

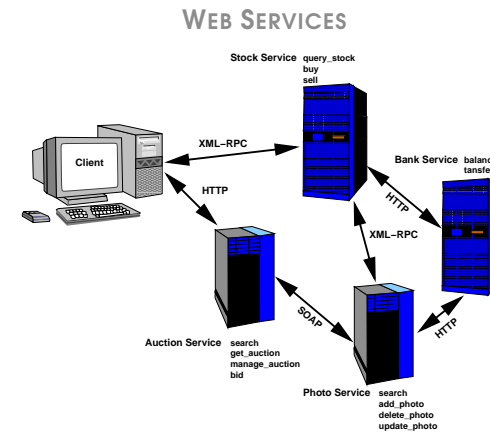
DISTRIBUTED SYSTEMS (COMP9243)

Lecture 4b: Web Services aka BUZZWORD and HYPE

Slide 1

- ① SOA
- ② SOAP, XML-RPC, REST
- ③ WSDL, UDDI
- ④ Web 2.0
- ⑤ Mashups
- ⑥ Problems and Research

Slide 3



WHAT'S A WEB SERVICE?

HYPE!

But really:

Slide 2

Services provided over the Web

Except it doesn't really require the Web

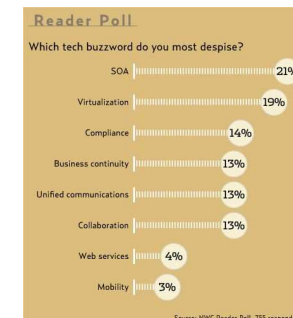
So...

A software system designed to support interoperable machine-to-machine interaction over a network.
(W3C)

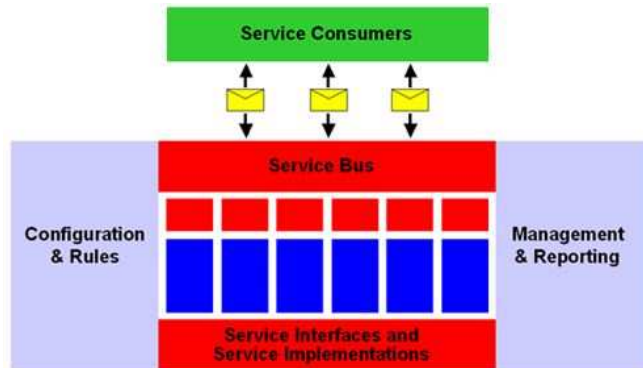
SOA: SERVICE ORIENTED ARCHITECTURE

MORE HYPE!

Slide 4



Slide 9



Slide 10

SERVICE ORIENTED ARCHITECTURE

SOA is a design for linking business and computational resources (principally organizations, applications and data) on demand to achieve the desired results for service consumers (which can be end users or other services). – wikipedia

A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations. – OASIS

SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners. – webservices.xml.com

Service Oriented Architecture is, after all, much more of a mind set, development philosophy, and architecture than it is anything concrete and definitive. – bobbreedlove.com

SOA PROPERTIES

Interfaces:

- Simple
- Generic
- Interface description, service contract

Messages:

- Descriptive (vs prescriptive)
- Structured

Loose Coupling:

- Minimized dependencies
- Interface independent of implementation

Service Discovery

Slide 11

WEB SERVICES VS SOA?

Are Web Services Service Oriented Architecture?

Scale:

- Web services: global scale
- SOA: typically local scale

Protocols:

- Internet/Web protocols
- HTTP is most common

Messages:

- XML-based

WEB SERVICES TECHNOLOGIES

Communication:

- XML-RPC
- SOAP
- REST

Slide 13

Interfaces:

- WSDL

Service Discovery:

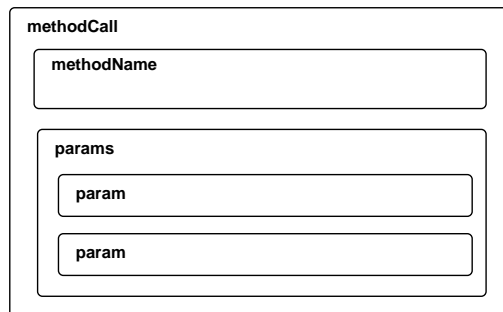
- UDDI

XML-RPC

XML-RPC:

- XML request/reply (no schemas)
- HTTP POST

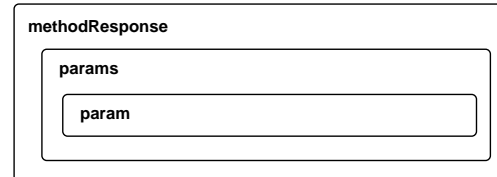
Slide 14



Request:

```
<methodCall>
<methodName>flickr.test.echo</methodName>
<params>
  <param>
    <value>
      <struct>
        <member>
          <name>name</name>
          <value><string>value</string></value>
        </member>
        <member>
          <name>name2</name>
          <value><string>value2</string></value>
        </member>
      </struct>
    </value>
  </param>
</params>
</methodCall>
```

Slide 15



Response:

Slide 16

```
<methodResponse>
<params>
  <param>
    <value>
      <string>
        &lt;method&gt;flickr.test.echo&lt;/method&gt;
        &lt;name&gt;value&lt;/name&gt;
      </string>
    </value>
  </param>
</params>
</methodResponse>
```

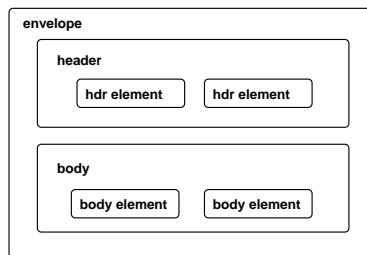
SOAP – SIMPLE OBJECT ACCESS PROTOCOL

The most-misnamed technology of all time

Prediction for new name:

Service Oriented Architecture Protocol

Slide 17



Request:

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
  <s:Body>
    <x:FlickrRequest xmlns:x="urn:flickr">
      <method>flickr.test.echo</method>
      <name>value</name>
    </x:FlickrRequest>
  </s:Body>
</s:Envelope>
```

Slide 18

Response:

```
<s:Envelope
  xmlns:s="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
  <s:Body>
    <x:FlickrResponse xmlns:x="urn:flickr">
      <method>flickr.test.echo</method>
      <name>value</name>
    </x:FlickrResponse>
  </s:Body>
</s:Envelope>
```

Slide 19

IS SOAP ANYTHING NEW?

SOAP vs RPC:

Slide 20 Is SOAP an extremely bloated RPC technology?

- ✓ includes RPC style communication
- ✗ not limited to RPC (async, documents, events)
- ✗ RPC is prescriptive/instructive rather than descriptive

Slide 21

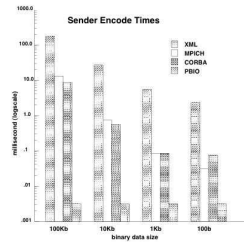


Figure 2. Send-side encoding times.

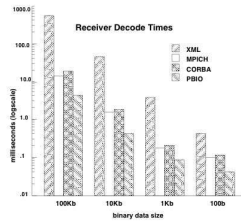


Figure 3. Receive side decode times.

From: "Efficient Wire Formats for High Performance Computing", Bustamante et al.

Slide 22

SOAP vs Distributed Objects:

Is SOAP a bloated and inefficient Distributed Object Middleware?

- ✓ SOAP is "Object Access Protocol"
- ✗ SOAP doesn't include instantiation of objects
- ✗ No remote object references to be passed
- ✗ Loose coupling

Slide 23

SOAP vs Message-Oriented Middleware?:

Is SOAP a limited MOM?

- ✓ allows document-based asynchronous communication
- ✓ doesn't provide message routing
- ✓ doesn't provide persistence
- ✗ also provides for RPC

Slide 24

REST – REPRESENTATIONAL STATE TRANSFER

Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use. – Roy Fielding

REST is:

→ Resources:

- nouns
- referenced by URIs
- different possible representations
- contain references to other resources

→ Operations (HTTP)

- GET: get representation of resource
- PUT, POST: overwrite or update a resource
- DELETE: remove a resource

→ Stateless interaction between requests

Example:

<http://api.flickr.com/services/rest/?method=flickr.photos.getFavorites&photoID=XXX>

✗ not really REST...

<http://api.flickr.com/services/rest/favorites/XXX>

Slide 25

```

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

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

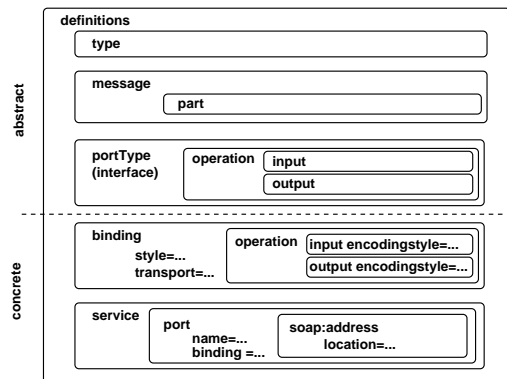
<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<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>

```

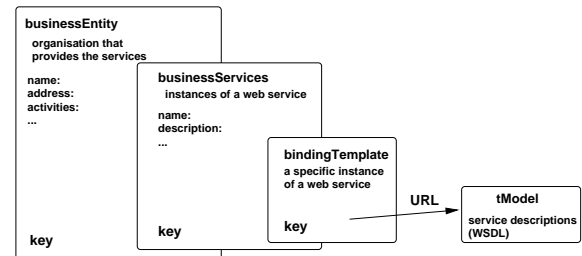
Slide 27

WSDL – WEB SERVICES DESCRIPTION LANGUAGE



Slide 26

UDDI – UNIVERSAL DIRECTORY AND DISCOVERY



Slide 28

WEB 2.0 – WEB SERVICES IN ACTION?

Even More HYPE!

Slide 29 Web 2.0 =

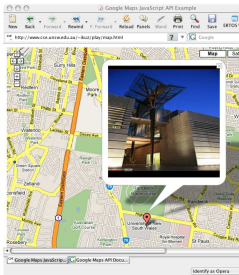
- AJAX: Asynchronous Javascript and XML
- Web Services
- participation: blogs, RSS, social networking, wiki, tagging, etc.
- list keeps growing

MASHUPS

Combine different Web Services together on separate, independent site.

Web Services become mainstream:

Slide 30



The code:

```
var photos;  
function jsonFlickrApi(rsp){  
    photos = rsp.photos.photo;  
}
```

Slide 31

...

```
<script type="text/javascript" src="http://api.flickr.com/services/  
rest/?method=flickr.photos.search&tags=unsw&format=json&api_key=XXX">  
</script>
```

```
function load() {  
    var map = new GMap2(document.getElementById("map"));  
    map.addControl(new GSmallMapControl());  
    map.addControl(new GMapTypeControl());  
    var geocoder = new GClientGeocoder();  
    var marker;
```

Slide 32

```
var address = "University of New South Wales";  
geocoder.getLatLng(address,  
    function(point) {  
        map.setCenter(point, 13);  
        marker = new GMarker(point);  
        map.addOverlay(marker);  
    } );  
...
```

```

...
GEvent.addListener(marker, "click", function() {
    marker.openInfoWindowHtml("");
    });
}

function newPhoto() {
    var i = Math.round(photos.length*Math.random());
    var farmid=photos[i].farm;
    var serverid=photos[i].server;
    var id=photos[i].id;
    var secret=photos[i].secret;
    return "http://farm"+farmid+".static.flickr.com/"+serverid+
        "/" +id+"_"+secret+"_m.jpg";
}

```

Slide 33

PROBLEMS OF WEB SERVICES AND WEB 2.0

Popularity:

- Everyone offers services
- Everyone is combining services
- Most of the WS infrastructure isn't used

Slide 34

Ignoring...:

- Previous work in distributed systems
- Scalability
- Synchronisation
- Fault tolerance
- Security

SCALABILITY

Scalability of individual services:

- Cluster scalability
- Limit usage (e.g., Google approach)

Slide 35

Scalability of compositions/Mashups:

- Size: data size and SOAP
- Dependencies: dependence on other services, latency hiding
- Geographic: Caching? (REST is good for caching)
- Replication: same issues as replicating dynamic Websites
- Administrative: ???

SYNCHRONISATION AND COORDINATION

Synchronisation within a service:

- implement required protocols
- tight coupling

Slide 36

Synchronisation of compositions/Mashups:

- Order in which to invoke services
- WS-Coordination - distributed transactions
- Choreography: WSDL-based coordination language
 - constraints on message ordering
 - contract between participants

FAULT TOLERANCE

Fault Tolerance of individual services:

- Replication/Redundancy
- Checkpoints/Rollbacks
- Transactions
- Essentially same as for current Web servers

Slide 37

Fault Tolerance of compositions/Mashups:

- Partial Failure in all its glory
 - Stateless services
 - Support for Transaction Services. Replication Services
 - Complex calling graphs - makes problems more severe
 - problem of replicated invocations
-

SECURITY

Securing communication channel:

- TLS/SSL

Securing the services:

- Firewalls
- Authentication Services
- Authorisation?

Slide 38

Securing the messages:

- Different services may handle the message
 - Each needs only some data
 - XML-Security
 - encrypt parts of messages
-

Securing the client:

- Downloading code from different services
- e.g., javascript code from Google, Flickr, etc.
- Running multiple mashups in single browser
- e.g., bank site and gambling page

Slide 39
