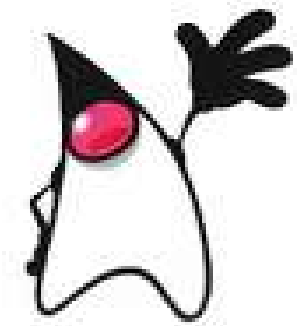
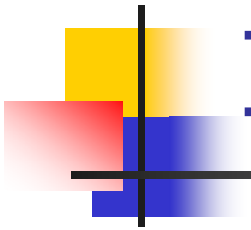
Introducción a los Servicios Web



Ingeniería del Software II
Curso 2008/2009

Sergio Ilarri Artigas
silarri@unizar.es



Índice

- Introducción
- Servicios Web SOAP
  - SOAP
  - WSDL
  - UDDI
  - APIs
- Servicios Web REST
- CORBA vs. Servicios Web

Introducción (I)

"In simple terms a Web Service is an application or business logic that is accessible using standard Internet protocols."

Fuente: http://faq.javaranch.com/java/WebServicesFaq

Introducción (II)

”Can I access a web service from any application?

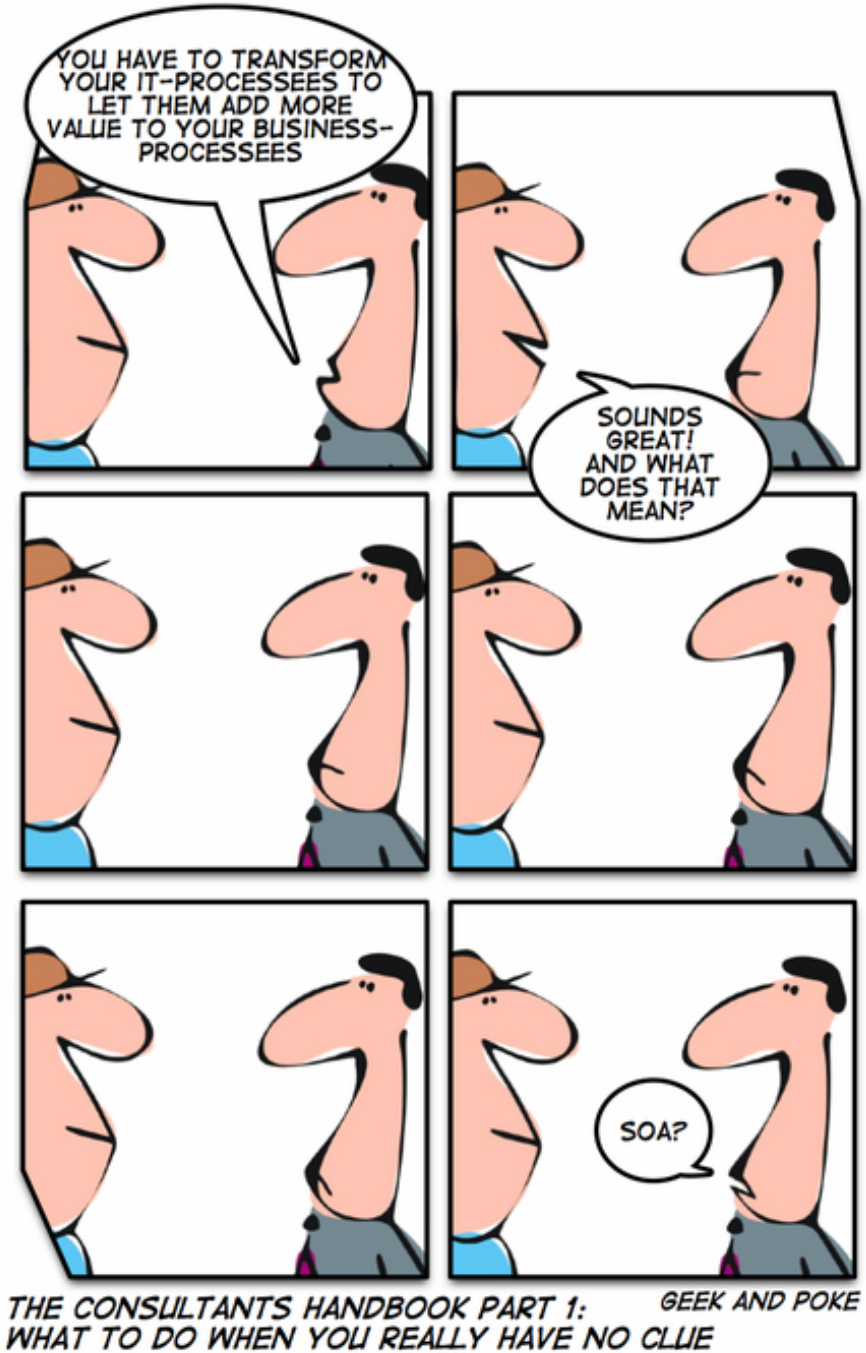
Yes, if your application supports XML based object request and response.”

Fuente: http://faq.javaranch.com/java/WebServicesFaq

Introducción (III)

Service-oriented architecture (SOA) is an evolution of distributed computing based on the request/reply design paradigm for synchronous and asynchronous applications. An application's business logic or individual functions are modularized and presented as services for consumer/client applications. What's key to these services is their loosely coupled nature; i.e., the service interface is independent of the implementation. Application developers or system integrators can build applications by composing one or more services without knowing the services' underlying implementations.

Fuente: http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html



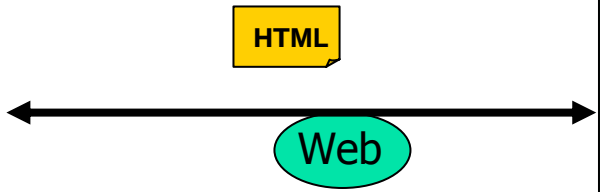
Fuente: http://geekandpoke.typepad.com/geekandpoke/2007/01/the_consultants.html

Introducción (IV)

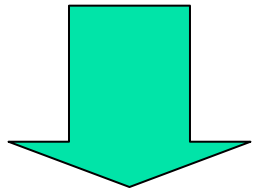
- Idea: utilizar la web como mecanismo de transporte "universal"
- Un servicio web proporciona una serie de puntos de acceso o ***endpoints*** que pueden invocarse remotamente
  - Accesibles a través de HTTP (normalmente)
  - Peticiones y respuesta en XML (normalmente)

Tecnologías web

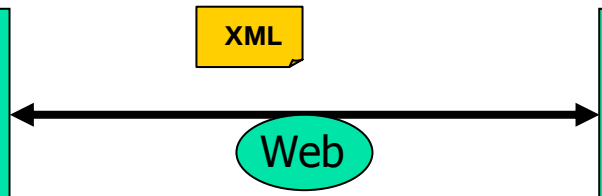
Introducción (V)



Aplicación Web



Aplicación Web



Servicio Web

Introducción (VI)

Mensajería

SOAP, XML

Descripción

WSDL, XML Schema

Descubrimiento

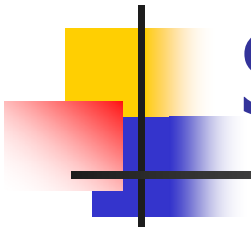
UDDI

Seguridad

TLS, SSL

Introducción (VII)

- Dos aproximaciones:
  1. Aproximación basada en *SOAP*
     - Estilo *RPC*
     - Tecnologías: *SOAP*, *WSDL*, *UDDI*
  2. Aproximación REST
     - *Representational Style Transfer*
     - Nuevo estilo arquitectónico
     - Tecnologías: *HTTP 1.1*, *XML*



SW SOAP: SOAP (I)

- *Simple Object Access Protocol* (inicialmente)
- Protocolo para el intercambio de mensajes (peticiones y respuestas) con estructura en XML
- Estándar del W3C: versiones 1.1 y 1.2
- Es sólo un formato de mensajes
- Métodos de transporte:
  - HTTP (normalmente; atraviesa *firewalls*)
  - SMTP
  - FTP
  - JMS (*Java Message Service*)
  - …

SW SOAP: SOAP (II)

- Servicios en SOAP:
  - Servicio: conjunto de puertos
  - Puerto: ofrece un conjunto de operaciones
  - Operaciones:
    - Nombre, parámetros (E, S, E/S), valor de retorno, *faults* (excepciones)
    - Síncronas (RPC) o asíncronas

SW SOAP: SOAP (III)

- SOAP estandariza la estructura de los mensajes de petición/respuesta:
  - *Envelope:* elemento raíz, *header+body*
  - *Header* (opcional)
  - *Body* (obligatorio): contenido del mensaje

SW SOAP: SOAP (IV)

Ejemplo de petición SOAP sobre HTTP

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Fuente: http://www.w3schools.com

SW SOAP: SOAP (V)

Ejemplo de respuesta SOAP sobre HTTP

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

Fuente: http://www.w3schools.com

SW SOAP: WSDL (I)

- *Web Services Description Language*
- Especificación en *XML* de las operaciones que ofrece un servicio web (su interfaz):
  - Puertos, operaciones, tipos de datos
- Versiones:
  - Versión 1.1 (WC3 Note, 2001)
  - Versión 2.0 (W3C Recommendation, 2007)

- Compilador de WSDL (APIs):
  - *Stub* en el cliente (patrón *Proxy*)
  - *Skeleton* en el servidor (patrón *Adapter*)
- En Axis: *Java2WSDL* y *WSDL2Java*
  (http://ws.apache.org/axis/java/user-guide.html)

Podríamos manejar los mensajes SOAP nosotros

SW SOAP: WSDL (II)

- Elementos básicos:
  - Definición de las operaciones (tipos de puertos): **<portType>** (≈ módulo de lenguaje de programación)
  - Definición de los mensajes (*input* y *output* de operaciones): **<message>**
    - "Partes"
      - Parámetros "In", "Out", e "InOut"
      - Valor de retorno
      - Excepción (*fault*)
  - Definición de los tipos de datos de los mensajes de petición y de respuesta (XML Schema, tipos SOAP): **<types>**
  - Definición de los protocolos de comunicación: **<binding>**
  - Definición de una dirección: **<port>**
  - Definición de servicios (conjuntos de puertos): **<service>**

SW SOAP: WSDL (III)

<definitions> Elemento raíz

<types> Tipos de datos que se transmiten

<message> Mensajes que se transmiten

<portType> Operaciones soportadas

<binding> Cómo se transmiten los mensajes por la red

<service> Dónde se localiza el servicio

SW SOAP: WSDL (IV)

1. *Service Definition Layer*
   - Tipos de datos
   - Tipos de mensajes
   - Operaciones
   - Servicios
2. *Binding Layer*
   - Protocolos de comunicación
   - Formatos de datos:
     - SOAP, HTTP, tipos MIME, etc.

Propiedades abstractas

Propiedades concretas

SW SOAP: WSDL (V)

Ejemplo

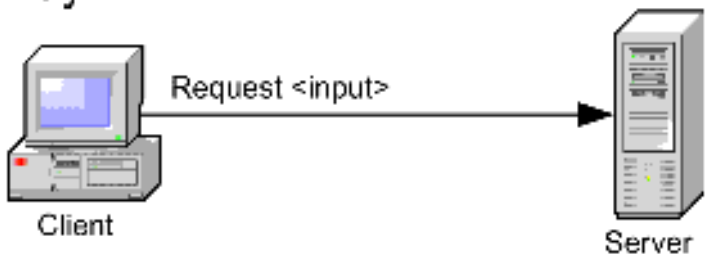
```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

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

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

Fuente: http://www.w3schools.com

SW SOAP: WSDL (VI)

- Tipos de operaciones (WSDL 1.1):

(1) One Way



Fuente: http://www2.roguewave.com/support/docs/leif/leif/html/webservicesug/5-2.html

SW SOAP: WSDL (VII)

Ejemplo *One-way*

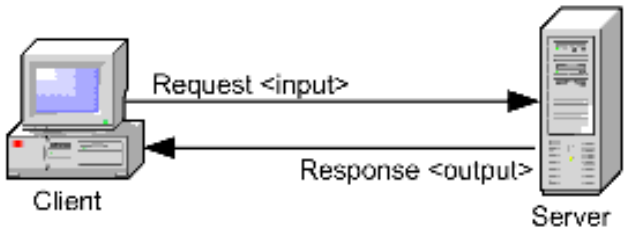
```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm"
      message="newTermValues"/>
  </operation>
</portType >
```

Fuente: http://www.w3schools.com

SW SOAP: WSDL (VIII)

- Tipos de operaciones (WSDL 1.1):

(2) Request-Response



Fuente: http://www2.roguewave.com/support/docs/leif/leif/html/webservicesug/5-2.html

SW SOAP: WSDL (IX)

Ejemplo *Request-Response*

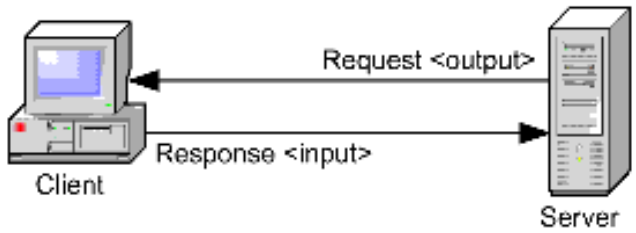
```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

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

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

Fuente: http://www.w3schools.com

SW SOAP: WSDL (X)

- Tipos de operaciones (WSDL 1.1):

(3) Solicit-Response



Fuente: http://www2.roguewave.com/support/docs/leif/leif/html/webservicesug/5-2.html

SW SOAP: WSDL (XI)

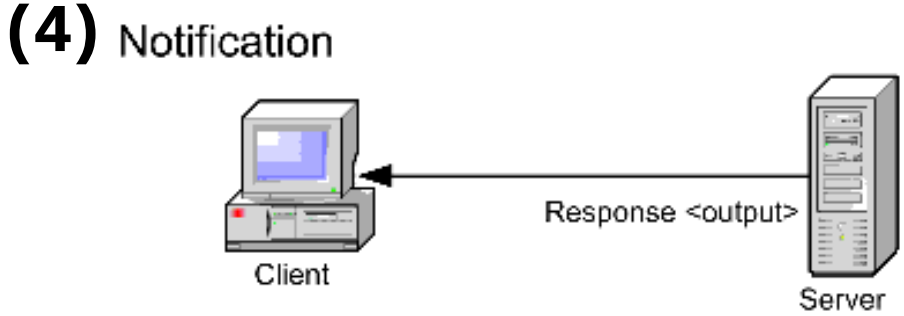
Ejemplo *Solicit-Response*

```
<message name="RenewResponse">
   <part name="RenewResponse" type="xsd:string"/>
</message>

 <message name="RenewRequest">
     <part name="RenewRequest" type="xsd:string"/>
  </message>

<portType name="weather">
  <operation name="weatherUpdateRenew">
     <output message="tns:RenewRequest"/>
     <input message="tns:RenewResponse"/>
  </operation>
</portType>
```

Fuente: http://www2.roguewave.com/support/docs/leif/leif/html/webservicesug/5-2.html

SW SOAP: WSDL (XII)

- Tipos de operaciones (WSDL 1.1):



Fuente: http://www2.roguewave.com/support/docs/leif/leif/html/webservicesug/5-2.html

SW SOAP: WSDL (XIII)

Ejemplo *Notification*

```
<message name="getSummaryResponse">
    <part name="weatherData" type="wsx:WeatherSummary"/>
  </message>

<portType name="weather">
   <operation name="weatherNotification">
      <output message="tns:getSummaryResponse"/>
   </operation>
</portType>
```

Fuente: http://www2.roguewave.com/support/docs/leif/leif/html/webservicesug/5-2.html

SW SOAP: WSDL (XIV)

- Patrones de intercambio de mensajes (MEPs) en WSDL 2.0:
  - In-Only
  - Robust In-Only
  - In-Out
  - In-Optional-Out
  - Out-Only
  - Robust Out-Only
  - Out-In
  - Out-Optional-In

http://www.w3.org/TR/2004/WD-wsdl20-extensions-20040803/#meps

SW SOAP: WSDL (XV)

- Un *binding* especifica el protocolo y formato de datos para un puerto (hijos *soap:binding* y *operation*)
- Atributos del elemento *binding*
  - *Name* : el nombre que se da al *binding*
  - *Type* : referencia al puerto para el *binding*
- Atributos del nodo hijo *soap:binding*
  - *Style* : *"rpc"* o *"document"*
  - *Transport* : protocolo SOAP (Ej., HTTP, SMTP, FTP, JMS)
- Nodo hijo *operation*
  - Define las operaciones que expone el puerto
  - Para cada operación, se indica la acción SOAP
  - Además, se debe especificar cómo se codifica la entrada y la salida (ej., "literal")

Usando más de un puerto, es posible ligar un *portType* a más de un protocolo

SW SOAP: WSDL (XVI)

Ejemplo de binding

```
<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>
```

Fuente: http://www.w3schools.com

SW SOAP: WSDL (XVII)

Ejemplo de definición de un servicio

```
<service name="StockQuoteService">
    <port name="StockQuotePort" binding="tns:StockQuoteSoap">
      <soap:address location="mailto:subscribe@example.com"/>
    </port>
</service>
```

endpoint

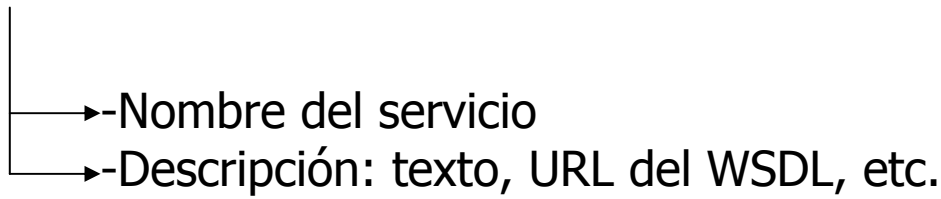
Otro ejemplo

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

Fuente: http://www.w3.org/TR/wsdl#_soap-b

SW SOAP: UDDI

- *Universal Description, Discovery and Integration* (directorio de servicios web)
- *OASIS* estándar (http://www.oasis-open.org)
- Interfaz SOAP (WSDL) de un servicio, el *Registro UDDI:*
  - Proporciona información acerca de servicios web disponibles (búsqueda por nombre, descripción, etc.)
  - Ofrece operaciones para registrar servicios web



SW SOAP: APIs

- Necesitamos un API para el lenguaje que utilicemos:
  - Java
    - Estandarizados:
      - *JAX-M* (*Java API for XML Messaging*)
      - *JAX-RPC* (*Java API for XML-based RPC*)
      - *JAXR* (*Java API for XML Registries*)
      - *JAX-WS* (*Java API for XML Web Services*)
  - Lenguajes de Microsoft: .NET

SW SOAP: JAX-RPC 1.1

- *The Java API for XML-based RPC*
- Basado en SOAP (vs. XML-RPC)
- Parte del API de *Java Enterprise Edition*
  - Paquete *javax.xml.rpc* (interfaces)
- Múltiples implementaciones:
  - Ej.: Apache Axis / Axis 2 (http://ws.apache.org/axis/)
- Especifica dos tipos de *mappings* :
  - Compiladores de (interfaz) Java a WSDL
  - Compilador de WSDL a Java: generan los *stubs* y *skeletons*
- Sucesor: *JAX-WS 2.0*

http://www.ibm.com/developerworks/webservices/library/ws-tip-jaxwsrpc.html

SW REST (I)

- *Representational Style Transfer*
- Propuesto por Roy Fielding (2000)
- Protocolo de transporte: HTTP
- El programador debe analizar los mensajes resultado en XML
- No utiliza RPC, no utiliza SOAP
- No precisa nuevos protocolos ni lenguajes: sólo HTTP 1.1 y XML
- Idea clave: cada petición tiene toda la información necesaria (servicios sin estado)
- Ejemplo: RSS/ATOM

SW REST (II)

- HTTP:
  - *Hypertext Transfer Protocol*
  - Estándar de W3C e IETF (versión actual: HTTP 1.1)
  - Identifica **recursos** mediante *URLs*
  - Petición HTTP (manipulación de recursos):
    - URL y método de acceso
      - *GET*, *POST*, *PUT*, *DELETE* (semántica definida)
    - Cabeceras
    - Cuerpo del mensaje (opcional)
  - GET no debe tener efectos secundarios

http://www.w3.org/Protocols/rfc2616/rfc2616.html

SW REST (III)

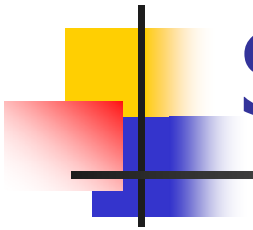
- En general, instalamos las aplicaciones web en *contenedores (servidores) de aplicaciones*. Ejemplos:
  - *Oracle WebLogic Application Server*
  - *IBM Websphere Application Server*
  - *BEA WebLogic Server*
  - *Sun GlassFish Enterprise Server*
  - *JBoss Application Server*
  - *Apache Tomcat* (servlets y JSPs)
  - *Jetty* (servlets)
  - *Geronimo*

Código abierto

SW REST (IV)

- Un aplicación web se encapsula en un *fichero WAR* (*deployable*):
  - *Web Application Archive* (.war)
  - Fichero JAR con una estructura estándar
    - WEB-INF/lib, WEB-INF/classes, WEB-INF/web.xml
    - Ficheros HTML, JSP, imágenes, etc.
    - Acceso a WEB-INF sólo para servlets y JSPs
- APIs estándar para aplicaciones web
- Para REST, podemos utilizar *Servlets*

SW REST: REST vs. RPC (I)

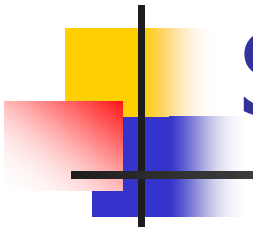


- REST ofrece un espacio de nombres global (URLs): fácil reutilizar recursos
- REST ofrece un conjunto delimitado de operadores con cierta semántica:
  - Por tanto, es menos flexible que RPC
  - Pero al tener estos operadores una semántica asociada se facilitan algunas tareas:
    - Se pueden repetir peticiones GET fallidas sin riesgo (son idempotentes)
    - Un recurso accedido mediante PUT, DELETE o POST puede ser invalidado automáticamente por un servidor de cache

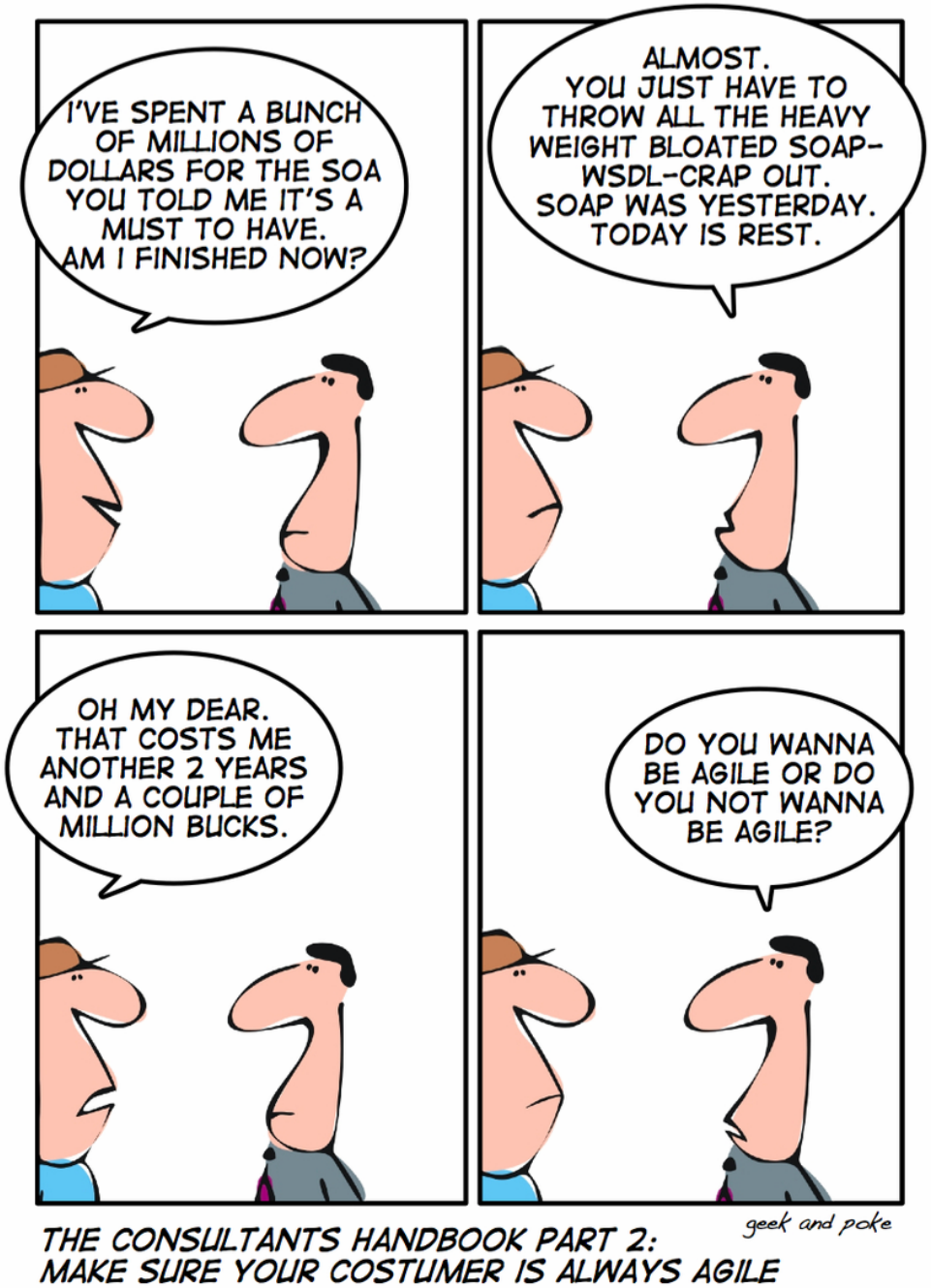
SW REST: REST vs. RPC (II)

- REST es más "accesible" (URLs y HTTP) que RPC:
  - RPC: librerías específicas y protocolo concreto
    - Acceder a un recurso implica implementar un programa que lo accede
  - REST: URLs y HTTP
    - Acceder a un recurso utilizando un navegador
    - Pero el cliente debe realizar las peticiones HTTP y analizar los documentos devueltos "a mano"
- "Acuerdos" para el uso:
  - En RPC hay que conocer el API
  - En REST hay que conocer el vocabulario XML

SW REST: REST vs. RPC (III)

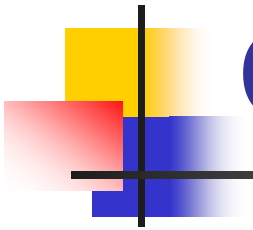


- Reutilización de mecanismos:
  - Con REST es posible reutilizar los mecanismos de seguridad de HTTP
    - Pero hay otros servicios que no proporciona HTTP (transacciones distribuidas, etc.)
  - RPC debe proporcionar mecanismos específicos
- Protocolo de transporte:
  - Con RPC pueden utilizarse distintos protocolos: HTTP, JMS, SMTP, etc.
  - HTTP puede ser menos eficiente y no garantiza la entrega de mensajes (sí JMS)



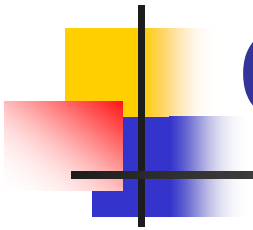
Fuente: http://geekandpoke.typepad.com/geekandpoke/2007/06/the_consultants.html

CORBA vs. Servicios Web (I)



- CORBA es una arquitectura orientada a objetos (y fuertemente acoplada)
- IIOP es más eficiente que SOAP
  - IIOP es binario, SOAP utiliza XML
- Con SOAP, pueden surgir algunos problemas de interoperabilidad (superados con IIOP)
- SOAP normalmente utiliza HTTP, "atravesando firewalls"

CORBA vs. Servicios Web (II)



- IDL es más fácil de leer que WSDL
- Los objetos CORBA pueden tener estado
- No hay un equivalente al POA en SW
- En ambos, los clientes pueden utilizar proxies o invocaciones dinámicas
- El API CORBA está estandarizado pero el de SW no completamente

CORBA vs. Servicios Web (III)

- CORBA viene con un conjunto estándar de servicios:
  - CORBA's *Naming Service* y *Trading Service* ~ UDDI
  - *Notification Service* y *Event Service*, uso de objetos *callback*
    - En WS: no se pueden manejar referencias a puertos
  - *Transaction Service*

CORBA vs. Servicios Web (IV)

- Integración de aplicaciones:
  - En Internet: servicios web
  - En intranets: CORBA
- CORBA (1991): más madura y completa
  - No apoyo de Microsoft
- Servicios Web (2000): más moderna

CORBA vs. Servicios Web (V)

Aspect	CORBA	Web services
Data model	Object model	SOAP message exchange model
Client-Server coupling	Tight	Loose
Location transparency	Object references	URL
Type system	IDL	XML schemas
	static + runtime checks	runtime checks only
Error handling	IDL exception	SOAP fault messages
Serialization	built into the ORB	can be chosen by the user
Parameter passing	by reference	by value (no notion of objects)
	by value (*valuetype*)	
Transfer syntax	CDR used on the wire	XML used on the wire
	binary format	Unicode
State	stateful	stateless
Request semantics	at-most-once	defined by SOAP
Runtime composition	DII	UDDI/WSDL
Registry	Interface Repository	UDDI/WSDL
	Implementation repository	
Service discovery	CORBA naming/trading service	UDDI
	RMI registry	
Language support	any language with an IDL binding	any language
Security	CORBA security service	HTTP/SSL, XML signature
Firewall Traversal	work in progress	uses HTTP port 80
Events	CORBA event service	N/A

Fuente: A. Gokhale, B. Kumar, A. Sahuguet. Reinventing the Wheel? CORBA vs. Web Services. In WWW2002 Conference Proceedings, 2002. http://www2002.org/CDROM/alternate/395/

CORBA vs. Servicios Web (VI)

CORBA stack	Web Services stack
IDL	WSDL
CORBA Services	UDDI
CORBA Stubs/Skeletons	SOAP Message
CDR binary encoding	XML Unicode encoding
GIOP/IIOP	HTTP
TCP/IP	TCP/IP

Fuente: A. Gokhale, B. Kumar, A. Sahuguet. Reinventing the Wheel? CORBA vs. Web Services. In WWW2002 Conference Proceedings, 2002. http://www2002.org/CDROM/alternate/395/

Referencias (I)

- WSDL:
  - Especificación 1.1: http://www.w3.org/TR/wsdl
  - Especificación 2.0: http://www.w3.org/TR/wsdl20/
- SOAP:
  - Especificación (1.1 y 1.2): http://www.w3.org/TR/soap/
- UDDI:
  - Especificación (2.0 y 3.0.2): http://uddi.xml.org/
- W3Schools Web Services Tutorial: http://www.w3schools.com/webservices/default.asp

Referencias (II)

- Tutoriales:
  - The Java Web Services Tutorial: http://java.sun.com/webservices/docs/1.6/tutorial/doc/
  - W3Schools SOAP Tutorial: http://www.w3schools.com/soap/default.asp
  - W3Schools WSDL Tutorial: http://www.w3schools.com/WSDL/default.asp
    - WSDL and UDDI: http://www.w3schools.com/WSDL/wsdl_uddi.asp
  - Getting Started with JAX-RPC: http://java.sun.com/developer/technicalArticles/WebServices/getstartjaxrpc/

Referencias (III)

- Cursos de interés:
  - "CSCI 7818: Web Services", Kenneth M. Anderson, Dennis Heimbigner, University of Colorado at Boulder (CO, USA): http://www.cs.colorado.edu/~kena/classes/7818/f06/index.html
  - "Análisis y Diseño Orientado a Objetos", Fernando Bellas Permuy, Universidad de A Coruña: http://www.tic.udc.es/~fbellas/teaching/adoo/index.html
- Artículos de interés:
  - IBM SOA and Web services: http://www.ibm.com/developerworks/webservices
  - "Reinventing the Wheel? CORBA vs. Web Services". A. Gokhale, B. Kumar, A. Sahuguet. In WWW2002 Conference Proceedings, 2002. http://www2002.org/CDROM/alternate/395/

Agradecimientos: Parte de esta presentación se apoya en información disponible para los dos primeros cursos de arriba. Mis agradecimientos por el excelente material que ofrecen.

Referencias (IV)

- APIs:
  - JAX-RPC Reference Implementation:
    https://jax-rpc.dev.java.net/
  - Java API for XML Registries (JAXR)
    http://java.sun.com/webservices/jaxr/index.jsp
  - JAX-WS Reference Implementation:
    https://jax-ws.dev.java.net/

Referencias (V)

- Protocolos:
  - HTTP - Hypertext Transfer Protocol: http://www.w3.org/Protocols/
  - JMS - Java Message Service: http://java.sun.com/products/jms/
  - SMTP - Simple Mail Transfer Protocol (RFC821): http://www.ietf.org/rfc/rfc2821.txt

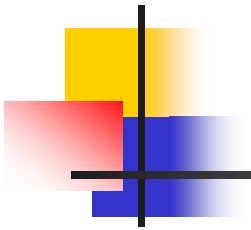
Referencias (VI)

- REST (y elementos útiles relacionados):
  - HTTP - Hypertext Transfer Protocol: http://www.w3.org/Protocols/
  - R. T. Fielding, "Architectural Styles and the Design of Network-Based Software Architectures", Doctoral Thesis. University of California, Irvine, 2000. Available at: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
  - Java Servlet Technology: http://java.sun.com/products/servlet/
  - JDOM: http://www.jdom.org/
  - Apache Maven Project: http://maven.apache.org/
  - Restlet (Lightweight REST framework for Java): http://www.restlet.org/
  - JSR 311 - JAX-RS: The JavaTM API for RESTful Web Services: http://jcp.org/en/jsr/detail?id=311

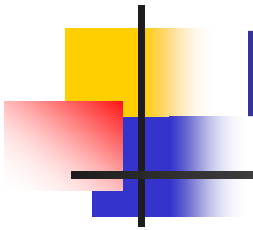
Referencias (VII)

- Servidores de aplicaciones:
  - Oracle WebLogic Application Server:
    - http://www.oracle.com/appserver/index.html
  - IBM Websphere Application Server:
    - http://www-01.ibm.com/software/webservers/appserv/was/
  - BEA WebLogic Server:
    - http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic/server
  - Sun GlassFish Enterprise Server:
    - http://www.sun.com/software/products/appsrvr/
  - JBoss Application Server: http://www.jboss.org/
  - Apache Tomcat: http://tomcat.apache.org/
  - Jetty: http://www.mortbay.org/jetty/
  - Geronimo: http://geronimo.apache.org/

(The Server Side Application Server Matrix: http://www.theserverside.com/tt/articles/content/ServerMatrix/matrix_print.html)



¿ Y qué pasa con los llamados *Servicios Web Semánticos* ?



Fin

Gracias por vuestra atención

Ésta es una primera versión de estas transparencias. Por favor, si detectas cualquier error o tienes sugerencias, contacta conmigo.

Sergio Ilarri