	The operation can send a message but will not wait for a response

WSDL Bindings

- The <binding> element defines the message format and protocol details
- The binding element has the *name* and type attributes that point to the port of binding
- The soap:binding element has the *style* and the *transport* attributes
- The *style* attribute can be "rpc" or "document“
- The transport attribute defines the transport protocol to use (e.g. http)
- The *operation* element defines each operation that the port exposes

```
<binding type="glossaryTerms" name="b1">
    <soap:binding style="document“ transport="http://schemas.xmlsoap.org/soap/http" />
        <operation>
            <soap:operation soapAction="http://example.com/getTerm"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
</binding>
```

*SoapAction Http header Value

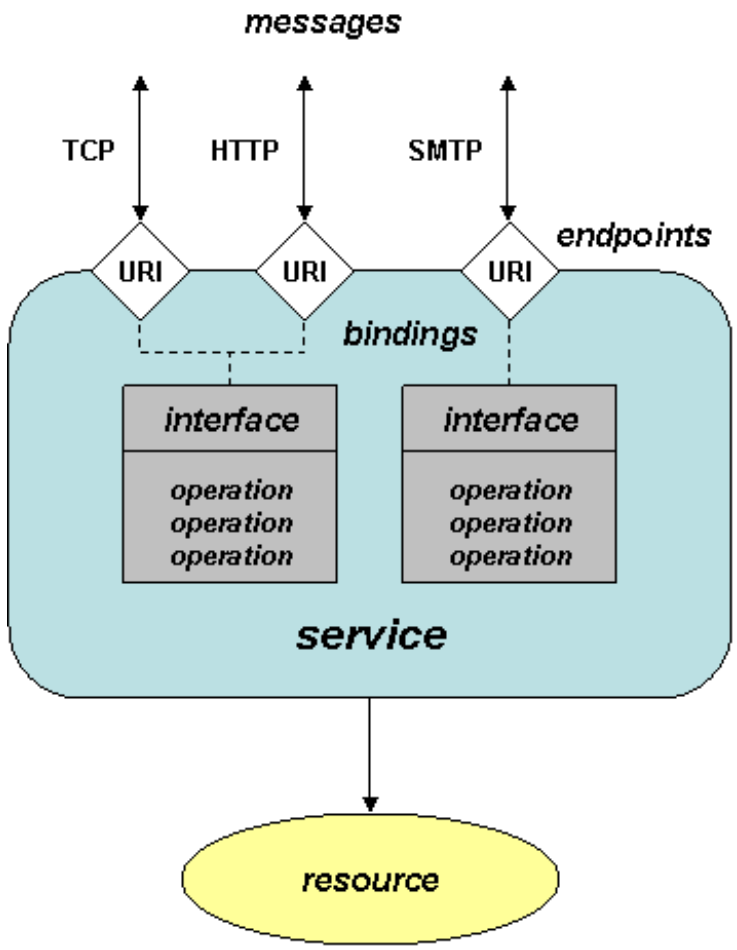
WSDL Service and Ports

- The <service> element defines
  - through which port(s) to access the web service
  - which binding to use
- The <port> element is contained within Service
- The <port> element defines the connection point to a web service
- Each port has a name and is assigned to a binding
- Within the port element, the address details are defined for that specific binding

```
<service name="MathService">
    <port name="MathEndpoint" binding="y:MathSoapHttpBinding">
     <soap:address  location="http://localhost/math/math.asmx"/>
    </port>
</service>
```

Service and Bindings

- A service can have multiple bindings for a given interface/portType
- Each binding can be only accessible at a unique URL (endpoint/port)



WSDL Example

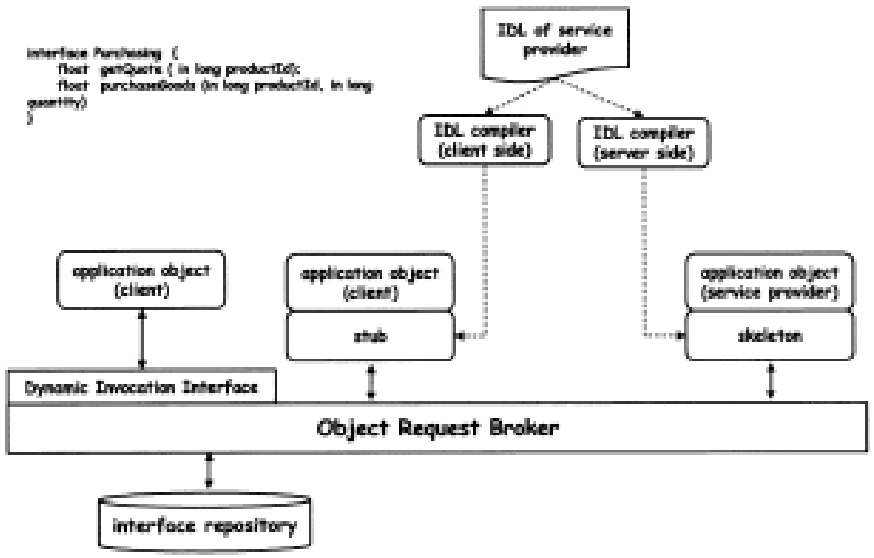
```
<definitions name="Procurement“ targetNamespace="http://example.com/procurement/definitions"
      xmlns:tns="http://example.com/procurement/definitions"          xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"              xmlns="http://schemas.xmlsoap.org/wsdl/" >

<message name="OrderMsg">
       <part name="productName" type="xs:string"/>
       <part name="quantity" type="xs:integer"/>
</message>

<portType name="procurementPortType">
       <operation name="orderGoods">
              <input message = "OrderMsg"/>
       </operation>
</portType>

<binding name="ProcurementSoapBinding" type="tns:procurementPortType">
       <soap:binding style="document“ transport="http://schemas.xmlsoap.org/soap/http"/>
       <operation name="orderGoods">
              <soap:operation soapAction="http://example.com/orderGoods"/>
              <input> <soap:body use="literal"/> </input>
       </operation>
</binding>

<service name="ProcurementService">
       <port name="ProcurementPort" binding="tns:ProcurementSoapBinding">
       <soap:address location="http://example.com/procurement"/>
       </port>
</service>

</definitions>
```

RPC Style Example

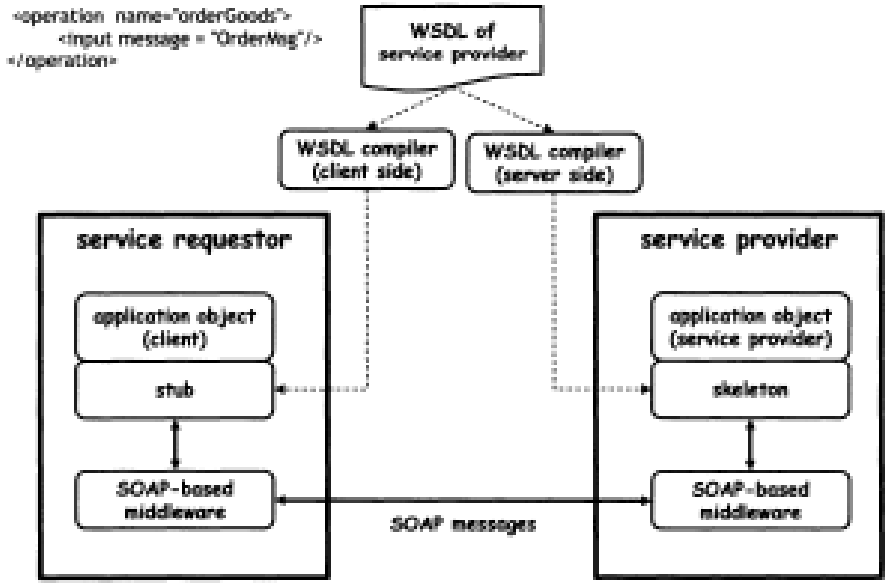
```
<binding name="TemperatureBinding" type="tns:TemperaturePortType">
<soap:binding style="rpc“ transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="getTemp">
        <soap:operation soapAction="" />
        <input>
            <soap:body use="encoded" namespace="urn:xmethods-Temperature“
            encodingStyle=http://schemas.xmlsoap.org/soap/encoding/ />
        </input>
        <output>
            <soap:body use="encoded" namespace="urn:xmethods-
            Temperature“ encodingStyle=http://schemas.xmlsoap.org/soap/encoding/ />
        </output>
    </operation>
</binding>

<service name="TemperatureService">
            <documentation>Returns current temperature in a given U.S. zipcode</documentation>
            <port name="TemperaturePort“ binding="tns:TemperatureBinding">
                        <soap:address location="http://services.xmethods.net:80/soap/servlet/rpc router" />
            </port>
</service>
```

Comparison between IDL and WSDL

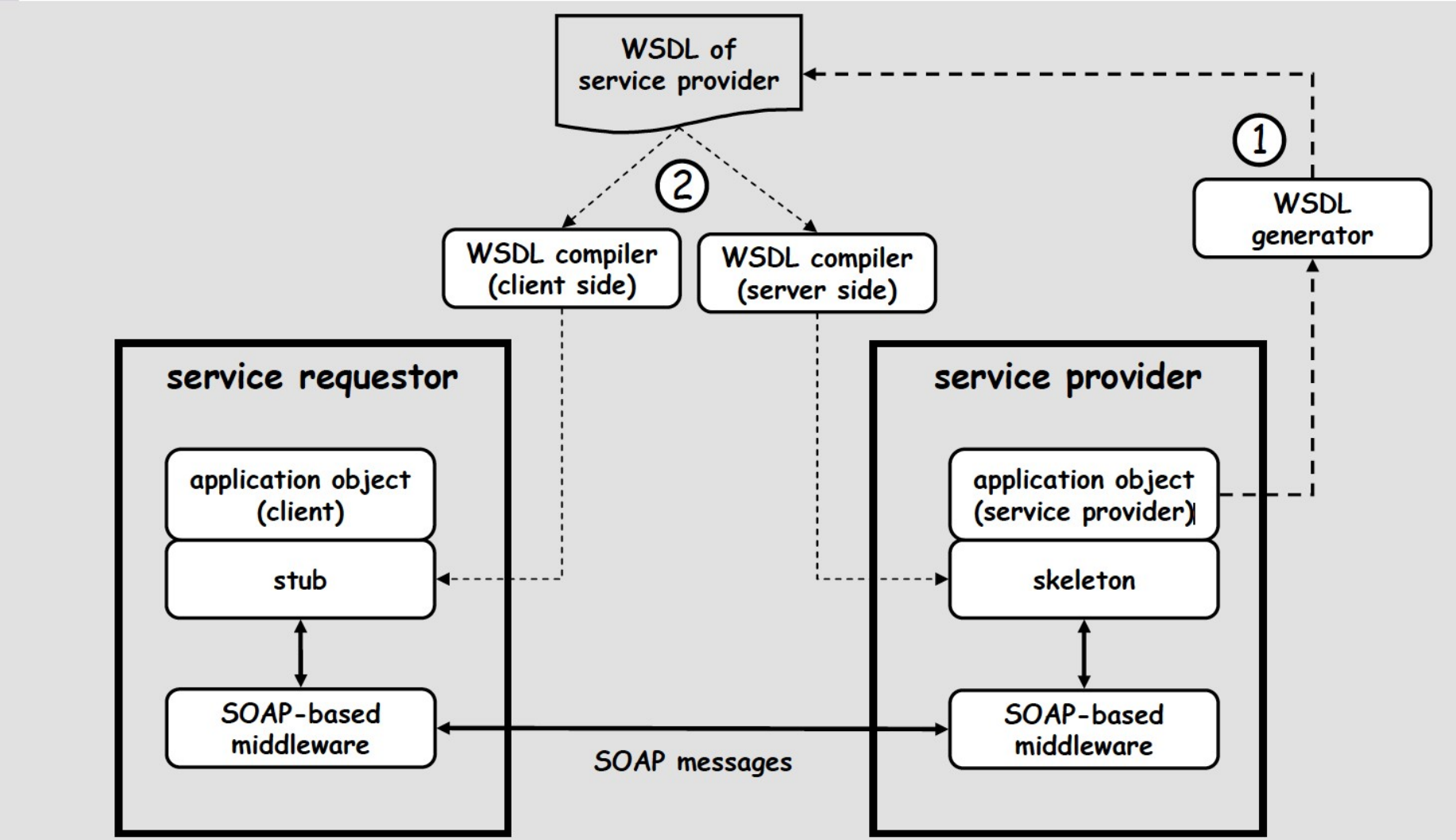
a) IDL specifications compiled into client-side stubs and server-side code skeletons. Distribution managed by ORB middleware



b) WSDL performs a similar function to IDL however without a common middleware platform.



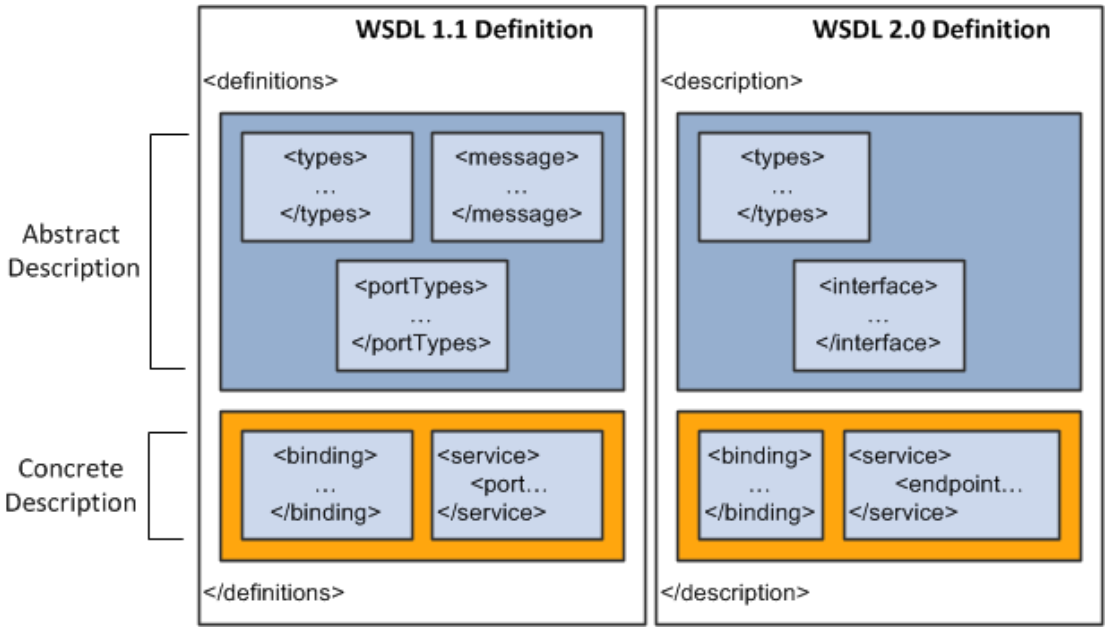
WSDL and Code Generation



Source: Alonso et. al. (2004) Copyright Springer Verlag Berlin Heidelberg

WSDL 1.1 and WSDL 2.0

- WSDL 1.2 renamed WSDL 2.0 because of substantial differences
- Adding further semantics to the description language.
- Removal of message constructs. These are specified using the XML schema type system in the types element.
- PortTypes renamed *interfaces* and Ports renamed *endpoints*.



WSDL 1.1	WSDL 2.0
Types	Types
Message	Removed
PortType	Interface (includes fault)
Operation	Operation
Binding	Binding
Port	Endpoint
Service	Service

Examples of WSDL

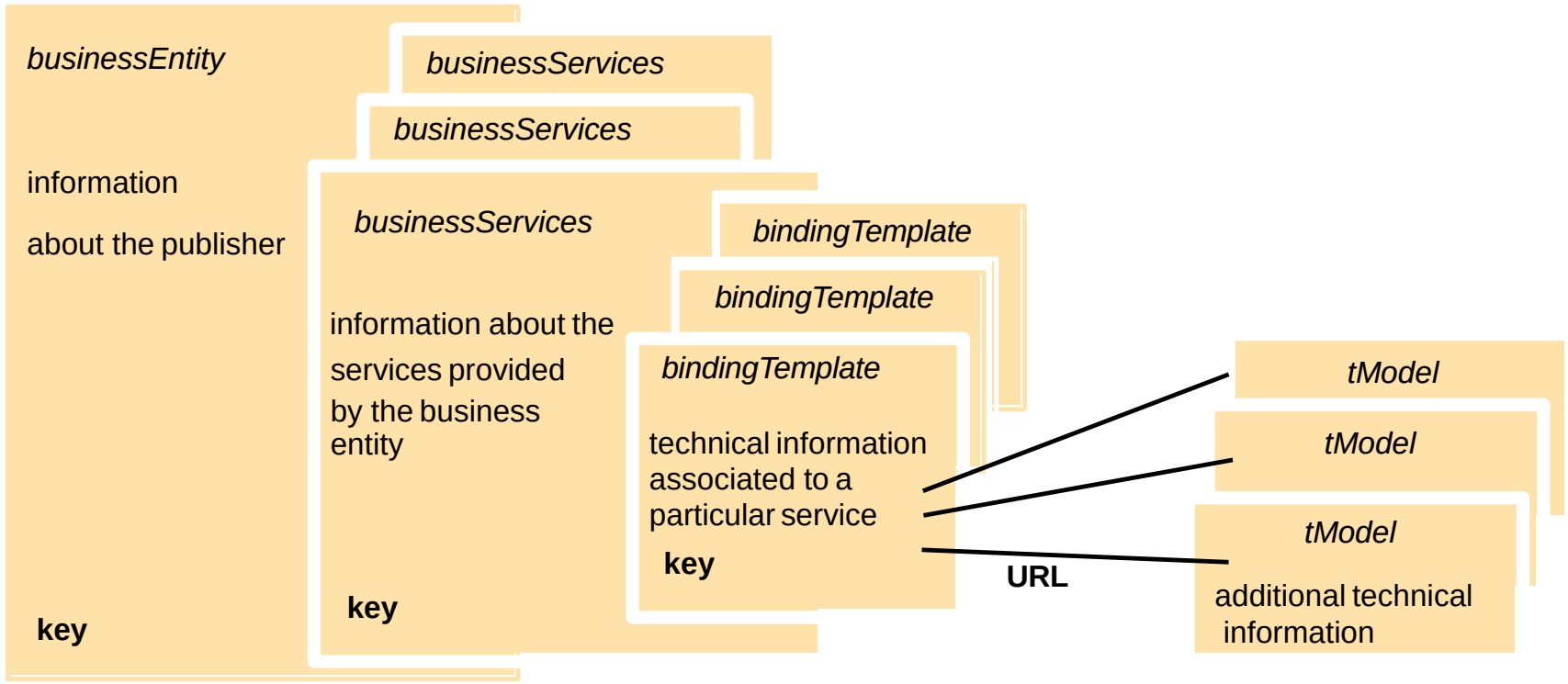
- http://opendap.co-ops.nos.noaa.gov/axis/

Universal Description, Discovery and Integration (UDDI)

UDDI (Universal Description, Discovery and Integration)

- A directory service where businesses can register and search for Web services.
- Originally, conceived as an “Universal Business Registry”
- Aims to find the service one actually wants among a potentially large collection of services and servers.
- The goal is that the client does not necessarily need to know where the server resides or even which server provides the service (pointers to WSDL).

The Main UDDI Data Structures



Adapted from (Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4)

UDDI

An entry in an UDDI registry is an XML document composed of different elements:

- **businessEntity**: a description of the organization that provides the service.
- **businessService**: a list of all Web services offered by the business entity.
- **bindingTemplate**: the technical aspects of the service being offered and the location of services

  e.g. <accesssPoint URLType="http">
  http://www.getquote.com/singlestockquote
  </accessPoint>

- **tModel**: ("technical model"): a generic element that can be used to store additional information about the service, including a reference to the WSDL. Also used for classification and categorization.

  e.g. <overviewURL>
  http://localhost/helloworld.wsdl
  </overviewURL>

UDDI

Together, these elements are used to provide:

- **white pages** information: Listing of organisations, contact information (phone, email addresses) and of the services these organisations provide
- **yellow pages** information: Classifications of both companies and Web services according to taxonomies that can be either standardised or user-defined
- **green pages** information: Information describing how a given Web service can be invoked (provided by means of pointers to service description documents)

Code Example

```
<businessEntity businessKey="ba744ed0-3aaf-11d5-80dc-002035229c64"
operator="www.ibm.com/services/uddi" authorizedName="0100001QS1"> <description
xml:lang="en">Web services resource  site</description>
 <contacts>
  <contact useType="Founder">
    <personName>Tony Hong</personName>
    <email useType="Founder">thong@xmethods.net</email>
  </contact>
 </contacts>
 <businessServices>
  <businessService serviceKey="d5921160-3e16-11d5-98bf-002035229c64"
        businessKey="ba744ed0-3aaf-11d5-80dc-002035229c64">
    <name>XMethods Delayed Stock Quotes</name>
    <description xml:lang="en">20-minute delayed stock quotes</description>
    <bindingTemplates>
    <bindingTemplate
        bindingKey="d594a970-3e16-11d5-98bf-002035229c64"
        serviceKey="d5921160-3e16-11d5-98bf-002035229c64">
      <description xml:lang="en">SOAP binding for delayed stock quotes service</description>
      <accessPoint URLType="http">http://services.xmethods.net:80/soap</accessPoint>
      <tModelInstanceDetails>
        <tModelInstanceInfo tModelKey="uuid:0e727db0-3e14-11d5-98bf-002035229c64”/>
      </tModelInstanceDetails>
      </bindingTemplate>
    </bindingTemplates>
  </businessService>
 </businessServices>
 <businessEntity>
```

from: Anders Møller and Michael Schwartzbach (2006), An Introduction to XMLand Web Technologies

Code Example

```
<tModel tModelKey="uuid:0e727db0-3e14-11d5-98bf-002035229c64"
        operator="www.ibm.com/services/uddi"
        authorizedName="0100001QS1">
         <name>XMethods Simple Stock Quote</name>
         <description xml:lang="en">Simple stock quote interface</description>
         <overviewDoc>
           <description xml:lang="en">wsdl link</description>
           <overviewURL>http://www.xmethods.net/tmodels/SimpleStockQuote.wsdl</overviewURL>
         </overviewDoc>
         <categoryBag>
           <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
                    keyName="uddi-org:types"
                    keyValue="wsdlSpec" />
         </categoryBag>
</tModel>
```

An example of a SOAP message sent to a UDDI registry to inquire about services named "delayed stock quotes″

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <find_service businessKey="..." generic="1.0" xmlns="urn:uddi-org:api">
          <categoryBag>
          <keyedReference tModelKey="UUID:DB77450D-9FA8-45D4-A7BC-04411D14E384"
                    keyName="Stock market trading services"
                    keyValue="84121801"/>
          </categoryBag>
    </find_service>
  </Body>

</Envelope>
```

from: Anders Møller and Michael Schwartzbach (2006), An Introduction to XMLand Web Technologies

UDDI: Why didn't work?

- The problem with UDDI is the initial ambitious goals:
  - The "Google" of web services (and standardized)
  - Automatically find business partners worldwide
  - Find the interface and build the application on the fly
- In reality:
  - Nobody does business with partners found at random in the Internet
  - Contract and SLAs are more than syntax
  - Trust and knowing the partner are very important in practice
  - The interface describes the syntax, conversations and more complex functions are not yet sufficiently standardized