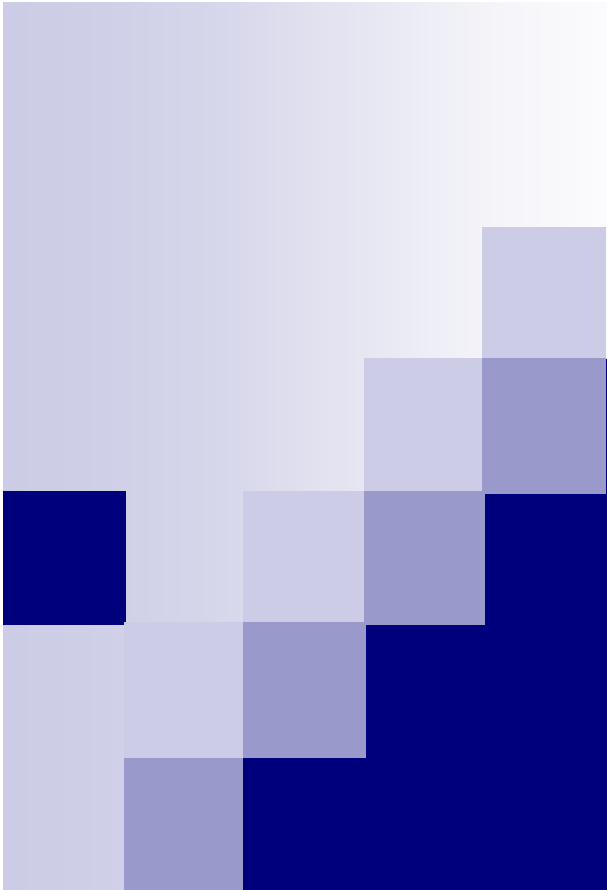




Topic 3

WSDL and WADL and UDDI

Assist. Prof. Thanisa Kruawaisayawan

Assoc.Prof. Dr. Thanachart Numnonda

Mini Master of Java Technology

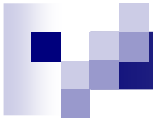
King Mongkut's Institute of Technology Ladkrabang

August 2010

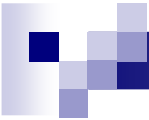


Agenda

- What is and Why WSDL?
- WSDL Elements
- WSDL Transmission Patterns
- WADL Basics and Elements
- UDDI Basics and Data Types

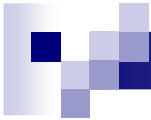


What is and why WSDL?



What is WSDL?

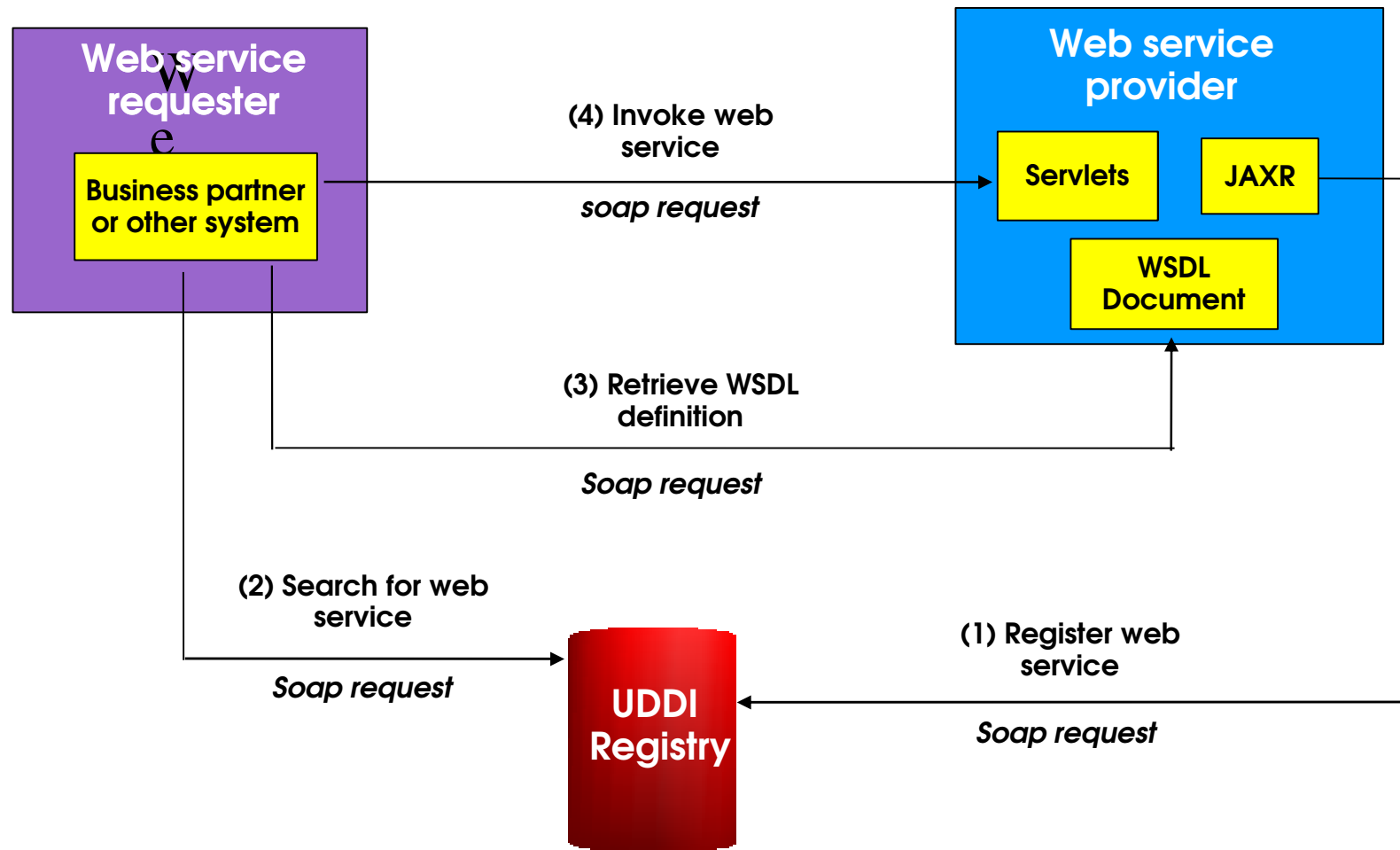
- XML language for **describing** web services
- Web service is described as
 - A set of **communication endpoints (ports)**
- Endpoint is made of two parts
 - **Abstract definitions of operations and messages**
 - **Concrete binding** to networking protocol (and corresponding endpoint address) and message encoding
- Why this separation?
 - Enhance **reusability** (of the abstract part, for example)



WSDL Service Description

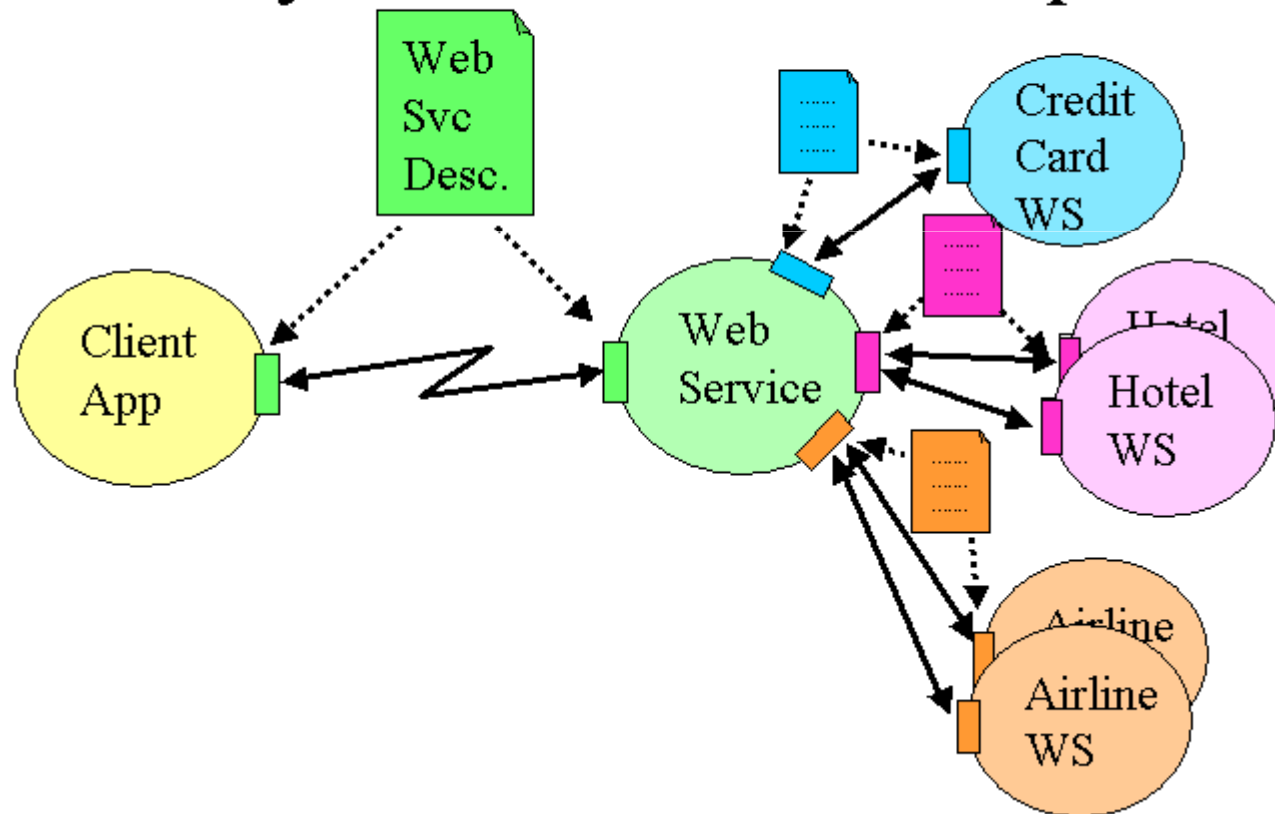
- WSDL is “the interface for Web Services” describing:
- **What** a service does - the operations (methods) the service provides, and the data (arguments and returns) needed to invoke them.
- **How** a service is accessed - details about data formats and protocols necessary to access the service operations.
- **Where** a service is located - details of the protocol-specific network address, such as a URL.

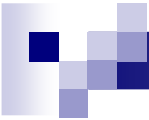
Where is WSDL Used?



Why WSDL?

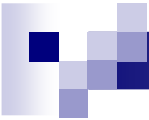
Many Web Service Descriptions





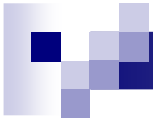
Why WSDL? (cont.)

- Enables **automation** of communication details between communicating partners
 - Machines can read WSDL
 - Machines can invoke a service defined in WSDL
- Discoverable through registry
- Arbitration
 - 3rd party can verify if communication conforms to WSDL



WSDL Document Structure

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"
    targetNamespace="your namespace here"
    xmlns:tns="your namespace here"
    xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap">
  <wsdl:types>
    <xs:schema targetNamespace="your namespace here (could be another) "
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      <!-- Define types and possibly elements here -->
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="some operation input">
    <!-- part(s) here -->
  </wsdl:message>
  <wsdl:message name="some operation output">
    <!-- part(s) here -->
  </wsdl:message>
  <wsdl:portType name="your type name">
    <!-- define operations here in terms of their messages -->
  </wsdl:portType>
  <wsdl:binding name="your binding name" type="tns:port type name above">
    <!-- define style and transport in general and use per operation -->
  </wsdl:binding>
  <wsdl:service>
    <!-- define a port using the above binding and a URL -->
  </wsdl:service>
</wsdl:definitions>
```



WSDL Elements

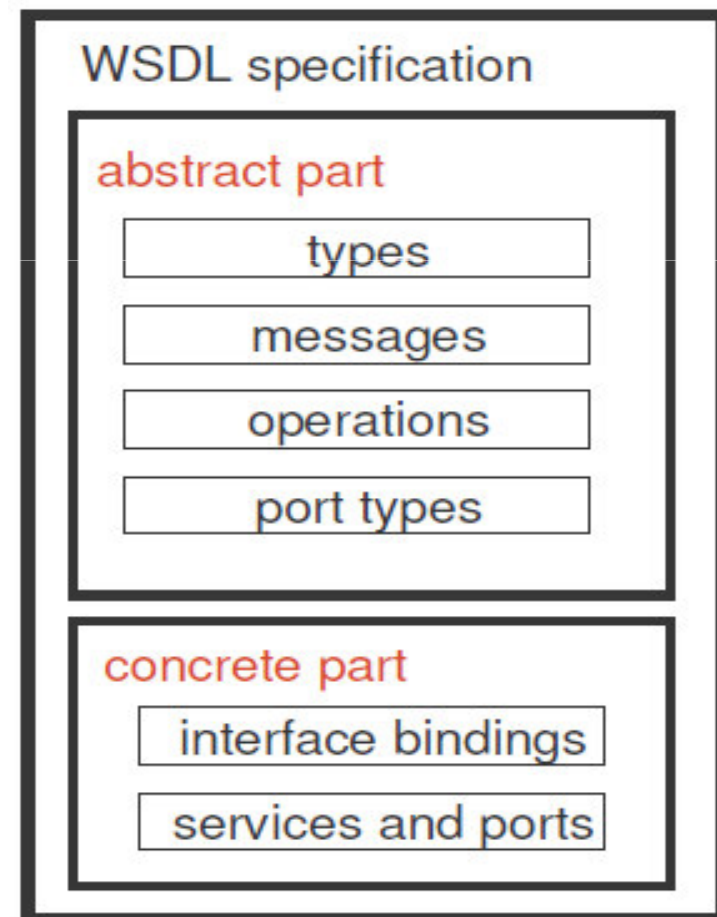
WSDL Structure

■ Abstract part

- Types
- Message
- Operation
- Port Type

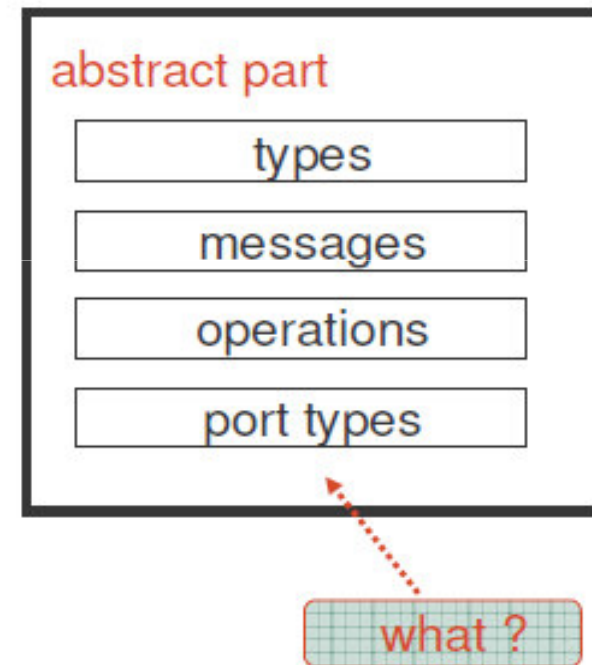
■ Concrete part

- Binding
- Port
- Service



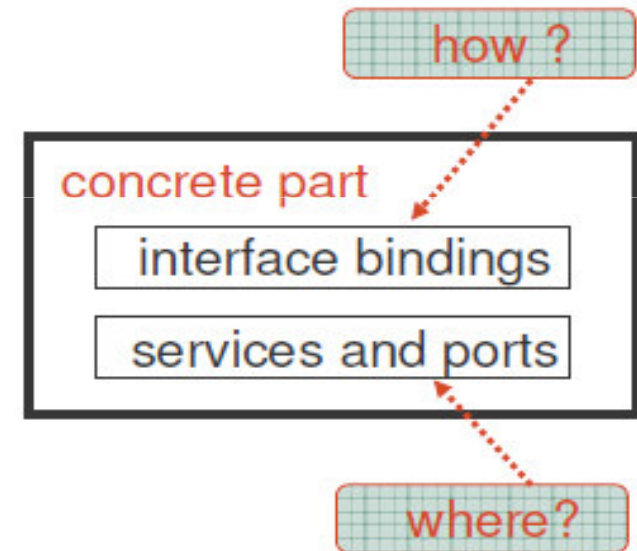
WSDL Structure - Abstract

- **port type** - logical collection of related operations
- **operation** - abstract description of an action supported by the service
- **message** - data exchanged in a single logical transmission
- **types** - data structures that will be exchanged as parts of messages

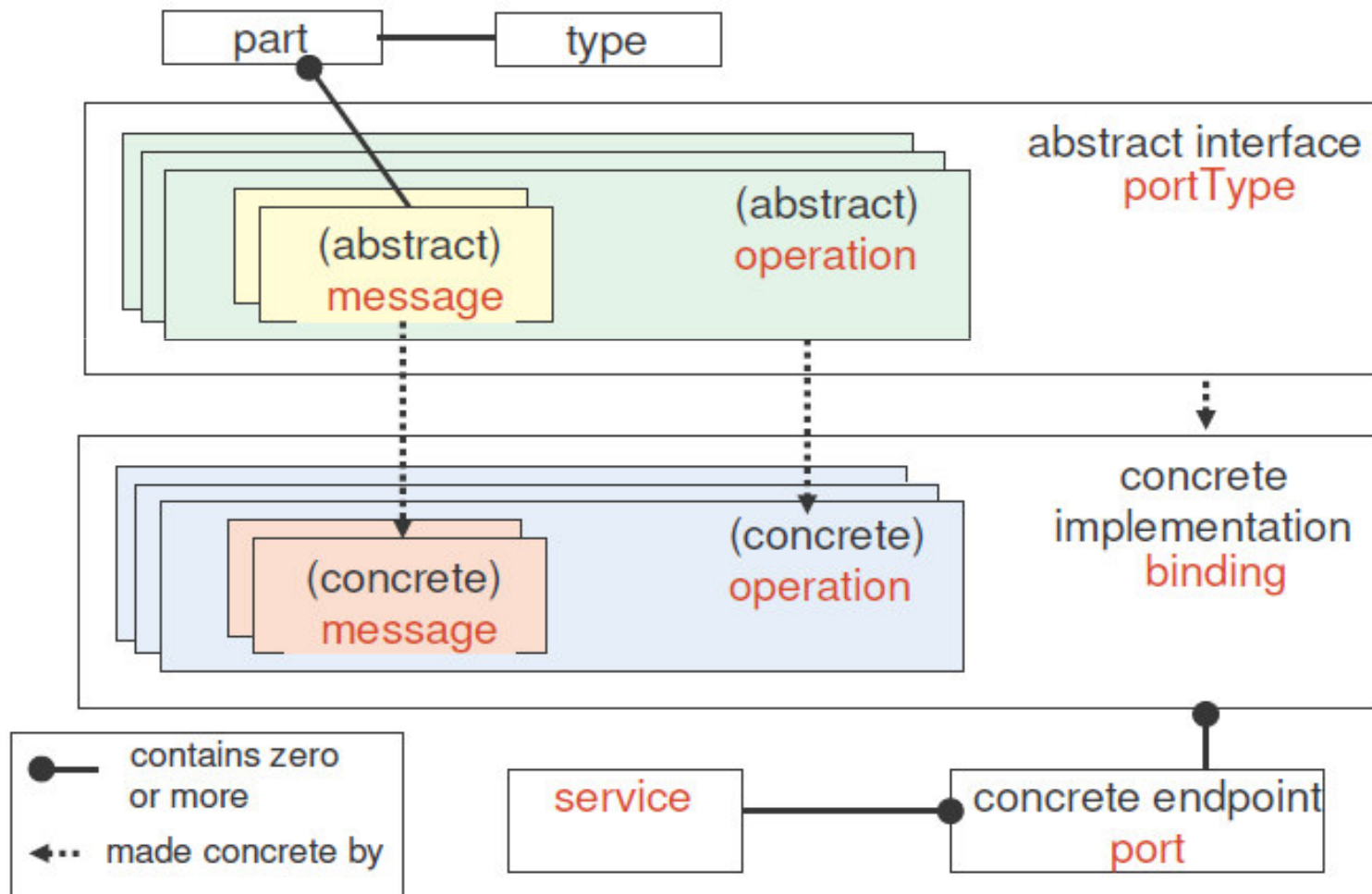


WSDL Structure - Concrete

- **interface bindings** - message encoding and protocol binding for all operations and messages defined in a given porttype
- **ports** - combine the interface binding information with a network address specified by a URI
- **services** - are logical groupings of ports



WSDL Information Model



WSDL : Example

```
<?xml version="1.0"?>
<definitions
  name= "Orders"
  targetNamespace="http://www.example.com/orders"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  xmlns:tns=http://www.example.com/orders">
```

```
<message name="OrderMsg">
  <part name="productId" type="xsd:string"/>
  <part name="quantity" type="xsd:integer"/>
</message>
```

```
<portType name="OrderPortType">
  <operation name="orderProductRequest">
    <input message = "OrderMsg"/>
  </operation>
</portType>
```

abstract part

messages

operation and
port type

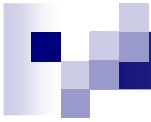
WSDL : Example (cont.)





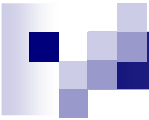
WSDL Elements : Definitions

- *name* attribute - corresponds to the name of the web service. It is only for documentation and is optional
- *targetNamespace* attribute - a URI for the entire WSDL file
- default namespace - all elements without a namespace prefix, such as *message* or *portType*, are assumed to be part of the default WSDL namespace: *http://schemas.xmlsoap.org/wsdl/*
- other XML namespace declarations



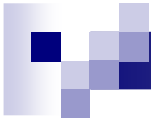
WSDL Elements : Type

- Data type definitions
- Used to describe exchanged messages
- Uses W3C XML Schema as canonical type system



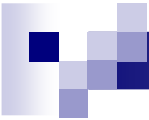
WSDL Example: Types

```
<definitions name="StockQuote"
  targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>
```



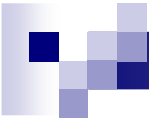
WSDL Elements : Message

- A message describes the abstract form of an input, output or a fault message.
- A message describes the data being communicated.
- Each message has a unique name within the WSDL document and contains a collection of **parts**.
- A **message** may have several **parts**.
- A **part** may belong to several **messages**.



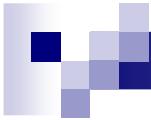
WSDL Elements : Part

- Parts provide a flexible mechanism for describing the logical content of messages.
- A **part** element has two properties:
 - name : represented by the **name** attribute, which must be unique among all the **part** elements of the **message** element
 - kind : defined as either a **type** or an **element** attribute:
 - element - the payload of the message on the wire is precisely the XML element
 - type - any element conforming to the type



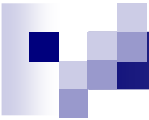
WSDL Elements : PortType

- portType is a collection of one or more related operations describing the interface of a web service.
- portType definition is a collection of **operation** elements.
- Generally, WSDL documents contain only one portType element, because different web service interface definitions are written with different documents.
- portType has a single **name** attribute.
- The name of portType together with the namespace of the WSDL document define a unique name for the portType.



WSDL Elements : Operation

- operation defines a method of a web service, including the name of the method, input parameters, and the output or return type of the method.
- All operations in a portType must have different names.
- Each operation may define:
 - input message
 - output message
 - fault message
- An operation in WSDL is the equivalent of a method signature in Java.



Abstract Elements : Example

```
<message name="GetLastTradePriceInput">  
  <part name="body" element="xsd1:TradePriceRequest"/>  
</message>
```

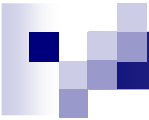
```
<message name="GetLastTradePriceOutput">  
  <part name="body" element="xsd1:TradePrice"/>  
</message>
```

```
<portType name="StockQuotePortType">  
  <operation name="GetLastTradePrice">  
    <input message="tns:GetLastTradePriceInput"/>  
    <output message="tns:GetLastTradePriceOutput"/>  
  </operation>  
  <!-- More operations -->  
</portType>
```



WSDL Elements : Binding

- The binding element specifies how to format messages in a protocol specific manner:
 - message encoding
 - protocol binding
- Each portType can have several binding elements associated with it.
- Each binding specifies how to invoke operations using particular transport protocols. For instance: SOAP over HTTP, SOAP over SMTP, etc.



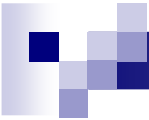
WSDL Elements : Binding (cont.)

- The binding element has two attributes:
 - name : must be unique among all binding elements defined in the WSDL document
 - type : identifies which portType the binding describes



WSDL Elements : Binding (cont.)

- Defines **protocol details** and **message format** for *operations* and *messages* defined by a particular *portType*
- *Specify one protocol out of*
 - *SOAP (SOAP over HTTP, SOAP over SMTP)*
 - *HTTP GET/POST*
- *Provides **extensibility** mechanism*
 - *Can includes **binding extensibility elements***
 - *Binding extensibility elements are used to specify the concrete grammar*



RPC and Document-style

RPC

- Procedure call
- Method signature
- Marshaling
- Tightly-coupled
- Point to point
- Synchronous
- Typically within Intranet

Document-style

- Business documents
- Schema
- Parsing & Validating
- Loosely coupled
- End to end
- Asynchronous
- Typically over internet



RPC and Document-style (cont.)

RPC

- Within Enterprise
- Simple, point-to-point
- Short running business process
- Reliable and high bandwidth
- Trusted environment

Document-style

- Between enterprise and enterprise
- Complex, end to end with intermediaries
- Long running business process
- Unpredictable bandwidth
- Blind trust



Binding Protocol Encoding Rules

- The binding also specifies the encoding rules used in serializing parts of a message into XML:
 - **literal encoding**: takes the WSDL types defined in XML Schema and “literally” uses those definitions to represent the XML content of messages. Abstract WSDL types become concrete types
 - **SOAP encoding**: considers the XML Schema definitions as abstract entities and translates them into XML using SOAP encoding rules
- Literal encoding is used for document style interactions.
- SOAP encoding is used for RPC style interactions.



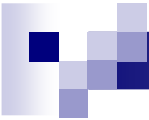
WSDL Elements : Port

- Port specifies the network address of the end-point hosting the web service.
- port is a single end-point defined as a combination of a binding and a network address.
- There can be many ports for a binding, just like many implementations for the same interface.
- The **soap:address** element is used to give a port an address.



WSDL Elements : Service

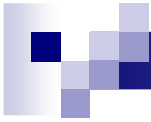
- A **service** is a collection of ports.
- Although a WSDL document can contain a collection of service elements, by convention a WSDL document contains a single service.
- Usage: group the ports that are related to the same service interface (**portType**) but expressed by different protocols (**binding**).



Concrete Elements : Example

```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
    <soap:operation
      soapAction="http://example.com/GetLastTradePrice"/>
    <input> <soap:body use="literal" />
    </input>
    <output> <soap:body use="literal" />
    </output>
  </operation>
</binding>

<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>
```



WSDL Transmission Patterns



Transmission Patterns in WSDL

■ One-way

- The endpoint receives a message

■ Request/response

- The endpoint receives a message, and sends a correlated message

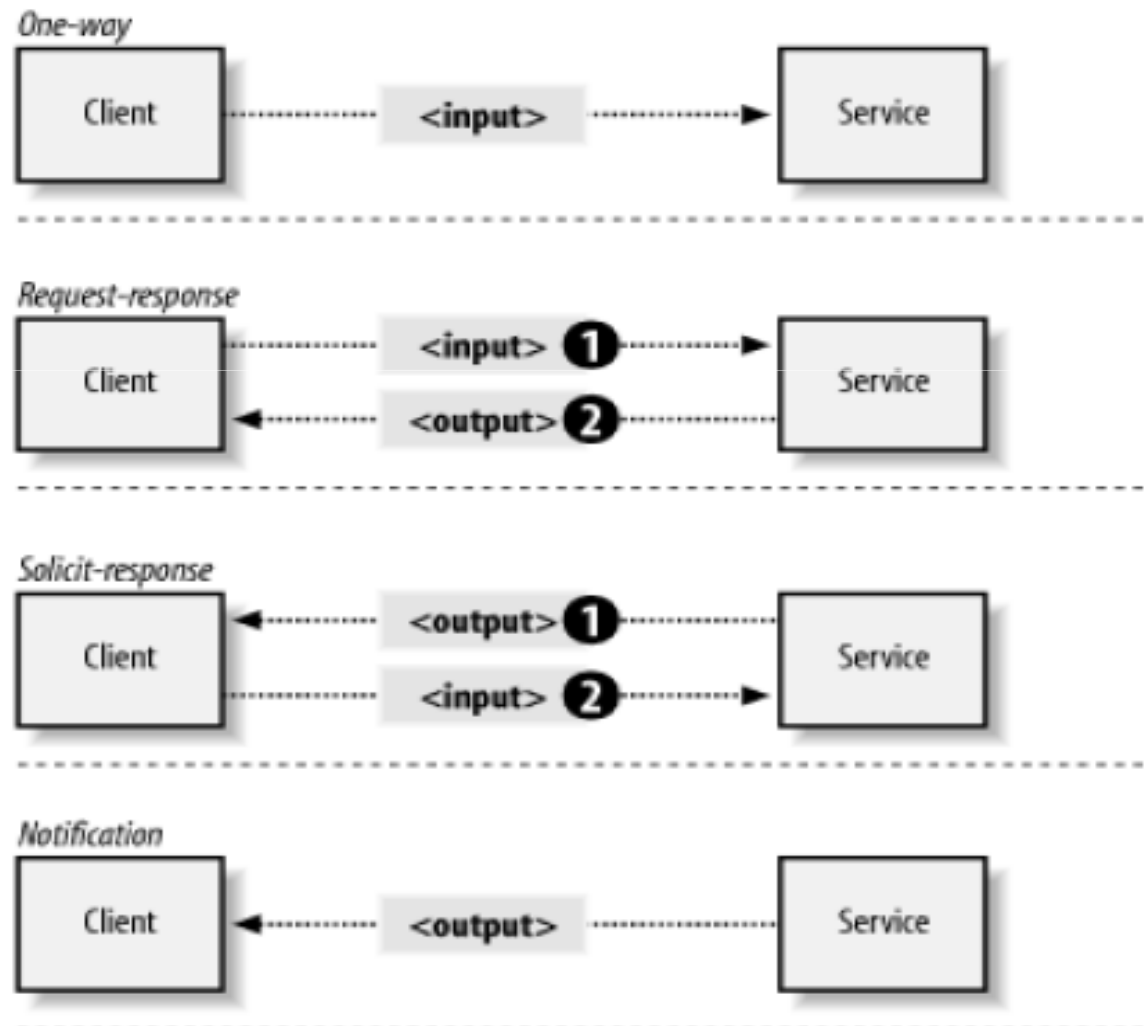
■ Notification

- The endpoint sends a message

■ Solicit/response

- The endpoint sends a message, and receives a correlated message

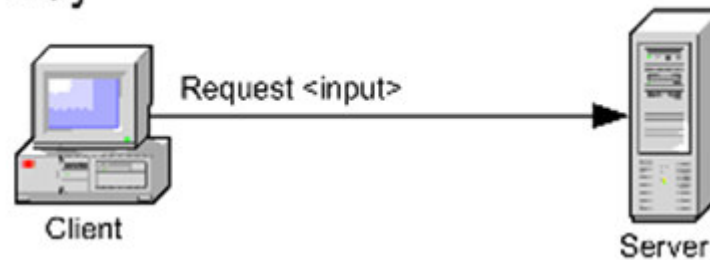
Transmission Patterns in WSDL



One-way Operation : Example

```
<operation name="submitPurchase">  
  <input message="purchase"/>  
</operation>
```

One Way

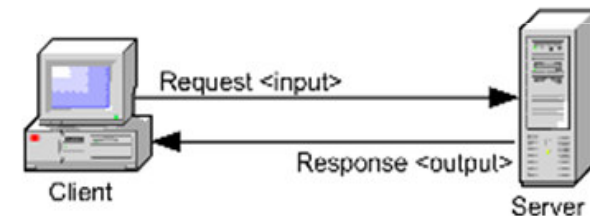


Request/Response Operation : Example

```
<operation name="submitPurchase">  
  <input message="purchase"/>  
  <output message="confirmation"/>  
</operation>
```

```
<operation name="submitPurchase">  
  <input message="purchase"/>  
  <output message="confirmation"/>  
  <fault message="faultMessage"/>  
</operation>
```

Request-Response



Notification Operation : Example

<operation name="deliveryStatus">

 <output message="trackingInformation"/>

</operation>

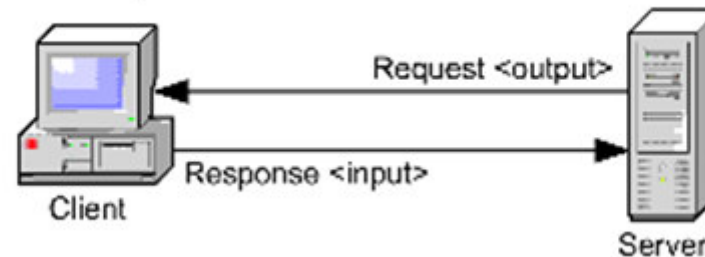
Notification

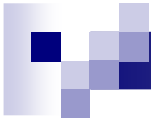


Solicit/Response Operation : Example

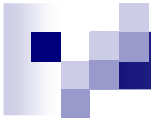
```
<operation name="clientQuery">  
  <output message="bandwidthRequest"/>  
  <input message="bandwidthInfo"/>  
  <fault message="faultMessage"/>  
</operation>
```

Solicit-Response



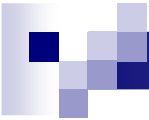


WADL Basic and Elements



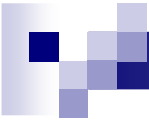
WADL

- Web Application Description Language
- An XML-based file format
- A machine-readable description of HTTP-based REST web Services
- Development language+platform neutral



WADL Elements

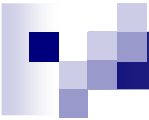
- **Grammars**
 - Currently specify use of W3C XML Schema or RelaxNG
- **Resources**
 - Identified by a URI template
 - Specify which methods are supported
- **Method**
 - Specify details of request and response contents
 - Often refer to representations
- **Representation**
 - Describe the format of a HTTP entity
 - Can refer to grammars



WADL Document Structure

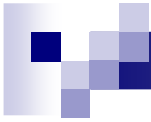
```
<application>
  <doc/>*
  <grammars/>?
  <resources base='anyURI'>?
    <doc/>*
    <resource path='template' type='anyURI+'?>+
      <doc/>*
      <param/>*
      ( <method/> | <resource/> )+
    </resource>
  </resources>
  ( <method/> | <representation/> | <fault/> |
    <resource_type/>)*
</application>
```

* => 0 or more ? => 0 or 1 + => 1 or more



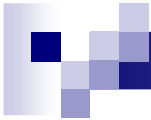
WADL Method Structure

```
<method name='NMTOKEN'? id='ID'? href='anyURI'?>
  <doc/>*
  <request>?
    <param>*
    <representation/>*
  </request>
  <response>?
    ( <representation/> | <fault/> ) *
  </response>
</method>
```



Yahoo News Search

- <http://api.search.yahoo.com/NewsSearchService/V1/newsSearch>
- Query parameters
 - appid: get this from Yahoo by registering
 - query: space separated list of keywords
 - many others including language, sort, result count etc.
- Get back results as XML, JSON or PHP
 - XML schema available for normal and error responses



Yahoo News Search in WADL

```
<application xmlns:...>
```

```
  <grammars>
```

```
    <include href="../../../NewsSearchResponse.xsd"/>
```

```
    <include href="../../../NewsSearchError.xsd"/>
```

```
  </grammars>
```

```
  <resources
```

```
    base="http://api.search.yahoo.com/NewsSearchService/V1/">
```

```
      <resource path="newsSearch">
```

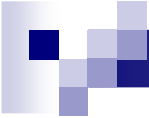
```
        <param name="appid" type="xsd:string"
```

```
          required="true" style="query"/>
```

```
        <method href="#search"/>
```

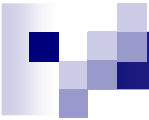
```
      </resource>
```

```
    </resources>
```



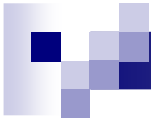
Yahoo News Search in WADL (cont.)

```
<method name="GET" id="search">
  <request>
    <param name="query" type="xsd:string"
      required="true" style="query"/>
    <param name="type" type="xsd:string"
      default="all" style="query">
      <option value="all"/>
      <option value="any"/>
      <option value="phrase"/>
    </param>
    ...
  </request>
  <response>
    <representation href="#resultSet"/>
    <fault href="#searchError"/>
  </response>
</method>
```



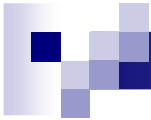
Yahoo News Search in WADL (cont.)

```
<representation id="resultSet"  
  mediaType="application/xml"  
  element="yn:ResultSet">  
  <doc xml:lang="en"  
    title="A matching list of news items"/>  
</representation>  
  
<fault id="searchError"  
  status="400"  
  mediaType="application/xml"  
  element="ya:Error"/>
```



wadl2java

- Open source project
 - <http://wadl.dev.java.net>
- Generates client-side stubs
- Command line or Apache Ant task
 - `java -jar wadl2java.jar`
- Uses JAXB for XML processing
- `file.wadl`



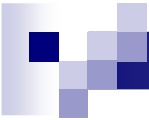
Yahoo News Search Stub

```
public class NewsSearch {  
    public NewsSearch() {...}  
    public ResultSet getAsResultSet(  
        String appid, String query) {...}  
    public DataSource getAsApplicationXml(  
        String appid, String query) {...}  
    public DataSource getAsApplicationJson(  
        String appid, String query) {...}  
    public DataSource getAsApplicationPhp(  
        String appid, String query) {...}  
    ...  
}
```



Mapping WADL to Java

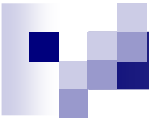
```
public class NewsSearch {  
    public NewsSearch() {...}  
    public ResultSet getAsResultSet(  
        String appid, String query) {...}  
}
```



Mapping WADL to Java (cont.)

```
public class NewsSearch {  
    public NewsSearch() {...}  
    public ResultSet getAsResultSet(  
        String appid, String query) {...}  
}
```

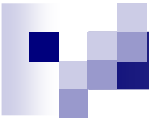
```
<resource path="newsSearch">  
    <param name="appid" style="query"/>  
    <method name="GET">  
        ...  
    </method>  
</resource>
```



Mapping WADL to Java (cont.)

```
public class NewsSearch {  
    public NewsSearch() {...}  
    public ResultSet getAsResultSet(  
        String appid, String query) {...}  
}
```

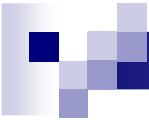
```
<resource path="newsSearch">  
    <param name="appid" style="query"/>  
    <method name="GET">  
        ...  
    </method>  
</resource>
```



Mapping WADL to Java (cont.)

```
public class NewsSearch {  
    public NewsSearch() {...}  
    public ResultSet getAsResultSet(  
        String appid, String query) {...}  
}
```

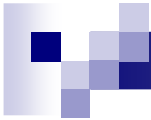
```
<resource path="newsSearch">  
    <param name="appid" style="query"/>  
    <method name="GET">  
        ...  
    </method>  
</resource>
```



Mapping WADL to Java (cont.)

```
public class NewsSearch {  
    public NewsSearch() {...}  
    public ResultSet getAsResultSet(  
        String appid, String query) {...}  
}
```

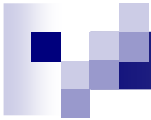
```
<method name="GET">  
    <request>  
        <param name="query" style="query"/>  
    </request>  
    <response>  
        <representation element="y:ResultSet"/>  
    </response>  
</method>
```



Mapping WADL to Java (cont.)

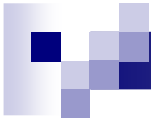
```
public class NewsSearch {  
    public NewsSearch() {...}  
    public ResultSet getAsResultSet(  
        String appid, String query) {...}  
}
```

```
<method name="GET">  
    <request>  
        <param name="query" style="query"/>  
    </request>  
    <response>  
        <representation element="y:ResultSet"/>  
    </response>  
</method>
```



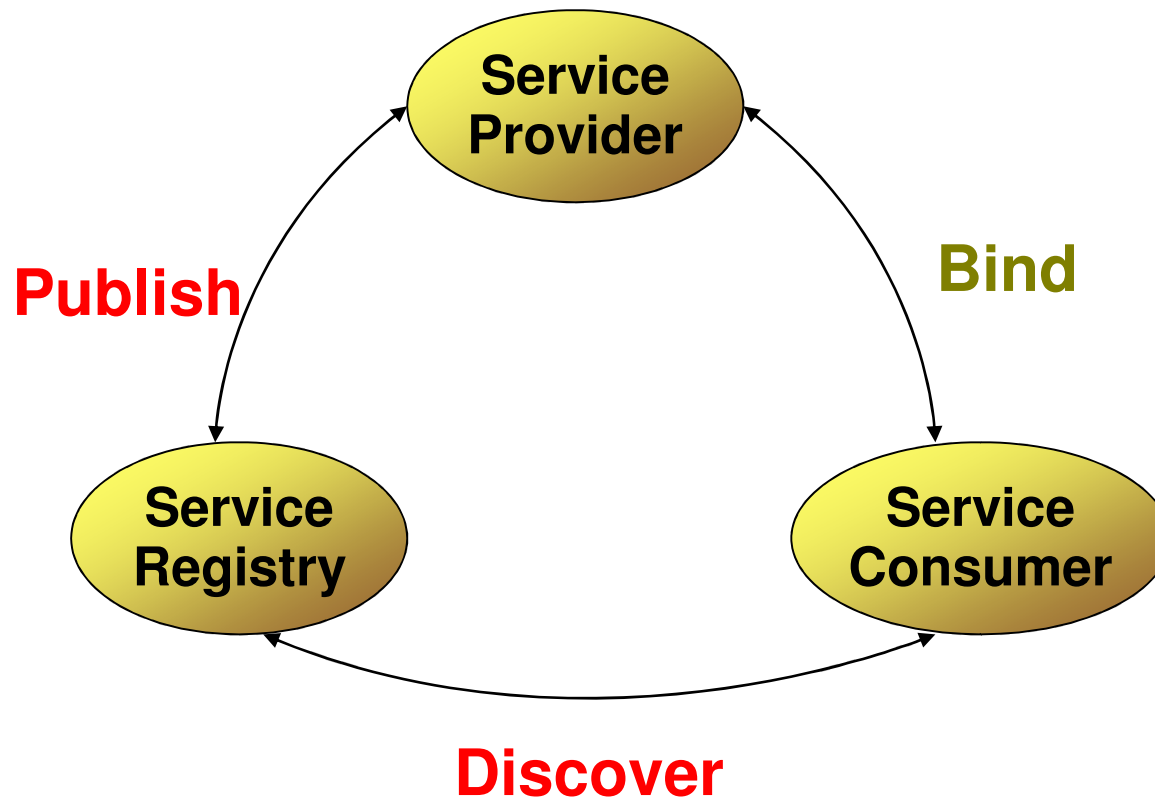
Client Code

```
NewsSearch s = new NewsSearch();  
ResultSet rs = s.getAsResultSet("some_app_id", "java");  
for (Result r: rs.getResultList()) {  
    System.out.printf("%s (%s)\n",  
        r.getTitle(),  
        r.getClickUrl());  
}
```



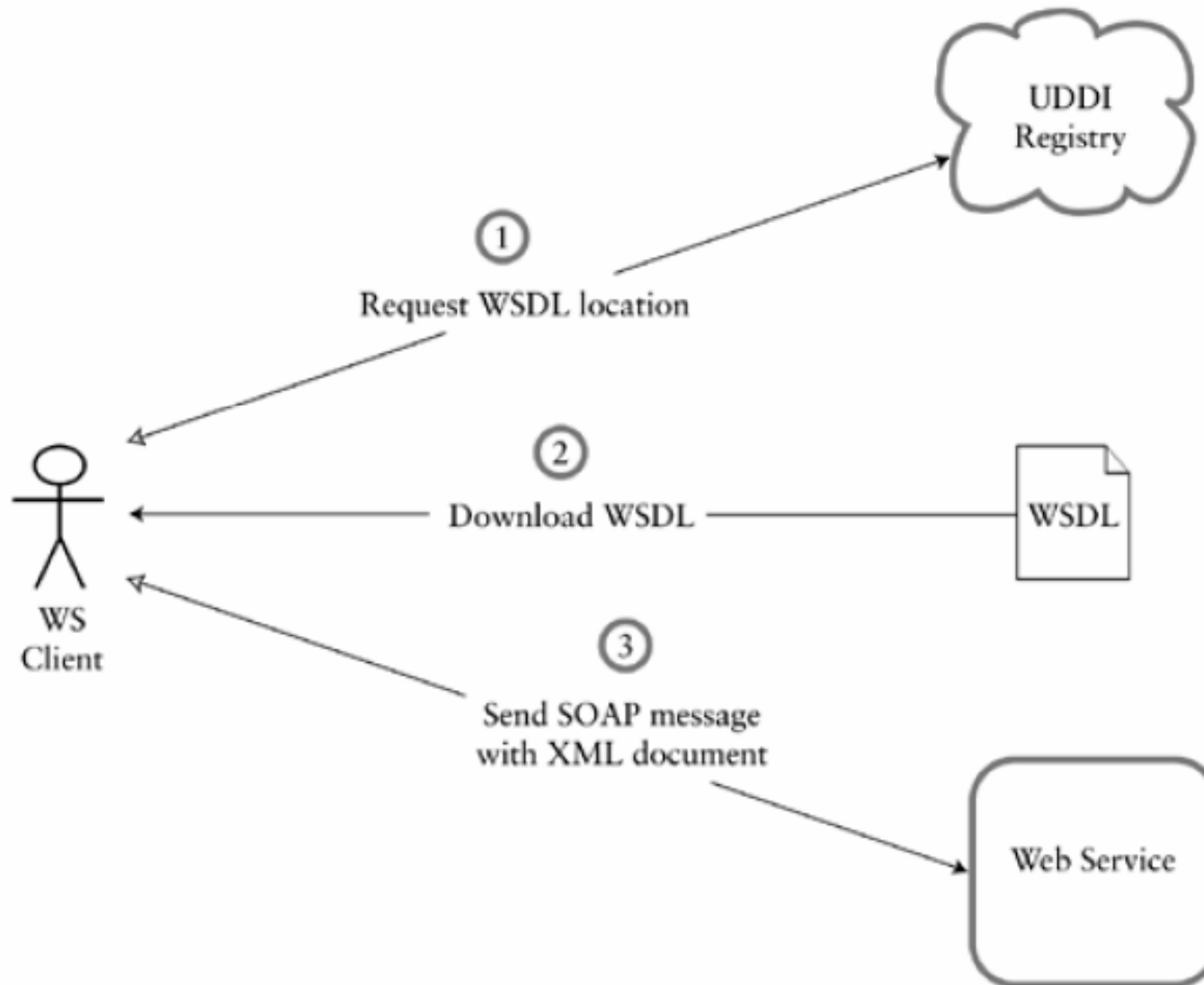
UDDI Basic and Data Types

Service Architecture

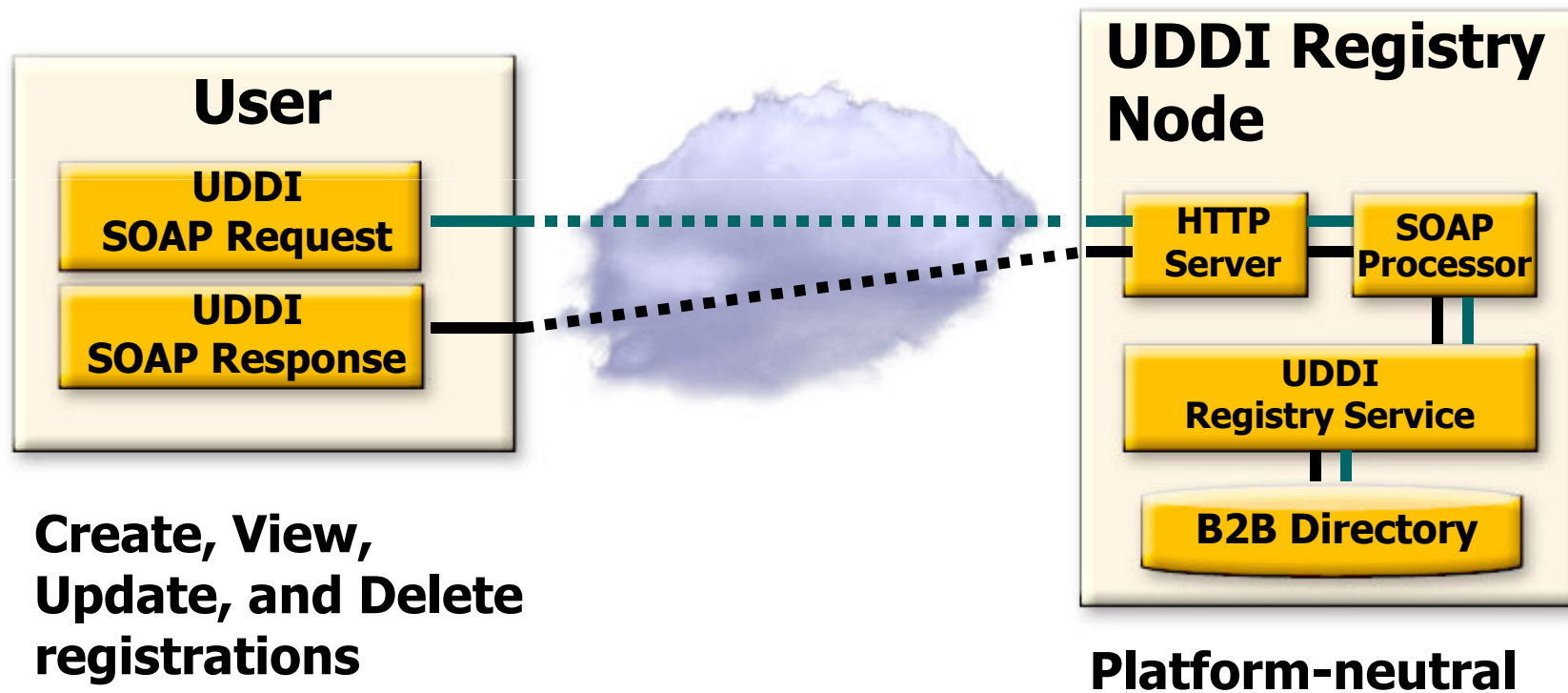


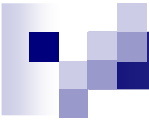
UDDI defines a scheme to **publish** and **discover** information about Web services.

WSDL & UDDI



UDDI Runs “Over” SOAP

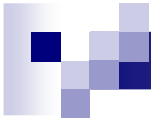




What is UDDI?

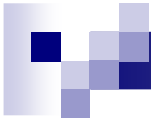
- Programmatic registration and discovery of business entities and their Web services
- Public UDDI registries

IBM, Microsoft, and SAP have shut down their public UDDI registries on January 12, 2006 after first announcement in 2000.
- Private UDDI registries within an intranet
(where we are today)



Business Registration Data

- “White pages”
 - address, contact, and known identifiers
- “Yellow pages”
 - industrial **categorizations**
 - Industry: NAICS (Industry codes - US Govt.)
 - Product/Services: UN/SPSC (ECMA)
 - Location: Geographical taxonomy
- “Green pages”
 - **technical information** about services



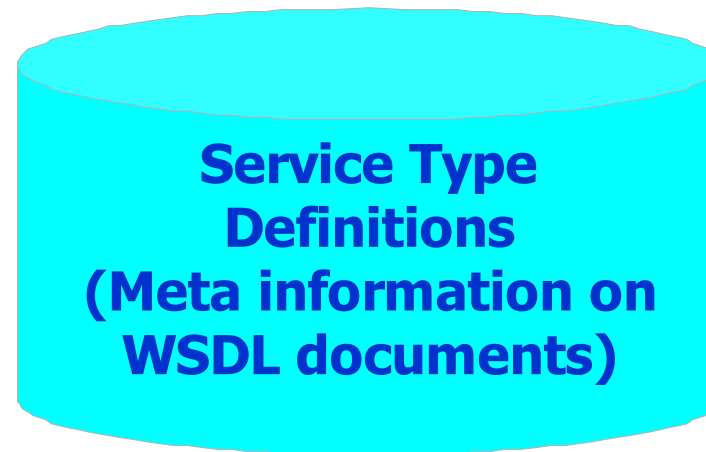
Registry Data

Created by businesses



**businessEntity's
businessService's
bindingTemplate's**

Created by standard organizations, industry consortium

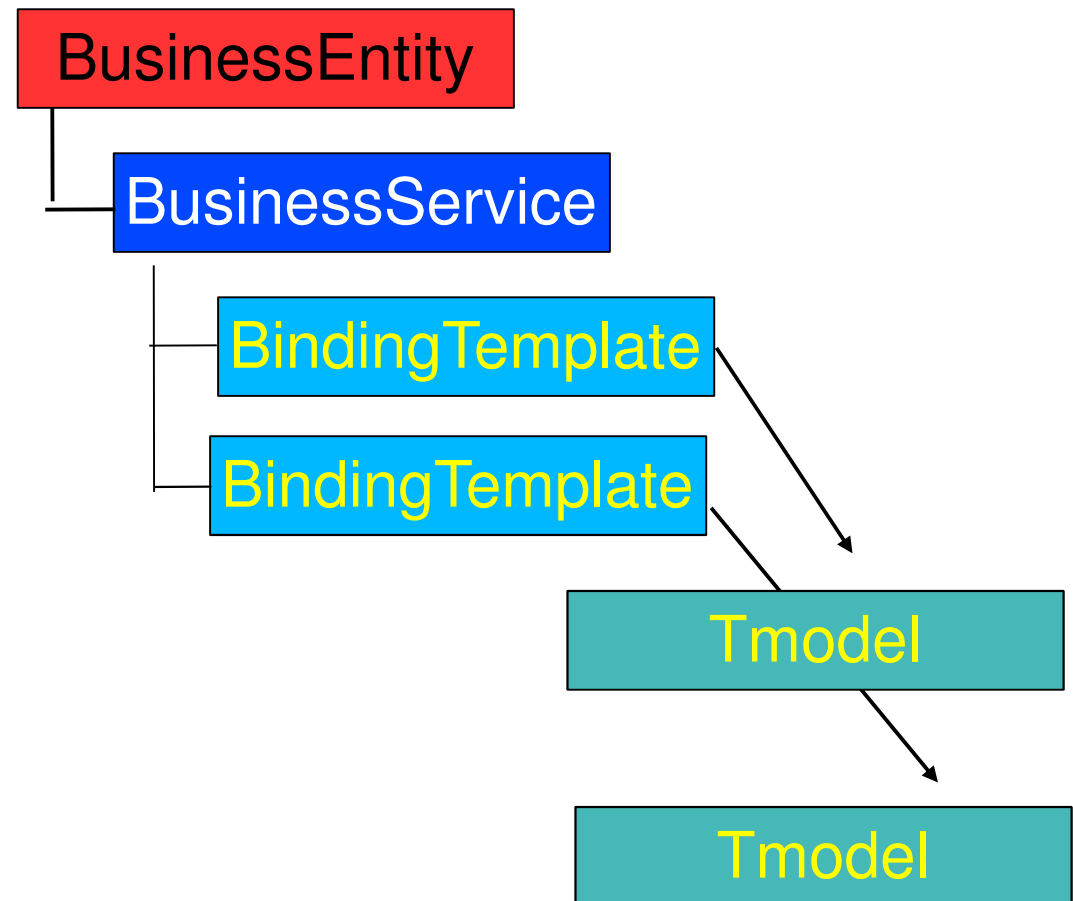


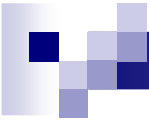
tModel's



UDDI Data Types

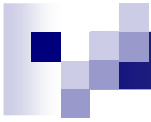
- **Business Entity**
 - White Pages information
- **Business Services**
 - Yellow Pages information
- **Binding Templates**
 - Green Pages information
 - Contains references to tModels
- **tModels**
 - Service Type Definitions
 - Contains references to WSDL documents





tModel Example

```
<tModel authorizedName="..." operator="..." tModelKey="...">
  <name>StockQuote Service</name>
  <description xml:lang="en">
    WSDL description of a standard stock quote service interface
  </description>
  <overviewDoc>
    <description xml:lang="en"> WSDL source document. </description>
    <overviewURL> http://stockquote-definitions/stq.wsdl </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="UUID:..."
      keyName="uddi-org:types"
      keyValue="wsdlSpec"/>
  </categoryBag>
</tModel>
```



Resources

- Some contents are borrowed from the presentation slides of Sang Shin, Java™ Technology Evangelist, Sun Microsystems, Inc.
- Some contents are borrowed from the presentation slides of Marc Hadley and Ayub Khan
- *Web Services and Java*, Elsa Estevez, Tomasz Janowski and Gabriel Oteniya, UNU-IIST, Macau



Thank you

thananum@gmail.com

twitter.com/thanachart

www.facebook.com/thanachart

www.thaijavadev.com

