WEB SERVICES

(v 0.1)

PXC

Manel Guerrero <guerrero@ac.upc.edu>

Contents

- Web Services
- XML-RPC
- SOAP
- WSDL
- UDDI

Sources

(That is, places from which we've done merciless cut 'n' pastes)

- Top Ten FAQs for Web Services

  http://webservices.xml.com/pub/a/ws/2002/02/12/webservicefaqs.html
- http://www.wikipedia.org/
- Slides FIB-AAD Jordi Torres.

Web Services

- A web service is any piece of software that makes itself available over the Internet and uses a standardized XML messaging system.
- XML is used to encode all communications to a Web service. For example, a client invokes a Web service by sending an XML message, then waits for a corresponding XML response.
- Because all communication is in XML, Web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications.

Web Services: Tell me more

- Beyond this basic definition, a Web service may also have two additional (and desirable) properties:
  - A Web service can have a public interface, defined in XML. The interface describes all the methods available to clients. Interface definition is accomplished via the Web Service Description Language (WSDL)
  - If you create a Web service, there should be some relatively simple mechanism for you to publish it. Likewise, there should be some simple mechanism for interested parties to locate the service and locate its public interface. The most prominent directory of Web services is currently available via Universal Description, Discovery, and Integration (UDDI).

Web Services: Standards used

- XML: All data to be exchanged is formatted with XML tags. The encoded message may conform to a messaging protocol such as SOAP, JAX-RPC, or XML-RPC.
- Transport protocols: data can be transported between applications using common protocols such as HTTP, FTP, SMTP and XMPP.
- WSDL: The public interface to the web service is described by Web Services Description Language, or WSDL. This is an XML-based service description.
- UDDI: The web service information is published using this protocol. It should enable applications to look up web services information in order to determine whether to use them.

Web Services: Protocol Stack

- **Service Discovery**: This layer is responsible for centralizing services into a common registry, and providing easy publish/find functionality. Currently, service discovery is handled via the UDDI.
- **Service Description**: This layer is responsible for describing the public interface to a specific Web service. Currently, service description is handled via the WSDL.
- **XML Messaging**: This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this includes XML-RPC and SOAP.
- **Service Transport**: This layer is responsible for transporting messages between applications. Currently, this includes HTTP, SMTP, FTP, and newer protocols, such as Blocks Extensible Exchange Protocol (BEEP).

Web Services: Advantages

- **Interoperability**: Web services provide interoperability between various software applications running on disparate platforms.
- Use of **open standards** and protocols. Protocols and data formats are text-based where possible, making it easy to comprehend.
- The use of **HTTP**: Web services can work through many common firewall security measures without changing the firewall rules.
- Web services easily allow software and **services** from different companies and locations to be **combined easily** to provide an integrated service.
- **Code reusability**: Web services allow the reuse of services and components within an infrastructure.

Web Services: Disadvantages

- Web services standards for features such as **transactions** are currently **nonexistent** or still in their infancy compared to more mature distributed computing open standards such as CORBA.
- Web services may suffer from **poor performance** compared to other distributed computing approaches such as RMI, CORBA, or DCOM. This is a common trade-off when choosing text-based formats. XML explicitly does not count among its design goals either conciseness of encoding or efficiency of parsing.
- By utilizing HTTP, web services can **evade** existing **firewall security** measures whose rules are intended to block or audit communication between programs on either side of the firewall.

Web Services: Reasons to use them

- They provide a very **loose coupling** between an application that uses the web service and the web service itself. This allows either piece to change without negatively affecting the other, as long as the interface remains unchanged.
- Web services are gaining in popularity because they are based on **HTTP** over TCP port 80, which is always open because it is used for web browsing.
- Is a good interface to access functionality on different computers on a network. Web services have a solid frameworks and are easy to use.

Web Services: Platforms

- Web services can be deployed by using application server software. A sample of application servers:
  - Axis and the Jakarta Tomcat server (both at the Apache project)
  - ColdFusion MX from Macromedia
  - Microsoft .NET servers from Microsoft
  - Mono development platform from Novell
  - WebSphere Application Server from IBM (based on the Apache server and the J2EE platform)
  - Zope is an object oriented web application server written in Python
  - Many more

XML-RPC

- XML-RPC is a remote procedure call protocol encoded in XML. It's a very simple protocol, defining only a handful of data types and commands.
- Requests are encoded in XML and sent via HTTP POST. XML responses are embedded in the body of the HTTP response.
- It was first created by Dave Winer of UserLand Software in 1995 with Microsoft. However, Microsoft considered it too simple and started adding functionality. After several rounds of this, the standard was no longer so simple and became what is now SOAP.

XML-RPC: Simple Data Types

- base64: `<base64>eW91IGNhbid0IHJlYWQgdGhpcyE=</base64>`
  Base 64-encoded binary data

- boolean: `<boolean>1</boolean>`
  Boolean logical value (0 or 1)

- date/time: `<dateTime.iso8601>19980717T14:08:55</dateTime.iso8601>`
  Date and time

- double: `<double>-12.53</double>`
  Double precision floating number

- integer: `<i4>42</i4>`
  Whole number, integer

- string: `<string>Hello world!</string>`
  String of characters. Must follow XML encoding.

- nil: `<nil/>`
  Null value; an XML-RPC extension. (like a NULL pointer in C)

XML-RPC: Structured Data Types

- array: Array of values, storing no keys

```
<array>
  <data>
    <value><i4>1404</i4></value>
    <value><string>Something Here</string></value>
    <value><i4>1</i4></value>
  </data>
</array>
```

- struct: Array of values, storing keys (<name> element)

```
<struct>
  <member>
    <name>foo</name>
    <value><i4>1</i4></value>
  </member>
  <member>
    <name>bar</name>
    <value><i4>2</i4></value>
  </member>
</struct>
```

XML-RPC: Examples

- An example of a typical XML-RPC request would be:

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
        <value><i4>41</i4></value>
    </param>
  </params>
</methodCall>
```

- An example of a typical XML-RPC response would be:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>South Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

XML-RPC: XML-RPC fault

- A typical XML-RPC fault would be:

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>4</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>Too many parameters.</string></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```

SOAP

- SOAP originally was an acronym for Simple Object Access Protocol, but the acronym was dropped in Version 1.2 of the SOAP specification.
- Originally designed by Dave Winer, Don Box, Bob Atkinson, and Mohsen Al-Ghosein in 1998 with backing from Microsoft (where Atkinson and Al-Ghosein worked at the time).
- The SOAP specification is currently maintained by the XML Protocol Working Group of the World Wide Web Consortium.

SOAP: Tell me more

- SOAP is a standard for exchanging **XML**-based messages over a computer network, normally using **HTTP**. SOAP forms the foundation layer of the web services stack, providing a basic messaging framework that more abstract layers can build on.
- SOAP facilitates the **Service-Oriented architectural pattern**.
- There are several different types of messaging patterns in SOAP, but by far the most common is the Remote Procedure Call (**RPC**) pattern, where one network node (the **client**) **sends a request message** to another node (the server), and the **server immediately sends a response message** to the client.

SOAP: Good or bad?

- The somewhat lengthy syntax of XML can be both a benefit and a drawback. Its format is easy for humans to read, but can be complex and slow down processing times.
- CORBA, GIOP and DCOM use much shorter, binary message formats.
- On the other hand, hardware appliances are available to accelerate processing of XML messages.
- Binary XML (the use of the word "XML" is controversial here) is also being explored as a means for streamlining the throughput requirements of XML.

Structure of a SOAP message

- A SOAP message is contained in an envelope.
- Within this envelope are two additional sections: the header and the body of the message. SOAP messages use XML namespaces.
- The header contains relevant information about the message. For example, a header can contain the date the message is sent, or authentication information. It is not required, but, if present, must always be included at the top of the envelope when it is present.
- The body contains the data of the message (the message itself).

SOAP Request Example

- Request to a weather service (with the HTTP Headers omitted):

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://www.w3.org/2001/09/soap-envelope"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:getWeather
         xmlns:ns1="urn:examples:weatherservice"
         SOAP-ENV:encodingStyle=
            "http://www.w3.org/2001/09/soap-encoding">
         <zipcode xsi:type="xsd:string">10016</zipcode>
      </ns1:getWeather>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Reply Example

- Reply from the weather service (with the HTTP Headers omitted):

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://www.w3.org/2001/09/soap-envelope"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:getWeatherResponse
         xmlns:ns1="urn:examples:weatherservice"
         SOAP-ENV:encodingStyle="http://www.w3.org/2001/09/soap-
encoding">
         <return xsi:type="xsd:int">65</return>
      </ns1:getWeatherResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Endpoint Reference

- The endpoint reference of a webservice:

  `85.34.235.95:80/servlet/rpcrouter/getWeather`

- Request to a weather service with the HTTP Headers:

```
POST /servlet/rpcrouter HTTP/1.1
Host: www.messages.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnn
SOAPAction: ""

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://www.w3.org/2001/09/soap-envelope"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:getWeather [...]
```

HTTP Header of the SOAP Reply

- Reply from the weather service with the HTTP Headers:

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


<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    [...]
</SOAP-ENV:Envelope>
```

WSDL

- The Web Services Description Language (WSDL) is an XML format published for describing Web services.
- Version V 1.1 has not been endorsed by the World Wide Web Consortium (W3C), however it has just (May 11, 2005) released a draft for version 2.0, that will be a recommendation (an official standard), and thus endorsed by the W3C.
- The Web Services Description Language (WSDL) currently represents the service description layer within the Web service protocol stack.
- Using WSDL, a client can locate a Web service, and invoke any of the publicly available functions. With WSDL-aware tools, this process can be entirely automated, enabling applications to easily integrate new services with little or no manual code.

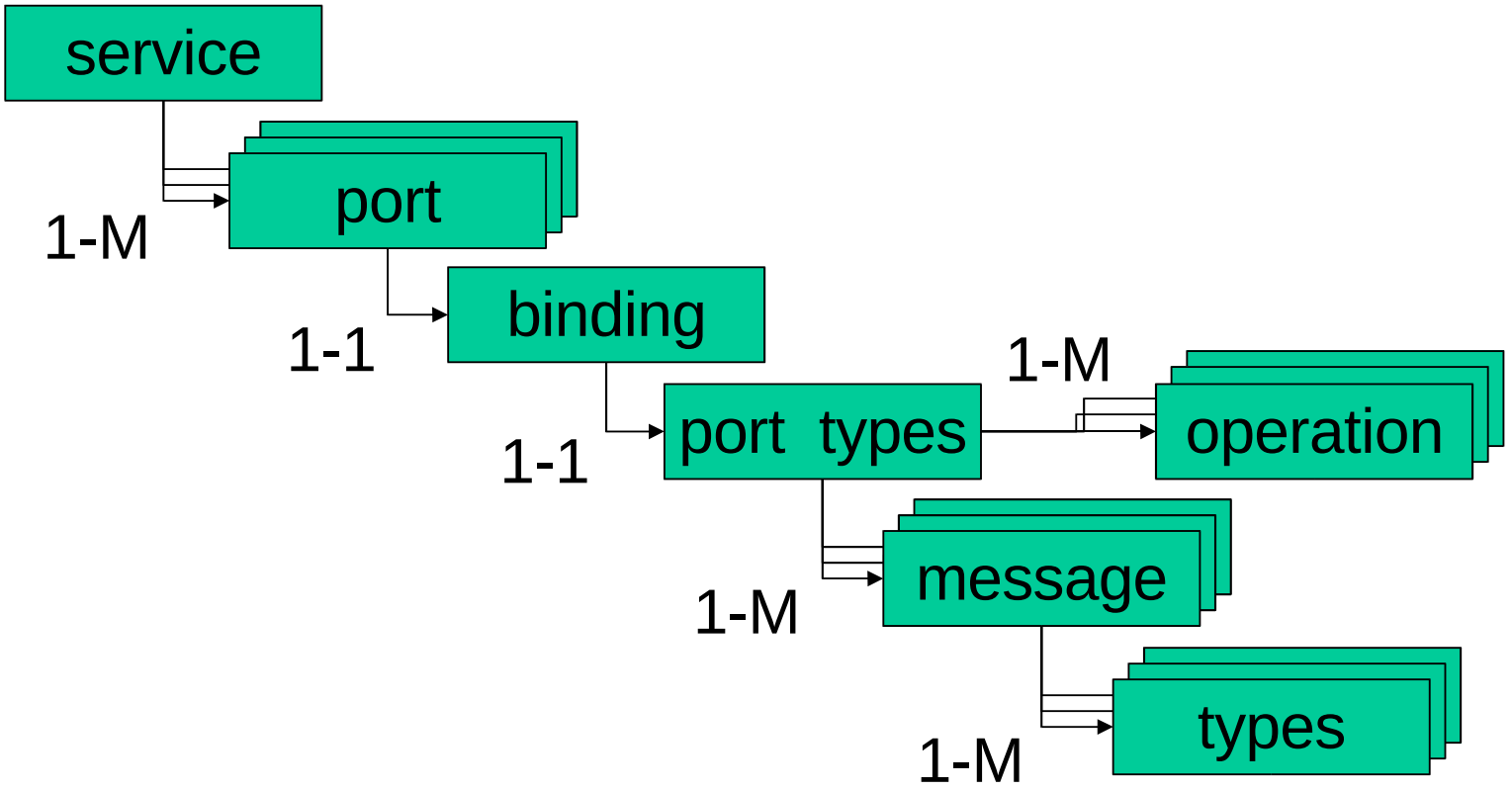
WSDL: Tell me more

- WSDL is an XML grammar for specifying a public interface for a Web service. This public interface can include the following:
  - Information on all publicly available functions.
  - Data type information for all XML messages.
  - Binding information about the transport protocol to be used.
  - Address information for locating the specified service.
- WSDL is often used in combination with SOAP and XML Schema to provide web services over the internet. A client connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

WSDL: Elements

- **Types**: A container for data type definitions using some type system (such as XML Schema).
- **Message**: An abstract, typed definition of the data being communicated.
- **Operation**: An abstract description of an action supported by the service.
- **Port Type**: An abstract set of operations supported by one or more endpoints.
- **Binding**: A concrete protocol and data format specification for a particular port type.
- **Port**: A single endpoint defined as a combination of a binding and a network address.
- **Service**: A collection of related endpoints.

WSDL: Structure



WSDL: Example (1/2)

- <message> specifies the individual XML messages that are transferred between computers. In this case, 'getWeatherRequest' and 'getWeatherResponse'.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="WeatherService"
   targetNamespace="http://www.ecerami.com/wsdl/WeatherService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.ecerami.com/wsdl/WeatherService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="getWeatherRequest">
      <part name="zipcode" type="xsd:string"/>
   </message>
   <message name="getWeatherResponse">
      <part name="temperature" type="xsd:int"/>
   </message>

   <portType name="Weather_PortType">
      <operation name="getWeather">
         <input message="tns:getWeatherRequest"/>
         <output message="tns:getWeatherResponse"/>
      </operation>
   </portType>
```

WSDL: Example (2/2)

- <service> specifies that the service is available via SOAP and its URL.

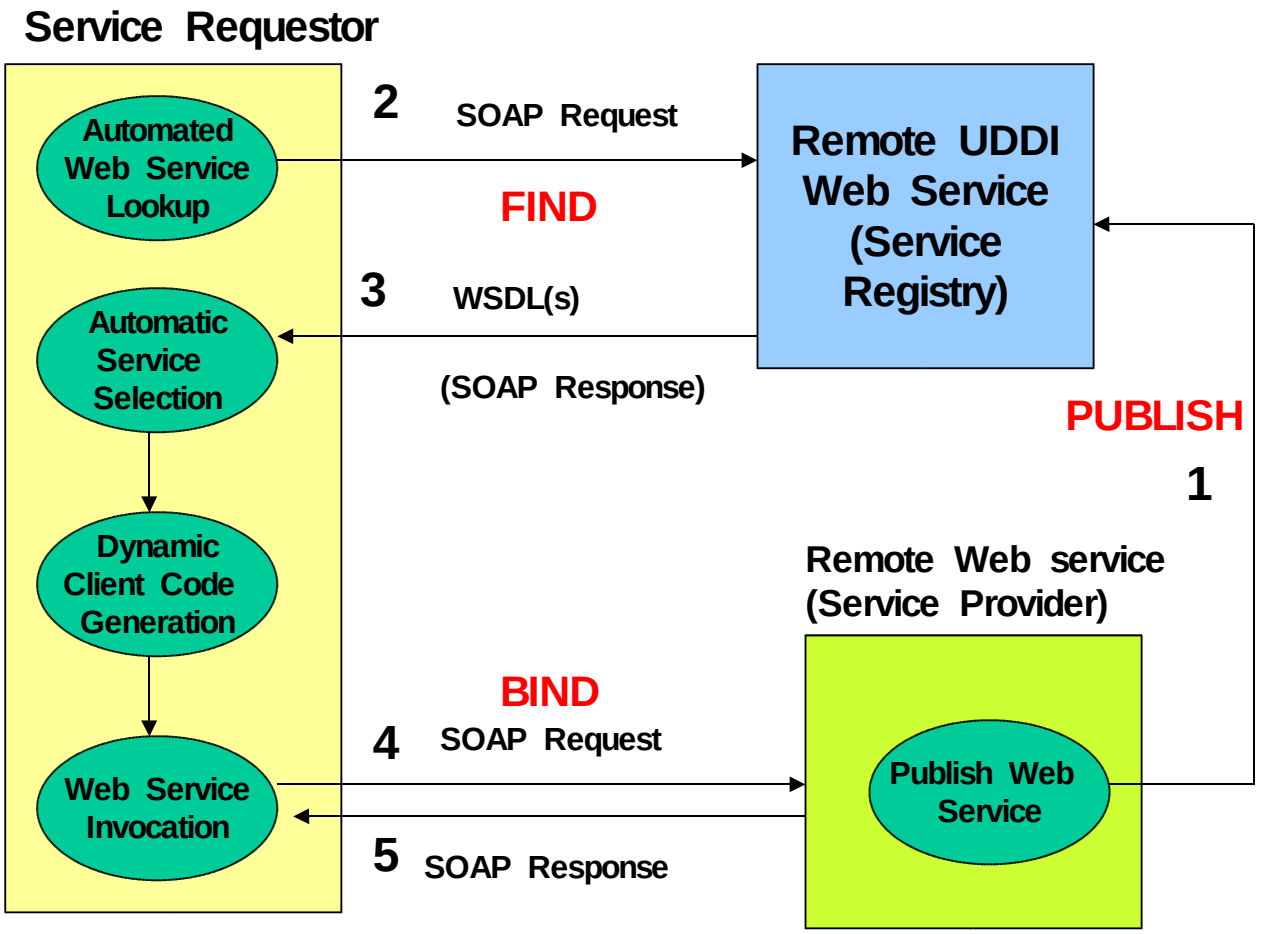
```
<binding name="Weather_Binding" type="tns:Weather_PortType">
   <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="getWeather">
      <soap:operation soapAction=""/>
      <input>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:weatherservice" use="encoded"/>
      </input>
      <output>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:weatherservice" use="encoded"/>
      </output>
   </operation>
</binding>

<service name="Weather_Service">
   <documentation>WSDL File for Weather Service</documentation>
   <port binding="tns:Weather_Binding" name="Weather_Port">
      <soap:address
         location="http://localhost:8080/soap/servlet/rpcrouter"/>
   </port>
</service>
</definitions>
```

UDDI

- UDDI (Universal Description, Discovery, and Integration): A platform-independent, XML-based registry for businesses worldwide to list themselves on the Internet. UDDI is an open industry initiative (sponsored by OASIS) enabling businesses to discover each other and define how they interact over the Internet. A UDDI business registration consists of three components:
  - White Pages: Address, contact, and known identifiers
  - Yellow Pages: Industrial categorizations based on standard taxonomies
  - Green Pages: Technical information about services exposed by the business
- UDDI is designed to be interrogated by SOAP messages and to provide access to WSDL docs describing the protocol bindings and message formats required to interact with the web services listed in its directory.

UDDI: Dynamic Service Invocation



- The UDDI directory exposes a set of APIs in the form of a SOAP-based Web Service.
- Publication API: Authenticated set of operations that allow organizations to publish businesses, services, service type specifications
- Inquiry API: Non authenticated public set of operations that allows users to extract information out of the UDDI registry.

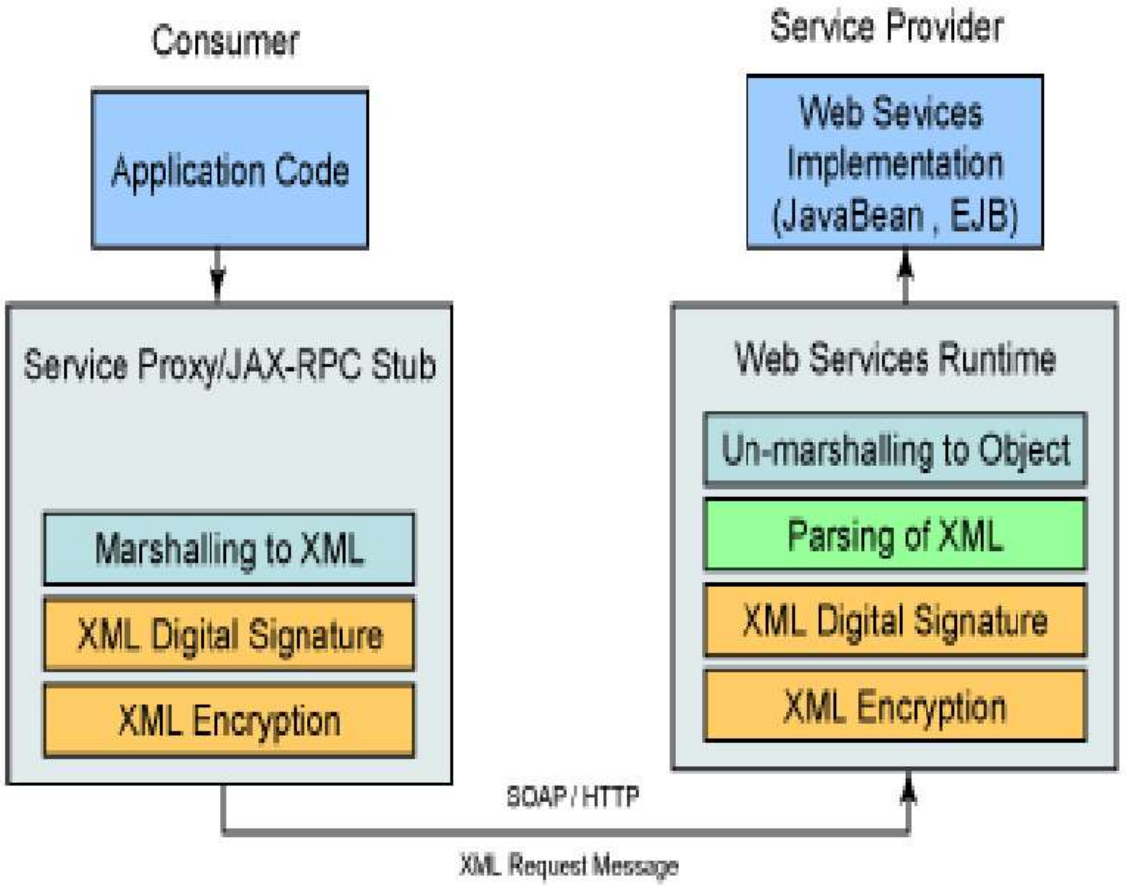
Business registry nodes

- http://uddi.microsoft.com
- http://uddi.ibm.com
- http://uddi.<companyname>.com
- ...

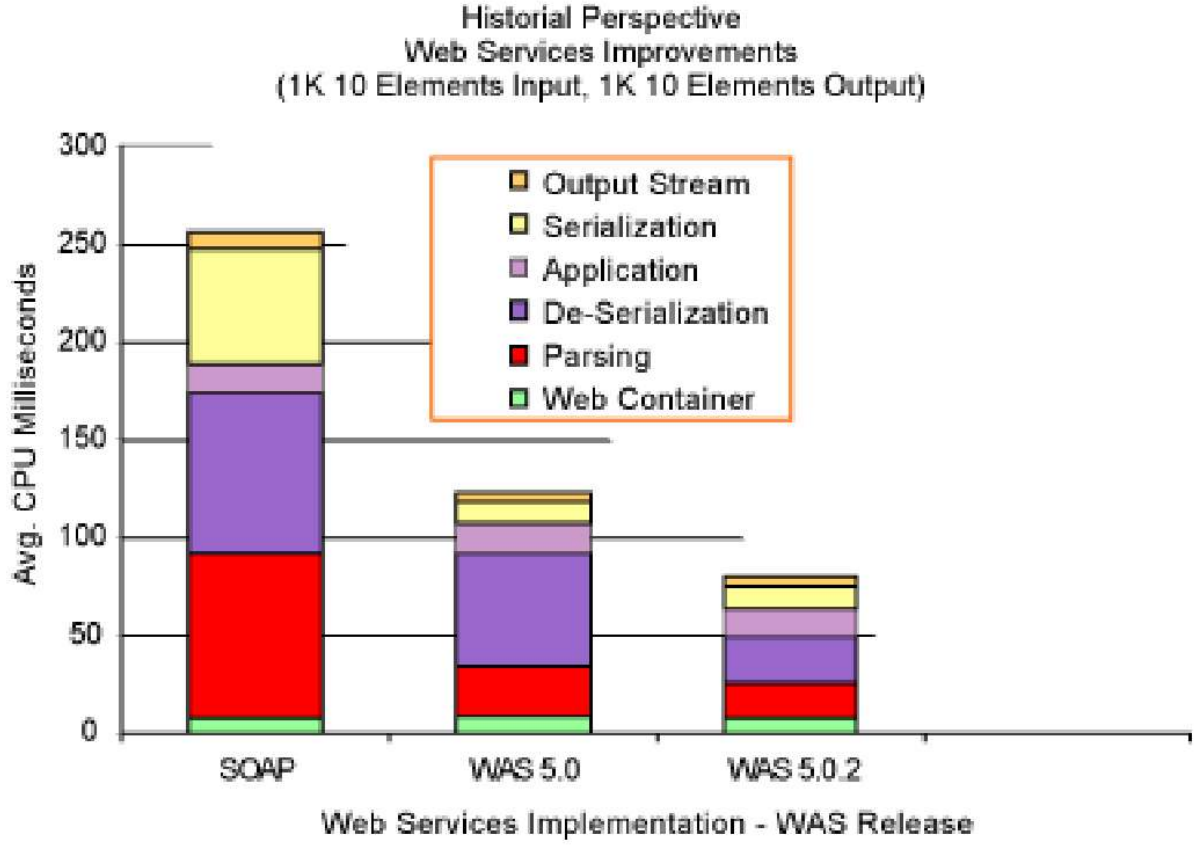
Web Services Performance

- New technologies are often marketed with grand promises to solve business problems, but ... PERFORMANCE?
- Performance criteria:
  - simultaneous requests,
  - latency in responding to requests,
  - levels of operational degradation due to increases in transaction loads.
  - ...

Web service component stack



Impact of parsing and marshalling



(source: http://www-106.ibm.com/developerworks/library/ws-best9)