ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στα Δίκτυα Υπηρεσιών

Διάλεξη 10η: SOAP - WSDL - UDDI

Χρήστος Νικολάου
Τμήμα Επιστήμης Υπολογιστών

Introduction to Service Networks

WSDL, SOAP and UDDI

Christos Nikolaou
Mariana Karmazi
Transformation Services Laboratory
CSD/UoC

XML Revisited - 1

In the mid 70's, Charles Gotfarb introduced the Standard Generalized Markup Language (SGML) for data representation.

Tim Berners-Lee conceived WWW and then founded the World Wide Web Consortium(W3C). One of the first initiatives: formal spec for Hypertext Markup Language (HTML), based on SGML.

HTML provides syntax for document text formatting and layout.

But then came e-business, and a need for standard business-centric data representation format. Result was the Extensible Markup Language (XML).

XML Revisited - 2

Say companies A and B want to do business over the Internet. A sends to B its financial report as an HTML document. We need some mechanism to look and **interpret** the document.

This is where XML comes in: supplements content with 'meta information', self descriptive labels for each piece of text.

Now company B can:

- Manipulate the report's data with a program
- Import report data into a database
- Store report in corporate document set
- Create different views of report – sort/filter data

XML Revisited -3

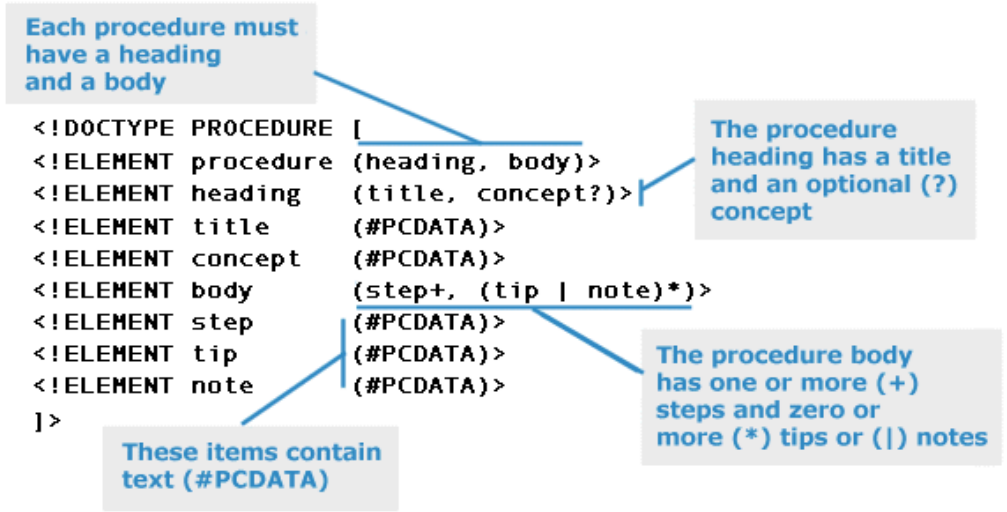
Like HTML, XML has **elements.** But unlike HTML, XML elements are not predefined. Set of related XML elements is a **vocabulary** (for example for invoices, purchase orders, etc.). Instance of a vocabulary is called an XML **document.** Organizations can agree on a standard set of vocabularies, or have dynamic transformation, see XSLT

XML Revisited - 4

<- Sample XML Document

```
<?xml version="1.0" encoding="utf-8" ?>
<CompanyInfo>
  <DepartmentInfo>
    <!-- contains all departments and employees-->
    <Department ID="D10">
      <Deptno>10</Deptno>
      <Dname>Accounting</Dname>
      <Location>Dallas</Location>
      <EmployeeInfo>
        <Employee ID="E1">
          <Empno>1001</Empno>
          <Ename>Jag</Ename>
          <Sal>2300</Sal>
          <Deptno>10</Deptno>
        </Employee>
      </EmployeeInfo>
    </Department>
  </DepartmentInfo>
</CompanyInfo>
```

Vocabularies can be defined formally Using a **Schema Definition Language.** Two popular ones**,** Document Type Definitions (DTD) and XML Schema Definition Language (XSD). See DTD example below:



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="orderperson" type="xs:string"/>
      <xs:element name="shipto">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="item" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="note" type="xs:string" minOccurs="0"/>
            <xs:element name="quantity" type="xs:positiveInteger"/>
            <xs:element name="price" type="xs:decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Sample XSD Schema

Definitions (1/2)

- "*A Web service is a software application identified by a URI, whose interfaces and bindings are capable of been defined, described, and discovered as XML artifacts. A Web service supports direct interaction with other software agents using XML-based messages exchanged via internet-based protocols*". Source: W3C's Web Services Architecture Working Group
- "*Any process that can be integrated into external systems through valid XML documents over Internet protocols*" Source: Robak, S., & Franczyk, B.(2003).

Definitions (2/2)

- *A piece of business logic accessible via the Internet using open standards* (Microsoft).
- *Encapsulated, loosely coupled, contracted software functions, offered via standard protocols over the Web* (DestiCorp).
- *Services are self-contained, reusable software modules that are independent of applications and the computing platforms on which they run. Services have well defined interfaces and allow a 1:1 mapping between business tasks and the exact IT components needed to execute the task* (IBM).

Web Services & SOA

- Web Service = program accessible over the Web
- Service-Oriented Architecture (SOA):
- Use Web Services as basic building blocks
- Dynamically find & invoke those Web services that allow to solve a particular request
- Web Service Basic Technologies
- WSDL Web Service Description Language
- SOAP XML Data exchange protocol for the Web
- UDDI registry for Web services

Components vs Services

source: ***www.iks.inf.ethz.ch/education/ws05/eai/slides/lec5.pdf***

- Software components are reusable
- To be used a component must:
  - Be packaged and deployed as part of some larger application system
  - Fit with the existing framework used to develop the system
- Components can be sold
  - Components developers charge a per-deployment basis: whenever a new client downloads the component
- There are many component frameworks available for building distributed systems (J2EE, DCOM, .NET, CORBA
- The problem is: they are not compatible

- Services are reusable
- To be used a service must:
  - Be published on the Web
  - Advertise its description and location to potential clients across the Web so that they access it using standard protocols
- Web services can be sold too
  - Service providers can charge a per-call basis: each time an existing client interacts with a service by exchanging a new message.
- Web services can be composed into larger systems and can be found on the Web
- Unlike components, Web services do not have to be downloaded and deployed in order to be used by clients.

Benefits of Web services

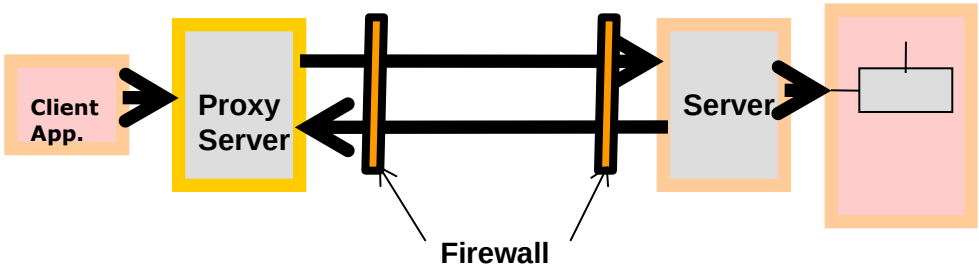
- The Web service architecture represented by SOAP, UDDI and WSDL is a direct descendant of conventional middleware platforms. They can be seen as the most basic extensions that are necessary to allow conventional synchronous (RPC-based) middleware to achieve interoperability.
- The model and even the notation followed in this architecture mimics to a very large extent what has been done in RPC, TP-Monitors, CORBA, etc.
- This dependency gives a very good hint of what can be done with these technologies today and what is missing to obtain a complete platform for business to business electronic commerce.
- First implementations are just extensions of existing platforms to accept innovations through a web service interface (e.g., database stored procedure published as Web services).
- One important difference with conventional middleware is related to the standardization efforts at W3C that should guarantee
  - Platform independence (hardware, OS)
  - Programming language neutrality
  - Portability across Vendor/Middleware tools
- What is the price to pay? Currently Web services standards are still rapidly evolving and the performance of some of the available tools and protocols are quite poor.

Simple Object Access Protocol

SOAP

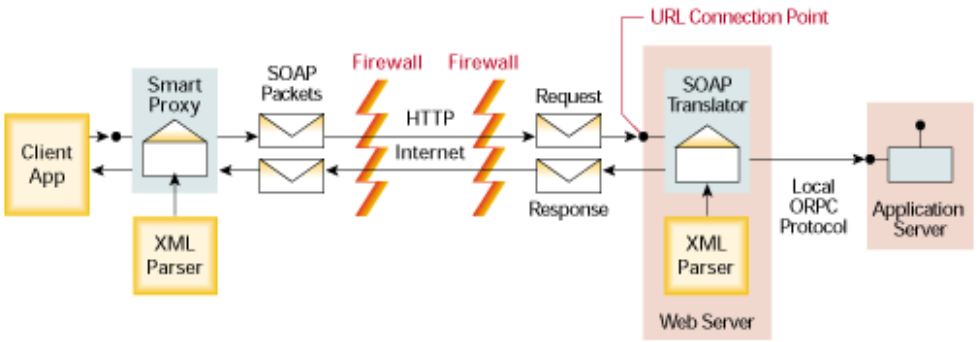
Problem Set and Solutions

- Why XML?
  - Simple text markup language
  - Platform, language and vendor agnostic
  - Easily extensible
  - Capable of solving interoperable problem
- Why HTTP?
  - Ubiquitous
  - Supported by every Web browser and server
  - Effective technology for transferring text, graphics and other information
- XML + HTTP = SOAP



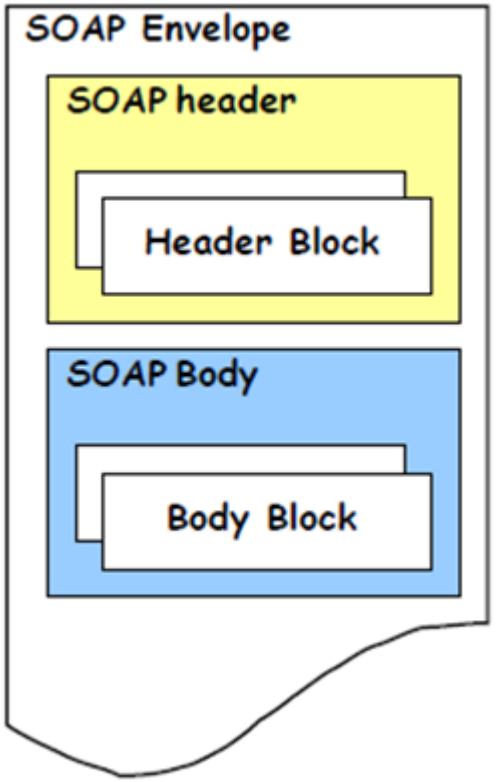
Fundamentals of SOAP

- What is SOAP
- Simple, lightweight protocol for exchanging XML messages over the Web using HTTP, SMTP and SIP. (in a decentralized, distributed environment)
- Primary focus of SOAP is Remote Procedure Calls transported via HTTP
- Facilitates interoperability in a platform-independent manner
- Similar to DCOM, CORBA and JAVA RMI; the main difference is that SOAP messages are written entirely in XML
- What is NOT:
- Object activation
- Bi-directional communications
- Distributed garbage collection
- Language bindings:
- Good for interoperability
- Source of payload is immaterial



SOAP Messaging

- Messages are structured within an envelope where the application encloses the data to be sent
- SOAP Elements
- **Envelope**: top XML element representing the message
- **Header**: (optional)
  - Determines how a recipient of a SOAP message should process the message
  - infrastructure level data: authentication, message routes, transaction management
- **Body**: (mandatory)
  - Exchanges information intended for the recipient of the message
  - typical use for RPC calls and error reporting
- A SOAP conversation includes:
- SOAP Request
  - Specify method name, method parameters, etc.
- SOAP Response
  - Specify return values or error conditions



SOAP Simple Example

```
<Envelope>
<Header>
    <transId>
    345
   </transId>
   </Header>
  <Body>
    <Add>
       <n1>3</n1>
       <n2>4</n2>
    </Add>
  </Body>
</Envelope>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/
envelope/" SOAP-
ENV:encodingStyle="http://
schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Header>
     <t:transId xmlns:t="http://a.com
/trans">
      345
     </t:transId>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:Add xmlns:m="http://a.com/
Calculator">
         <n1>3</n1>
         <n2>4</n2>
      </m:Add>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soa
p/envelope/"
   SOAP-ENV:encodingStyle="http://
schemas.xmlsoap.org/soap/encoding/"
>
   <SOAP-ENV:Header>
      <t:transId xmlns:t="http://a.
com/trans">345</t:transId>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:AddResponse xmlns:m=
"http://a.com/Calculator">
         <result>7</result>
      </m:AddResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Message Processing Model

- SOAP specifies in detail how messages must be processed (in particular, how header entries must be processed)
- Each SOAP node along the message path looks at the role associated with each part of the message
- There are three standard roles: none, next, or ultimateReceiver
- Applications can define their own roles and use them in the message
- The role determines who is responsible for each part of a message
- if a block does not have a role associated to it, it defaults to ultimateReceiver
- If a mustUnderstand flag is included, a node that matches the role specified must process that part of the message, otherwise it must generate a fault and not forward the message any further
- SOAP 1.2 includes a relay attribute. If present, a node that does not process that part of the message must forward it (i.e., it cannot remove the part)
- The use of the relay attribute, combined with the role next, is useful for establishing persistence information along the message path (like session information)

Pros and Cons

- Advantages of SOAP
- Uses HTTP which is widely used and scalable
- Wide remote system interoperability
- Flexible for growth because of XML properties
- It but can be used for RPC.
- Disadvantages of SOAP
- No good way to describe the serialization pattern (XML schema is optional at this point)
- Parsing of SOAP packet and mapping to objects reduces performance
- Doesn’t implement security because it is a wire protocol—relies on HTTP

SOAP Summary

- SOAP, in its current form, provides a basic mechanism for:
- encapsulating messages into an XML document
- mapping the XML document with the SOAP message into an HTTP request
- transforming RPC calls into SOAP messages
- simple rules on how to process a SOAP message
- SOAP is a very simple protocol intended for transferring data from one middleware platform to another.
- SOAP takes advantage of the standardization of XML to resolve problems of data representation and serialization (it uses XML Schema to represent data and data structures, and it also relies on XML for serializing the data for transmission). As XML becomes more powerful and additional standards around XML appear, SOAP can take advantage of them by simply indicating what schema and encoding is used as part of the SOAP message. Current schema and encoding are generic but soon there will be vertical standards implementing schemas and encoding tailored to a particular application area (e.g., the efforts around EDI)

Web Service Description Language

WSDL v1.1 & v2.0

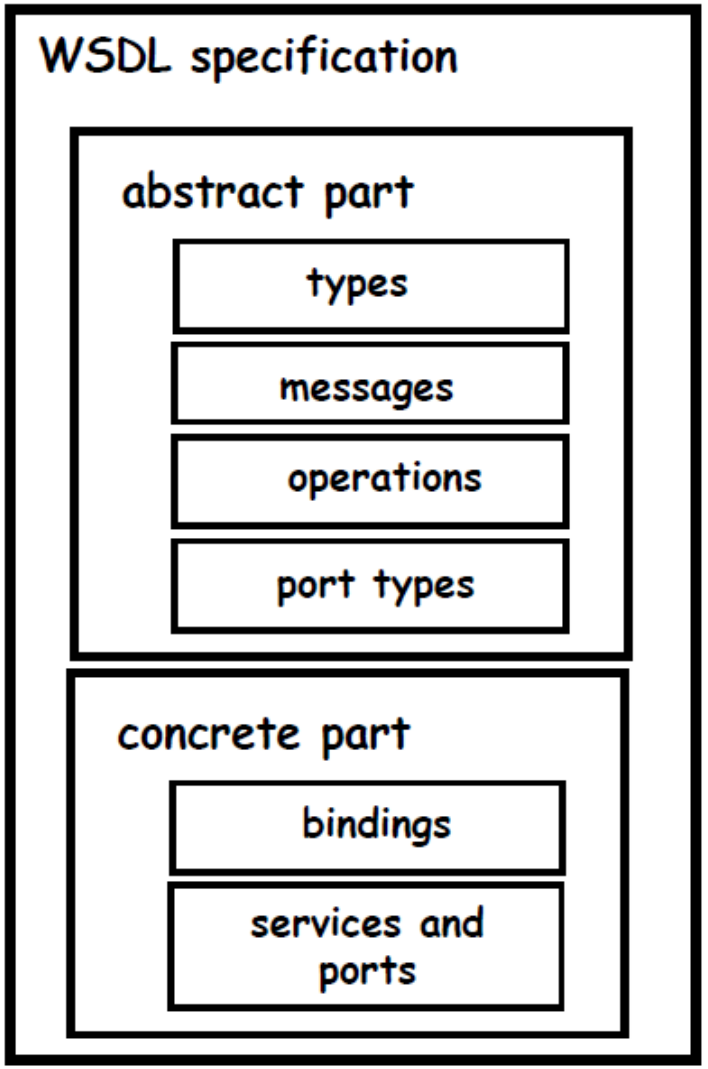
WSDL

- WSDL is an XML grammar to describe Web services by specifying the location of the service and the operations exposed
- WSDL service description indicates how potential clients are intended to interact with the described service.

- Data Elements for V2.0
  - `description`
  - `types`
  - `interface`
  - `binding`
  - `service`
- Data Elements for V1.1
  - `definitions`
  - `types`
  - `portType`
  - `binding`
  - `service`

WSDL V1.1 Document Structure

- Abstract Part: describes
  - the messages it sends and receives
  - the operation associates with a message exchange pattern with one or more messages
- Concrete Part: specifies
  - transport and wire format details for one or more interfaces
  - a port (an endpoint) associates a network address with a binding
  - a service which groups together endpoints that implement a common interface



Example of WSDL V1.1

```
<?xml version="1.0"?>
<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>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>

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

abstract part

messages

operation and port type

concrete part

binding

port and service

WSDL 1.1 Elements: “definitions”

- “*definitions*” (element):
- Root element
- Defines the name of the service, declares the namespaces used through the WSDL file and contains all the service elemenets described here.
- *targetNamespace* defines the namespace to which the items defined by the WSDL belong

```
<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/" >
```

WSDL 1.1 Elements: Abstract part

- "*types*" element (optional): describes the data types, W3C XML Schema
- "*message*" element: describes the messages, defines the message name and the parameters of the message
- "part" element : name + type/element
- "*portType*" (interface) element: defines operations and their messages (inputs, outputs)
- Operations: one-way, Request-Response, solicit-response, notification
- Allows for overloaded operations

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

messages

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

operation and port type

WSDL 1.1 Elements: Concrete part (1/3)

- "*binding*" element: defines of how the service will be implemented on the wire. Includes:
  - Attributes
    - name: any name
    - type: points to the port type defined in the abstract part
  - "soap:binding" element: its attributes are
    - style: "rpc" or "document"
    - transport: communication protocol:SOAP, transport protocol: HTTP/SMTP
  - "operation" element:
    - Defines each operation that the port exposes
    - Must also specify how the input and output are encoded
    - "literal" for document
    - "SOAP" for rpc

WSDL 1.1 Elements: Concrete part (2/3)

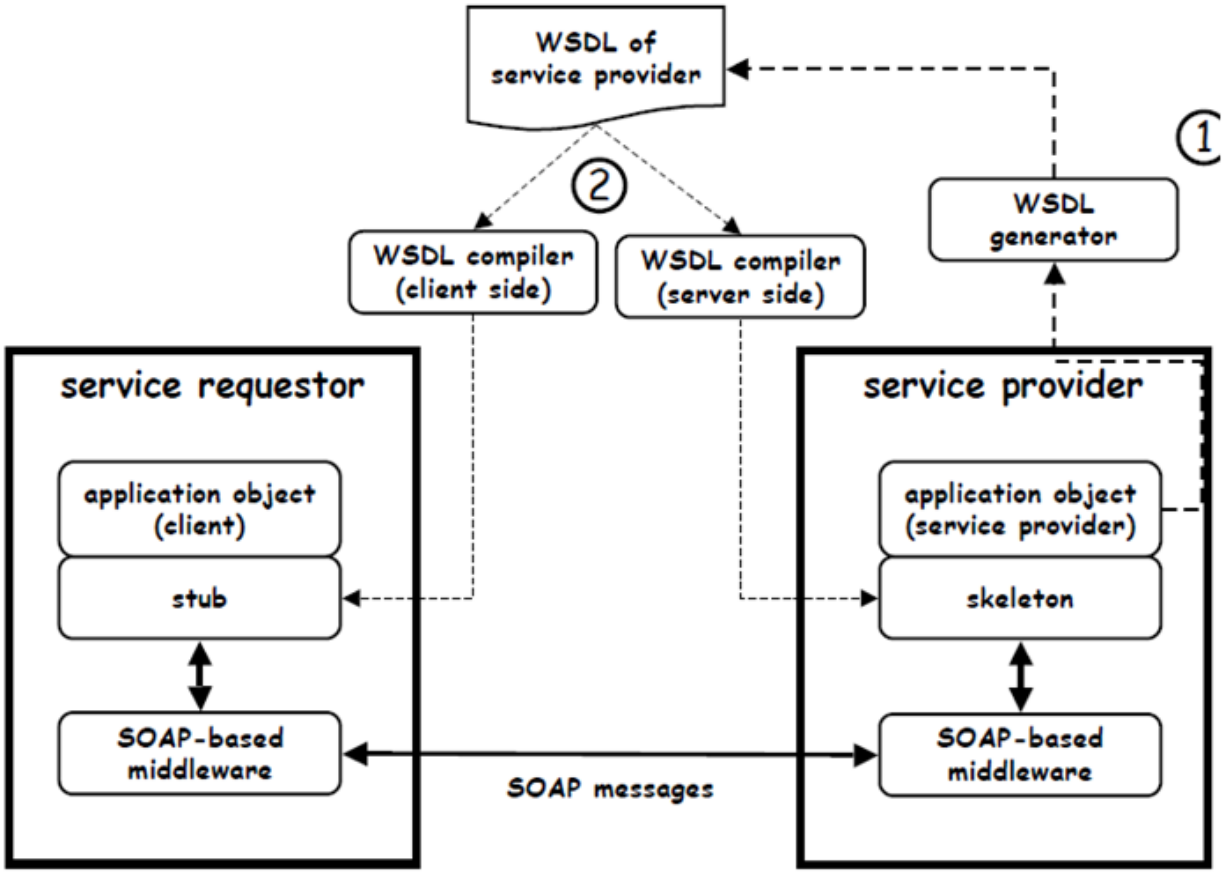
```
<binding name="ProcurementSoapBinding" type="tns:procurementPortType">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="orderGoods">
      <soap:operation soapAction="http://example.com/orderGoods"/>
      <input>
        <soap:body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

WSDL 1.1 Elements: Concrete part (3/3)

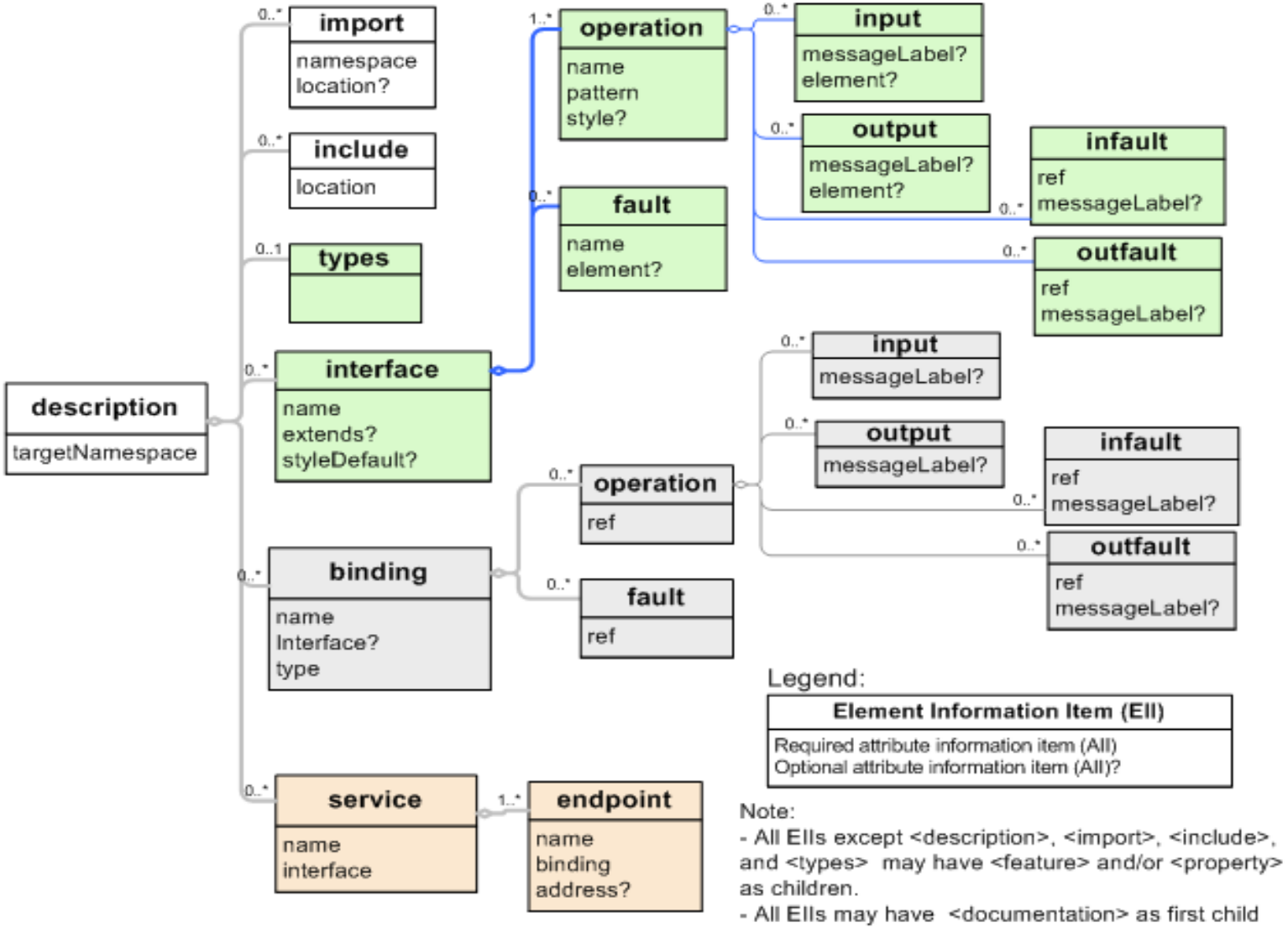
- "`service`": defines the address for invoking the specified service
  - May have more than one endpoint, with each one defined by its own port element
- "`port`": corresponds to a particular binding, and includes information on how to access it (URI)
  - Different ports represents different bindings for the same port type
  - Allows the same functionality to be accessible via multiple transport protocols and interaction styles

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

Using WSDL



WSDL V2.0 Component Model



WSDL V1.1 vs WSDL V2.0

- Authoring styles
- V1.1: <wsdl:import> enables importing other WSDL definitions defined in separate files with the same or different namespace.
- V2.0: <wsdl:import> for WSDL definitions with a different namespace and <wsdl:include> for WSDL definitions with the same namespace
- Namespace
- V1.1: http://schemas.xmlsoap.org/wsdl/
- V2.0: http://www.w3.org/2004/08/wsdl
- Message Exchange Patterns (MEPs)
- V1.1: one-way, request-response, solicit-response, notification
- V2.2:
  - In-Bound: In-Only, Robust In-Only, In-Out, In-Optional-Out
  - Out-Bound: Out-Only, Robust Out-Only, Out-In, Out-Optional-In
- Schemas: V1.1 supports XML Schema, V2.0 supports other schemas such as RELAX NG and Schematron

WSDL Styles for SOAP requests

- Five models depending on WSDL SOAP binding – dictate how to translate a WSDL binding to SOAP message
- RPC/encoded:
  - operation name in the message, message overhead because of namespaces in the parameters, cannot easily be validated not WS-I compliant
- RPC/literal
  - Operation name in the message, WS-I compliant, type encoding eliminated, cannot easily validate the message
- Document/encoded
  - Not followed, not WS-I compliant
- Document/literal
  - No type encoding info, can be validated, a little bit complicated, no operation name in the message, WS-I compliant with restrictions
- Document/literal wrapped
  - Basic characteristics: the input message has a single part; the part is an element; the element has the same name as the operation, the element's complex type has no attributes.
  - Operation name included, no type encoding info, can be easily validated, WS-I compliant but the WSDL is more complicated.

WSDL RPC styles – SOAP requests

```
<message name="myMethodRequest">
<part name="x" type="xsd:int"/>
<part name="y" type="xsd:float"/>
</message> <message name="empty"/>
<portType name="PT">
<operation name="myMethod">
     <input
message="myMethodRequest"/>
     <output message="empty"/>
</operation>
</portType>
```

WSDL file segment

```
<soap:envelope>
<soap:body>
          <myMethod>
               <x
xsi:type="xsd:int">5</x>
               <y
xsi:type="xsd:float">5.0</y>
</myMethod>
 </soap:body>
</soap:envelope>
```

SOAP RPC/literal

```
<soap:envelope>
<soap:body>
          <myMethod>
               <x>5</x>
               <y>5.0</y>
          </myMethod>
 </soap:body>
</soap:envelope>
```

SOAP RPC/encoded

WSDL Document Style - SOAP requests (1/2)

```
<types>
<schema>
          <element name="xElement"
               type="xsd:int"/>
          <element name="yElement"
               type="xsd:float"/>
</schema>
 </types>
<message name="myMethodRequest">
     <part name="x" element="xElement"/>
     <part name="y" element="yElement"/>
</message> <message name="empty"/>
<portType name="PT">
     <operation name="myMethod">
          <input message="myMethodRequest"/>
          <output message="empty"/>
</operation>
</portType>
```

```
<soap:envelope>
<soap:body>

<xElement>5</xElement>

<yElement>5.0</yElement>
</soap:body>
</soap:envelope>
```

SOAP Document/literal

WSDL Document Style - SOAP requests (2/2)

```
<types>
    <schema>
      <element name="myMethod">
        <complexType>
          <sequence>
            <element name="x" type="xsd:int"/>
            <element name="y" type="xsd:float"/>
          </sequence>
        </complexType>
      </element>
      <element name="myMethodResponse">  <complexType/>
    </element>
    </schema>
</types>
<message name="myMethodRequest">
    <part name="parameters" element="myMethod"/>
</message>
<message name="empty">
    <part name="parameters" element="myMethodResponse"/>
 </message>
<portType name="PT">
    <operation name="myMethod">
       <input message="myMethodRequest"/> <output
    message="empty"/>
</operation>
</portType>
```

```
<soap:envelope>
<soap:body>
            <myMethod>
                  <x>5</x>
                  <y>5.0</y>
            </myMethod>
</soap:body>
</soap:envelope>
```

SOAP Document/literal wrapped

35

Universal Description, Discovery and Integration Standard

UDDI V3.0

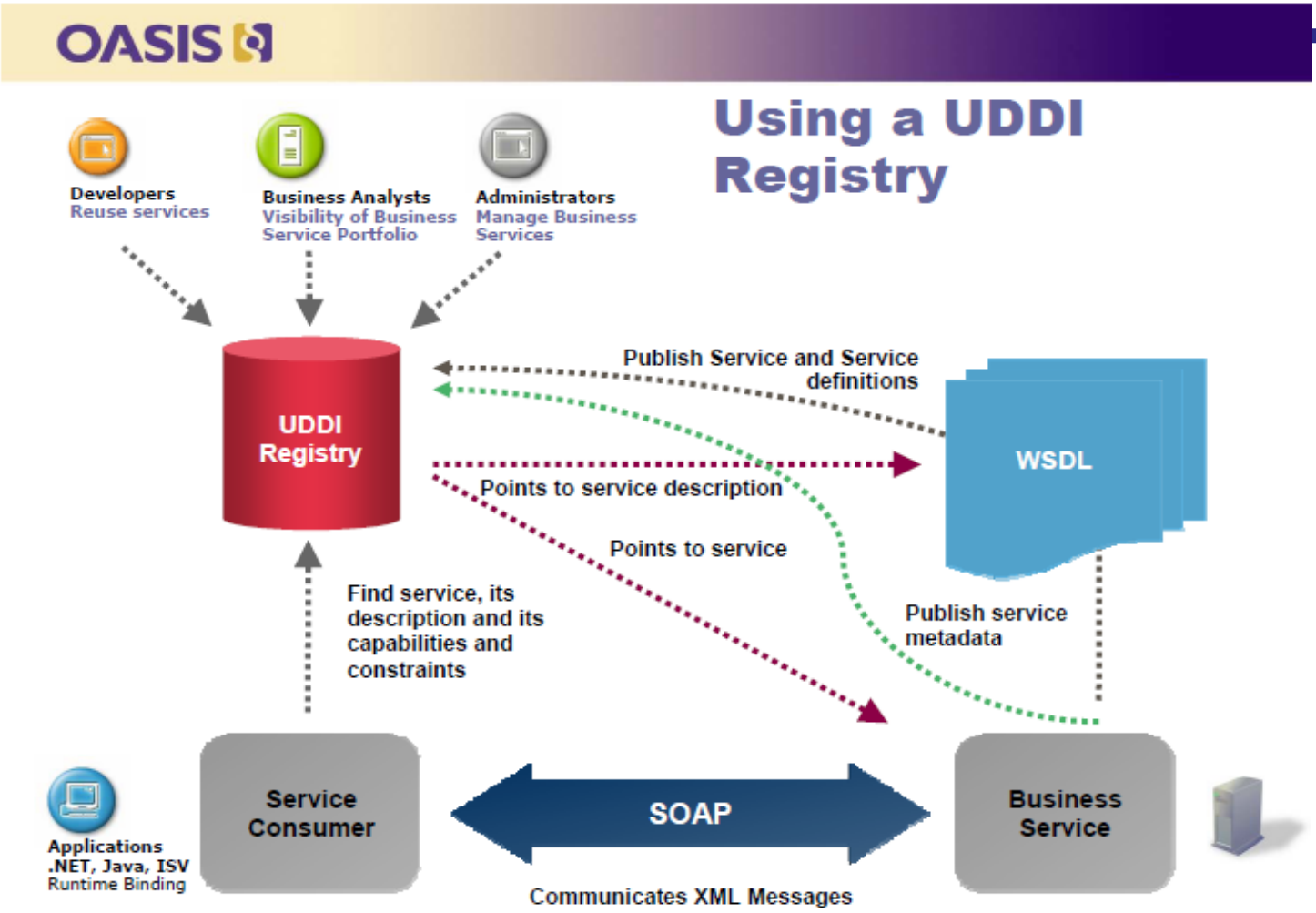
UDDI

- Introduced by OASIS
- UDDI v3 OASIS standard in 2005
- Registry for WS
- Unified and systematic way to find service providers like a “phone directory” of web services
- Specifications
- Schemas for service and business description
- Query and update API for the registry
- WS-I compatible, based on HTTP, IP, SOAP, WSDL standards
- All APIs in the UDDI are defined in XML, wrapped in a SOAP Envelope and sent over HTTP

UDDI

- UDDI registries organization structure
  - White pages: organizations' information (name, description, etc.)
  - Yellow pages: taxonomy of businesses & services
  - Green pages: bindings/technical details for specific services as well as specifications that services implement
- Private vs Public Registries
  - Private: intranet or extranet applications, managed internally
  - Public: UDDI Business Registry (UBR), IBM-Microsoft-SAP and NNT, all nodes are synchronized

UDDI


OASIS
Using a UDDI Registry
Developers
Reuse services
Business Analysts
Visibility of Business Service Portfolio
Administrators
Manage Business Services
UDDI Registry
Publish Service and Service definitions
WSDL
Points to service description
Points to service
Find service, its description and its capabilities and constraints
Publish service metadata
Applications
.NET, Java, ISV
Runtime Binding
Service Consumer
SOAP
Business Service
Communicates XML Messages

UDDI v3 Data Model

- **businessEntity**: describes a business or other organization that provides Web services
- **businessService**: describes a collection of related Web services offered by an organization described by **businessEntity**
- **bindingTemplate**: describes the technical information necessary to use a particular Web service
- **tModel**: describes a “technical model” representing a reusable concept, such as a Web service type, a protocol used by Web services or a catalog system. Used as references to the **bindingTemplate** for designation of the interface specifications for a service.
- **publisherAssertion**: describes, in the view of one **businessEntity**, the relationship that the businessEntity has with another **businessEntity**
- **subscription:** describes a standing request to keep track of changes to the entities described by the subscription

UDDI v3 API Sets

- Node API Sets
- *UDDI Inquiry*: retrieves information from registries
  - find_business, find_service, find_relatedBusinesses, find_tModel,
  - get_bindingDetail, get_businessDetail get_o`perationalInfo, get_serviceDetail, get_tModelDetail
- *UDDI Publicat*ion: inserts/updates info
  - add_publisherAssertions, delete_binding, delete_business, delete_publisherAssertions, delete_service, delete_tModel, get_assertionStatusReport, get_publisherAssertions, get_registeredInfo, save_binding, save_business, save_service, dave_tModel, set_publisherAssertions
- *UDDI Security*:
  - discard_authToken, get_authToken
- *UDDI Custody Transfer*
- *UDDI Subscription:* receiving info regarding changes in registry
- *UDDI Replication:* perform replication and issue notifications

- Client API Sets
- *UDDI Subscription listener*
- *UDDI Value Set*

UDDI Node, Registry & Affiliated Registries

- Definition of the hierarchical relationship between instances of a UDDI implementation
- Three major classifications of UDDI server:
- Node: UDDI server that supports at least the minimum set of functionality defined in the specification. It is a member of exactly one UDDI registry.
- Registry: composed of one or more nodes. A registry performs the complete set of functionality as defined in the specification.
- Affiliated Registries: individual UDDI registries that implement policy-based sharing of information among them
- They share a common namespace for UDDI keys that uniquely identify data records

<businessEntity businessKey="ba744ed0-3aaf-11d5-80dc-002035229c64" operator="www.ibm.com/services/uddi" authorizedName="0100001QS1">

<discoveryURLs>

<discoveryURL useType="businessEntity"> http://www.ibm.com/services/uddi/uddiget?businessKey= BA744ED0-3AAF-11D5-80DC-002035229C64

</discoveryURL>

</discoveryURLs>

<name>.....</name> <description>.....</description>

<contacts> .....</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 ser</description>

<accessPoint URLType="http">http://services.xmethods.net:80/soap</accessPoint>

<tModelInstanceDetails>

<tModelInstanceInfo tModelKey="uuid:0e727db0-3e14-11d5-98bf-002035229c64" />

</tModelInstanceDetails>

</bindingTemplate>

</bindingTemplates>

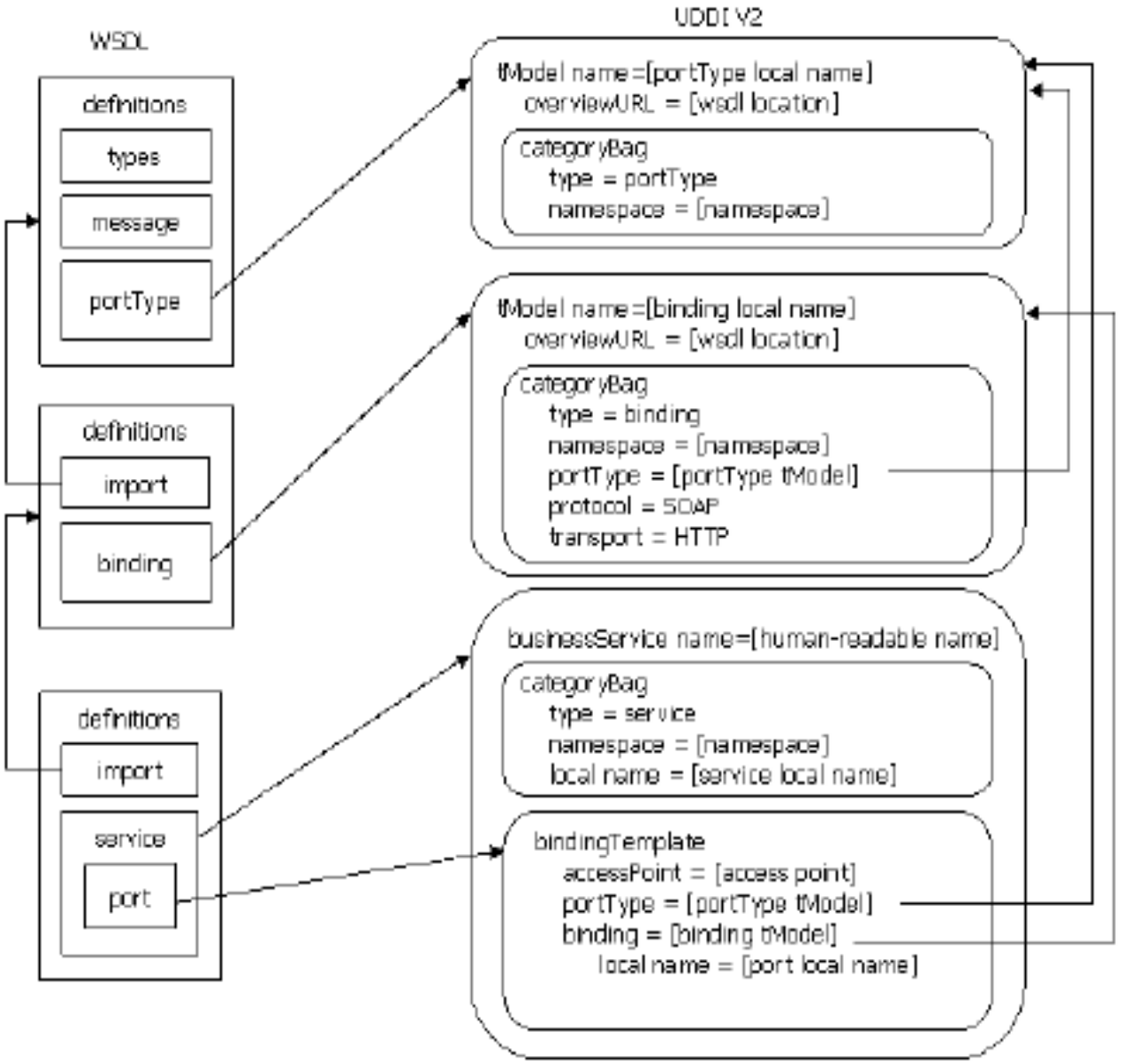
</businessService>

</businessServices>

UDDI Example 2

```
<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</ove
rviewURL>
</overviewDoc>
<categoryBag>
      <keyedReference tModelKey="uuid:c1acf26d-9672-4404-9d70-
39b756e62ab4" keyName="uddi-org:types" keyValue="wsdlSpec" />
</categoryBag>
</tModel>
```

WSDL to UDDI mapping



Policy

- Policies within UDDI are statements of required and expected behavior.
- Policies:
  - The registry defines the domain of the policy for the nodes
  - The registry may delegate the definition of a particular policy to one or more of the nodes within its domain.
  - A hierarchical relationship between registry policies and node policies
    - e.g, whether a registry allows nodes to specify polices
  - The Registries also identify the Policy Decision Points & Policy Enforcement Points
  - Affiliated registries are sets of registries that share compatible policies for assigning keys and managing data

Security model for UDDI registry

- The security model for a UDDI registry can be characterized by the collection of registry and node policies and the implementation of these policies by a UDDI node.
- In order to authorize or restrict access to data in a UDDI registry, an implementation of a UDDI node MAY be integrated with one or more identification systems.
- Integration of UDDI APIs and data with an identification system MAY be implemented through the authentication and authorization APIs to provide access control.
- Other authentication and authorization mechanisms and policies are represented in UDDI through use of tModels.
- UDDI also supports XML Digital Signatures on UDDI data to enable inquirers to verify the integrity of the data with respect to the publisher.

Standards Convergence to UDDI

- Several Domain Specific standards
- Policy: mapping of WS-Policy onto UDDI
- Orchestration: publication and discovery of metrics and manageability
- Management: publication and discovery of metrics and manageability provider information – WSDM
- Portal Integration: publication and discovery of WSRP Producer and Portlet services

Web services with Java

- xml.apache.org/axis - Apache **AXIS** (Apache Extensible Interaction System)
  - a Java-based implementation of SOAP+WSDL
  - largely allows the programmer to forget these technologies
  - typically used together with Tomcat
- www.alphaworks.ibm.com/tech/ettk - alphaWorks's **EETK** (Emerging Technologies Toolkit)
  - support for SOAP, WSDL, UDDI and much more...
- java.sun.com/webservices/downloads/webservicespack.html - Sun's **Java WSDP** (Web Services Developer Pack)
  - support for SOAP, WSDL, UDDI, ...
  - JAX-RPC maps SOAP/WSDL to RMI (Java Remote Method Invocations)

Τέλος Ενότητας

Χρηματοδότηση

•Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

•Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

•Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

•Ως **Μη Εμπορική** ορίζεται η χρήση:

–που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο

–που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο

–που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

•Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

.

Σημείωμα Αναφοράς