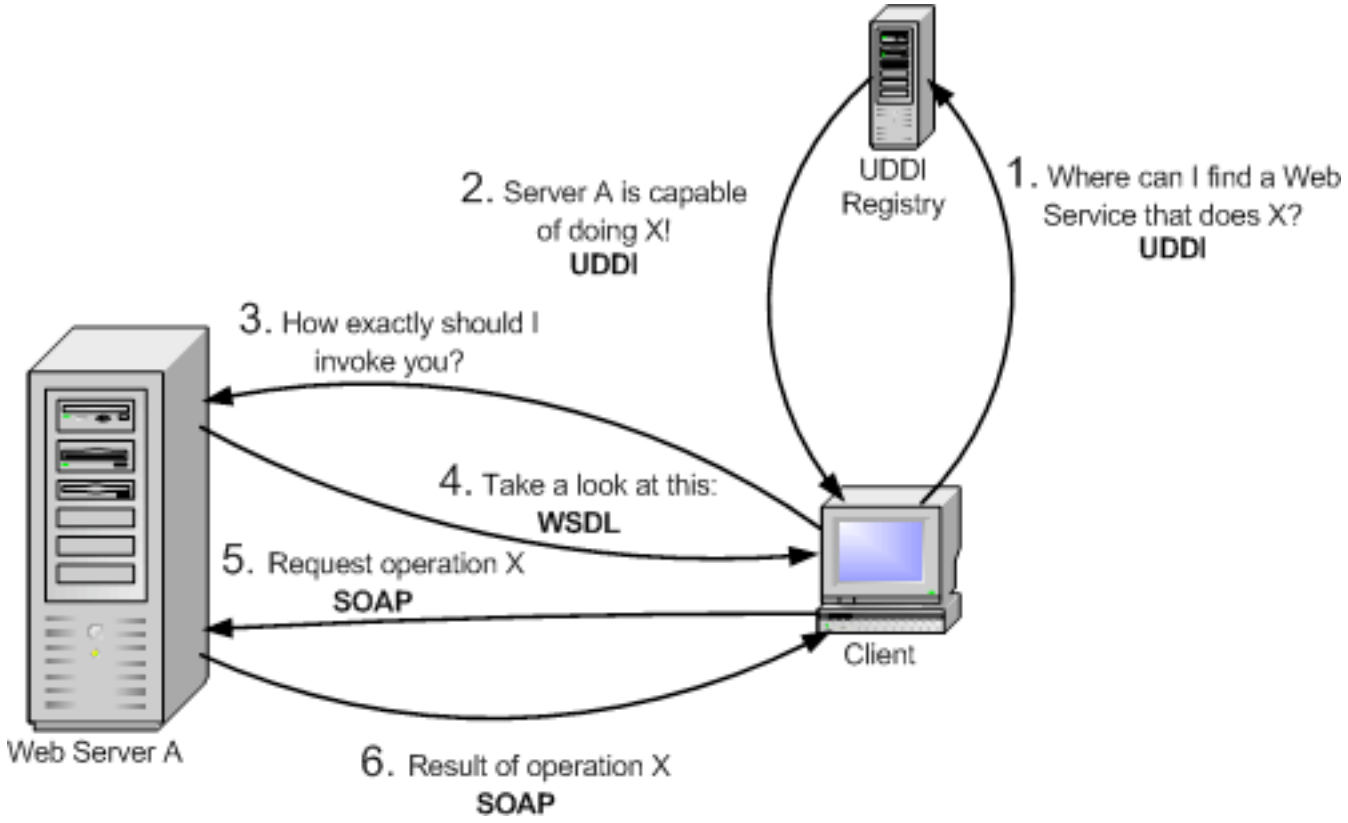
95-702 Distributed Systems

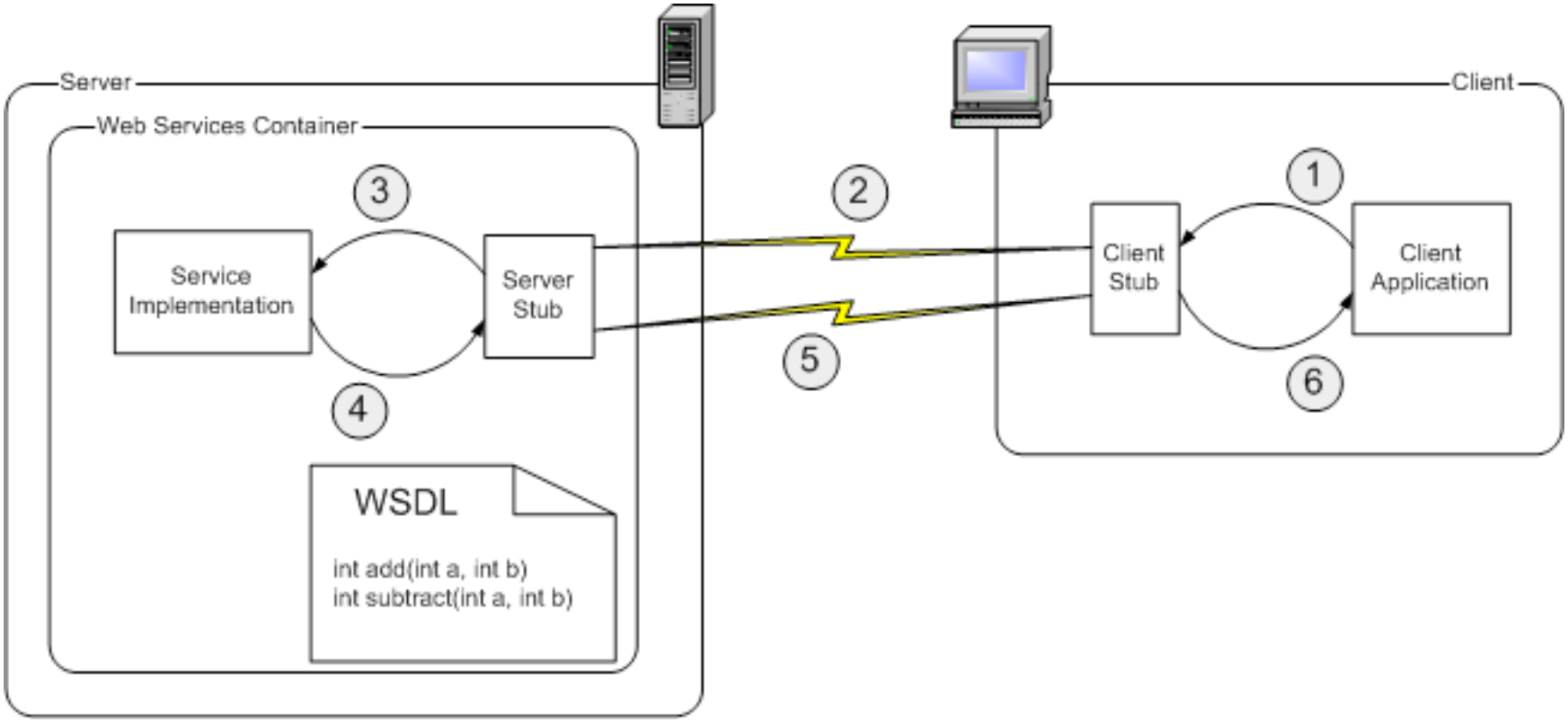
Lecture 4: Web Services
Chapter 19 of Coulouris

In A Nutshell



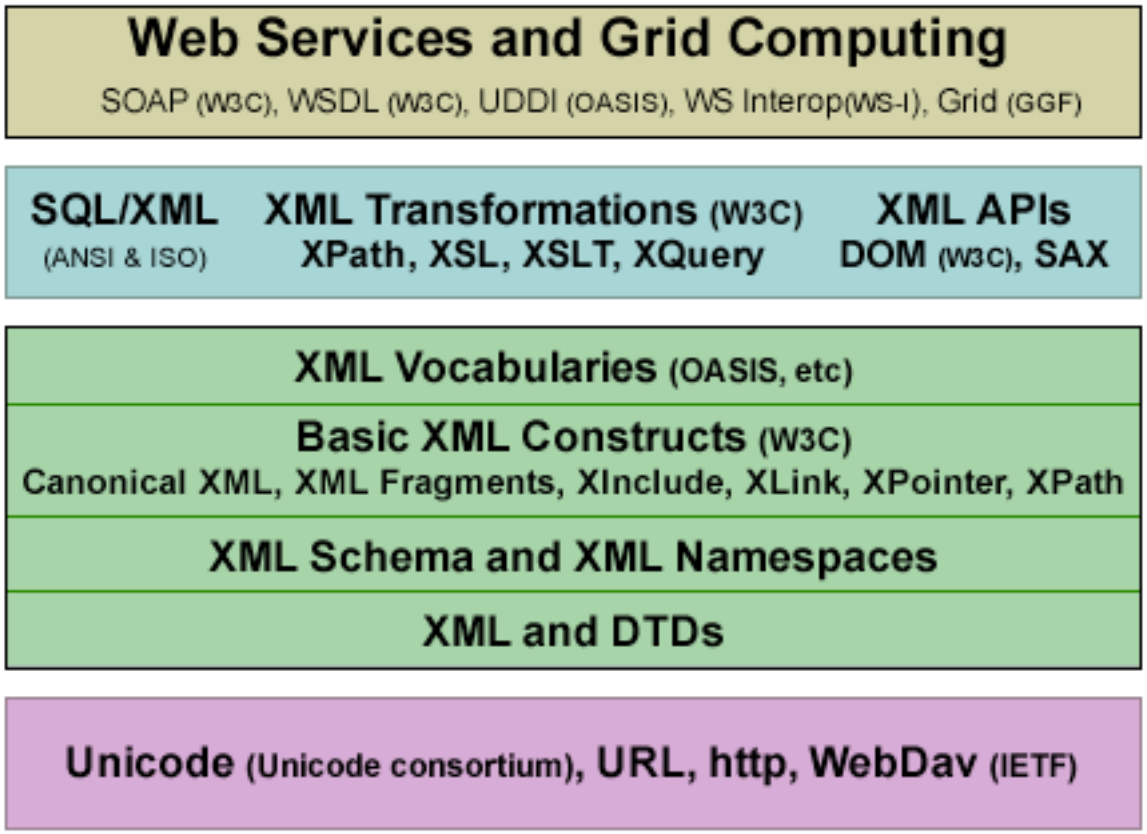
From Globus.org
(Grid computing)

With Stubs



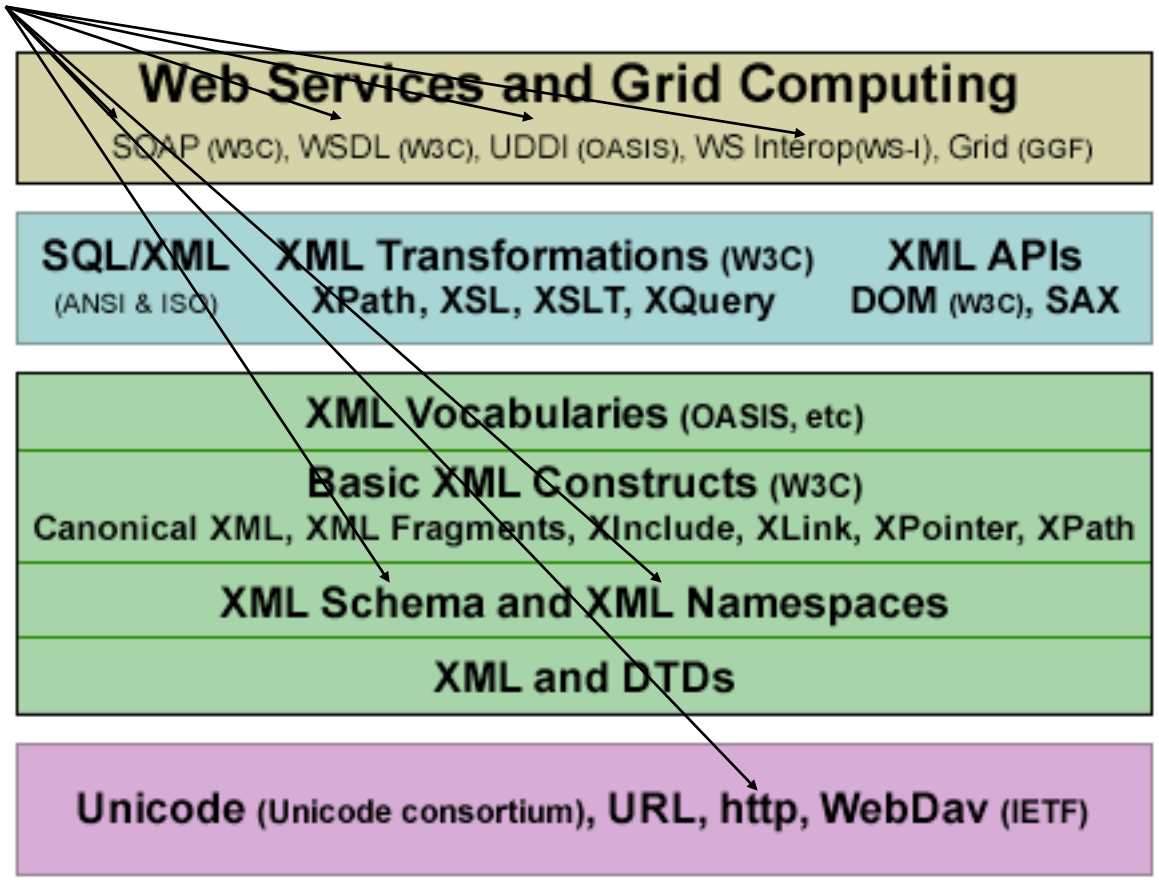
From Globus.org
(Grid computing)

Some Important Standards



Some Important Standards

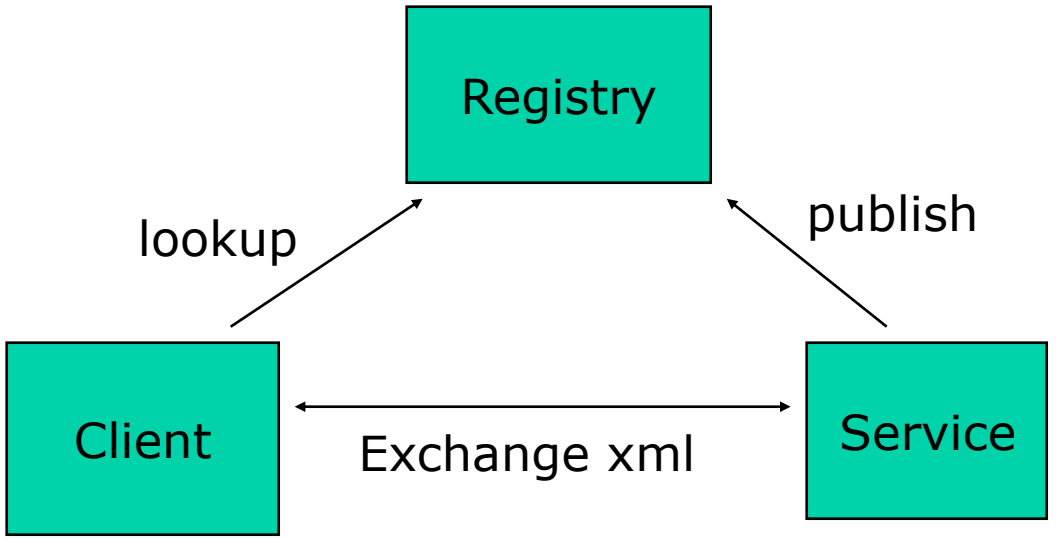
Very important with respect to XML web services.



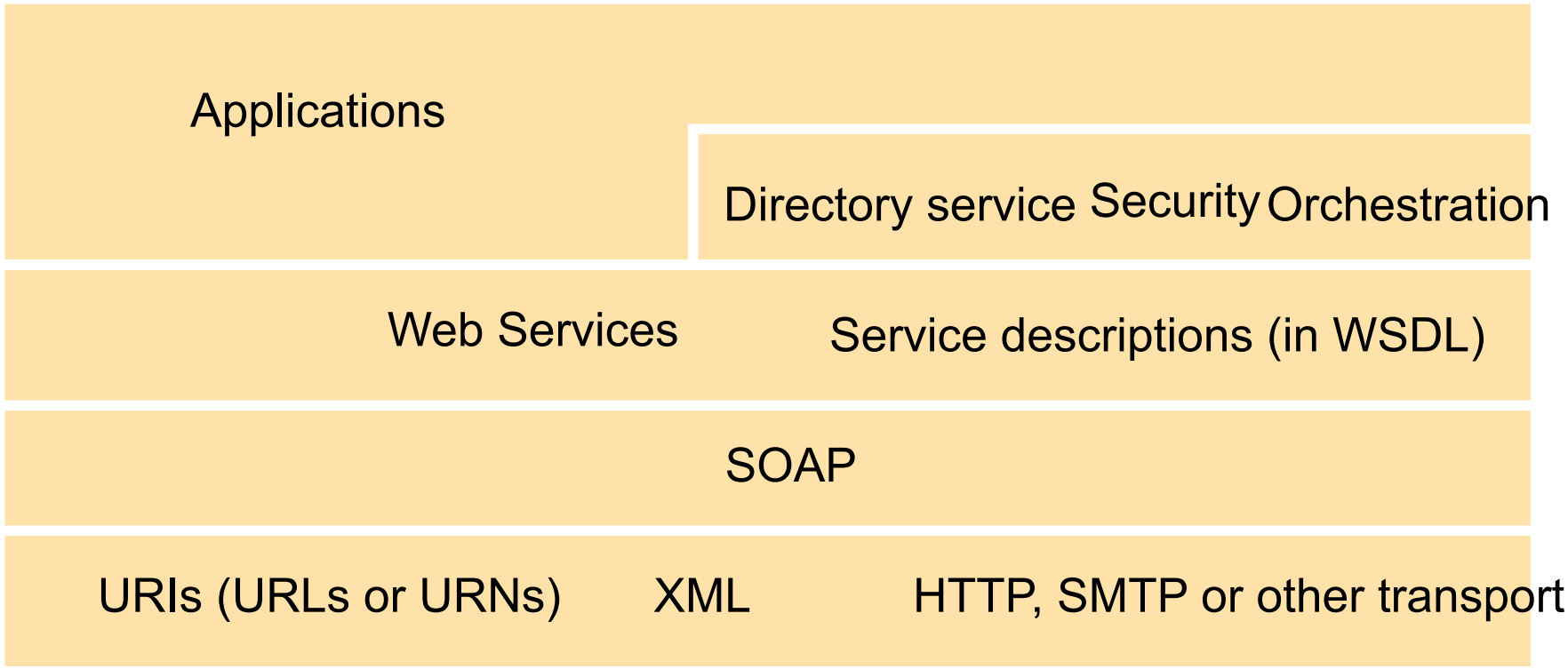
Web Services

- Provide service interfaces.
- Communicate using request and reply messages made of SOAP or some other XML document.
- Have an Interface Definition Language (IDL) called WSDL (Web Service Definition Language)
- May be looked up in a web service UDDI registry (Universal Directory and Discovery Service).
- Are language independent.
- May be synchronous or asynchronous.

Web Services



Web Services Infrastructure and Components



Communication Patterns

- In general, web services use either a **synchronous request-reply** pattern of communication with their clients or they communicate by **asynchronous messages**.
- The client does not block on asynchronous calls. Do you block when you are expecting an important phone call? If not then you are planning on handling the call asynchronously.
- To allow for a variety of patterns, SOAP is based on the packaging of single one-way messages.
- SOAP is used to hold RPC style parameters or entire documents.
- SOAP may be used over different transports (SMTP, TCP, UDP, or HTTP)

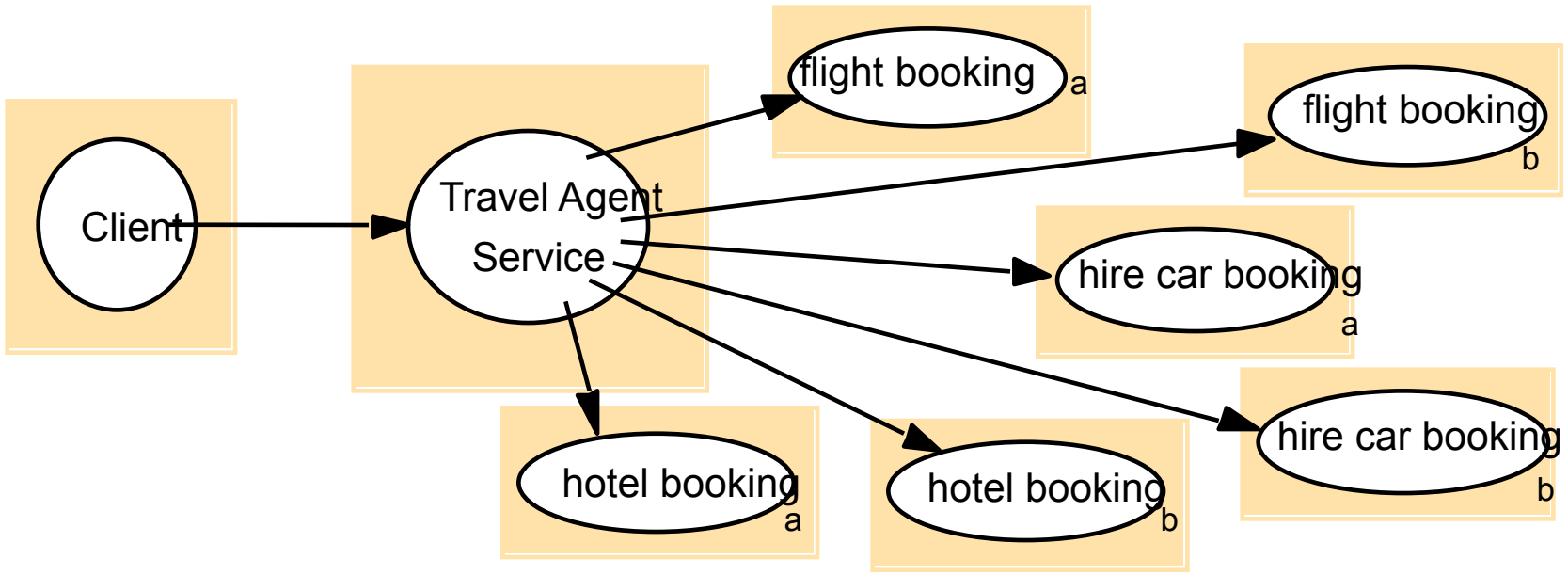
Service References

- URI's are **Uniform Resource Identifiers**.
- URL's are **Uniform Resource Locator** URI's that include location information. Thus, resources pointed to by URL's are hard to move.
- URN's are **Uniform Resource Name** URI's that include no location information.
- A URN lookup service may be used to determine a URL from a URN.
- URL's are the most frequently used form of URI.

Examples (the third is from Wikipedia):

1. URL: http://www.cmu.edu/service
2. URN: urn:ISBN:0-111-2345-6
3. "you can find urn:ietf:rfc:3187 (URN) over at http://tools.ietf.org/html/rfc3187.html (URL)."

Web Service Composition

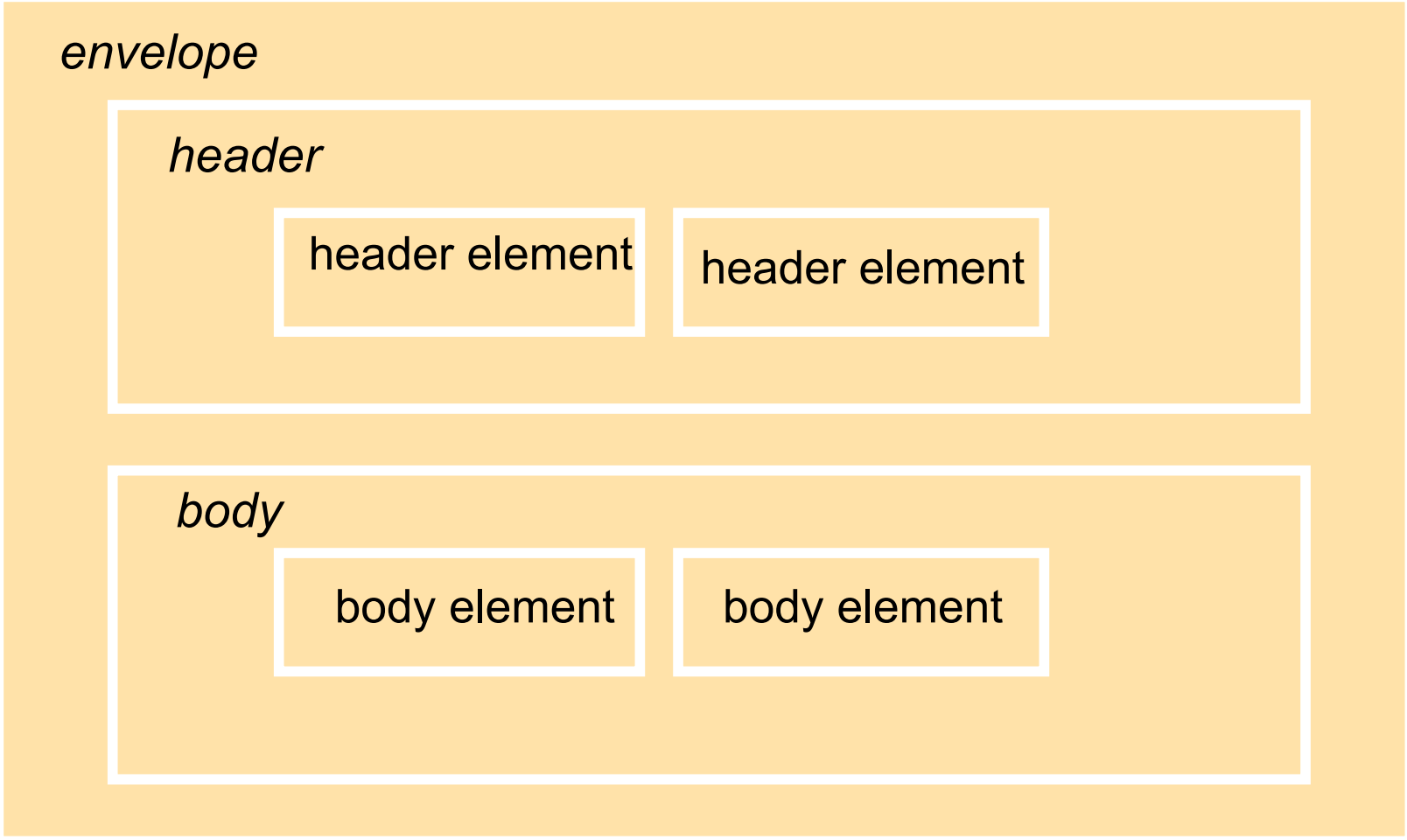


What concerns are not shown?
Transactions, Security (privacy, identification, authentication, authorization), Reliability, Orchestration tooling, Interoperability through Standards, RPC or Messaging, Service Level agreements

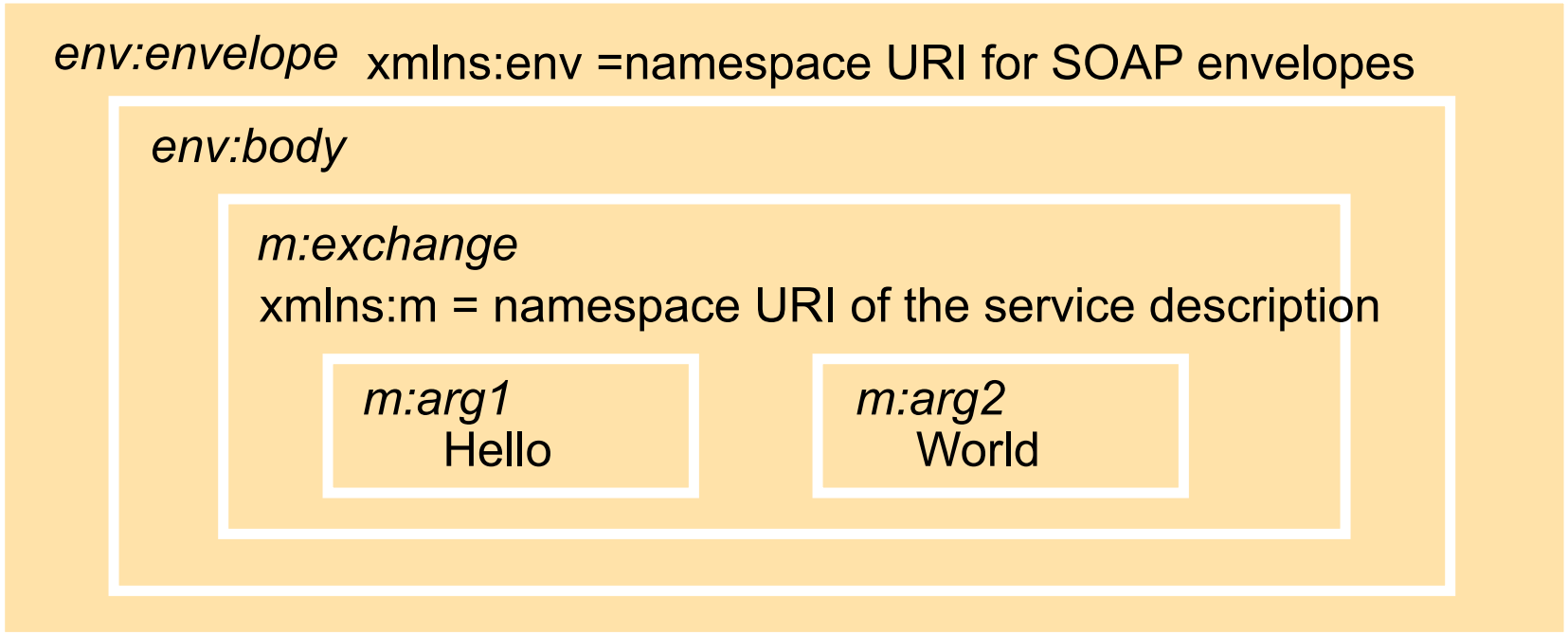
SOAP

- Defines a scheme for using XML to represent the contents of request and reply messages as well as a scheme for the communication of XML documents.
- It is intended that a SOAP message can be passed via intermediaries on the way to the computer that manages the resources to be accessed.
- The intermediaries may process the SOAP to provide security or transaction support as well as other services.
- Typically, the SOAP header is processed by intermediaries and the SOAP body holds the request or reply.

SOAP Envelope

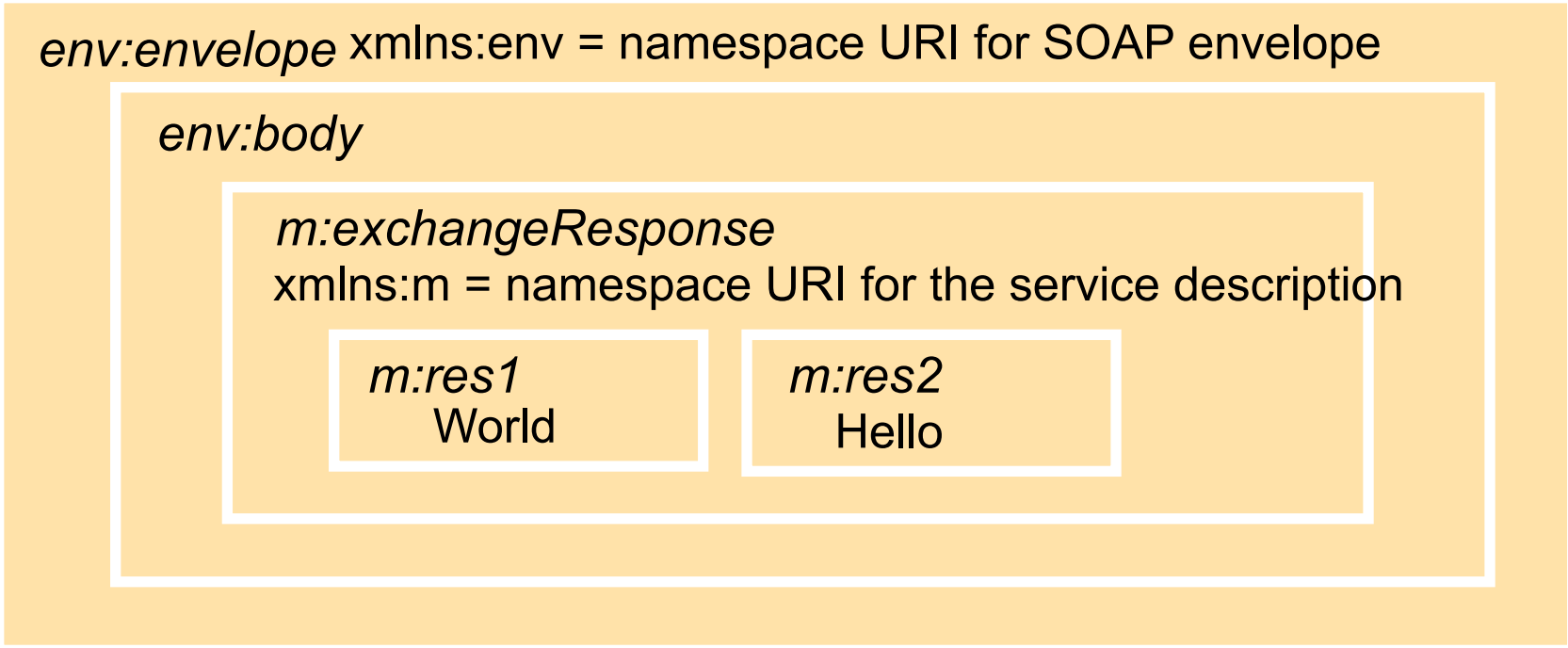


Request Without Headers



Why is envelope in a different namespace than arg1? Consider GreenBayPackers:MikeMcCarthy and CMU:MikeMcCarthy

Corresponding Reply



HTTP POST Example

```
POST /examples/stringer
Host: www.cdk4.net
Content-Type: application/soap+xml
Action: http://www.cdk4.net/examples/stringer#exchange


<env:envelope xmlns:env=namespace URI for SOAP envelope>
<env:header> </env:header>
<env:body> </env:body>
</env:Envelope>
```

endpoint address

action

HTTP header

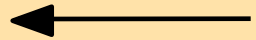
Soap message

A transport protocol is required to send a SOAP document to its destination.

Other transports may be used. WS-Addressing may be used to include destination and source. Thus, different protocols might be used over different parts of the route of a message.

REST Style WS

POST /examples/stringer
Host: www.cdk4.net



Use a URI and an HTTP method to select what needs to be done.

Drop the SOAP and use name value pairs for the request.
Use XML or JSON for the response.

Appropriate for ad-hoc mashups. Some believe that REST is not appropriate for enterprise computing.

WS-Addressing

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
            xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <S:Header>
      <wsa:MessageID>
          uuid:6B29FC40-CA47-1067-B31D-00DD010662DA
      </wsa:MessageID>
      <wsa:ReplyTo>
          <wsa:Address>http://business456.example/client1</wsa:Address>
       </wsa:ReplyTo>
       <wsa:To>http://fabrikam123.example/Purchasing</wsa:To>
       <wsa:Action>http://fabrikam123.example/SubmitPO</wsa:Action>
    </S:Header>
  <S:Body>
    ...
  </S:Body>
</S:Envelope>
```

Address information included within the document rather than only being specified by the transport. What does this buy us?

Distributed Objects?

At first glance, the interaction between client and server seems like RMI. We will look at RMI soon.

But, RMI permits the creation of **remote objects**. These may then be accessed via remote references.

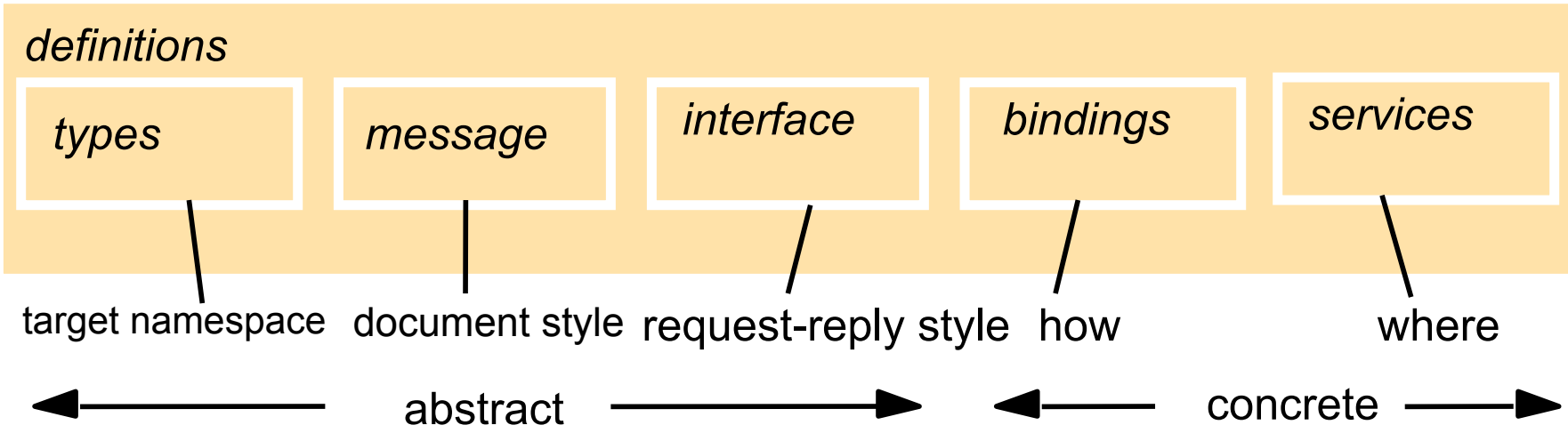
Web services may create and use objects but never return a remote reference to a remote object. **A web service is a single object that offers a set of procedures**.

Web services are simpler than distributed objects. Simple is good.

Service Descriptions

- The primary means of describing a web service is by using WSDL (the Web Services Description Language)
- XML Schema may be used to describe documents.
- WSDL makes use of XML Schema to describe an exchange of messages.
- A Service Description (WSDL document) is an IDL (interface definition language) plus it contains information on how and where the service may be accessed.
- It contains an abstract part and a concrete part. The abstract part is most like a traditional interface. The concrete part tells us how and where to access the service.

The Main Elements in a WSDL Description



A binding is a choice of protocols.
A service holds an endpoint address.
Client or server side code may be generated automatically from the WSDL.
A WSDL document may be accessed directly or indirectly through a registry like UDDI (Universal Directory and Discovery Service).

WSDL MEPS

Name	*Messages sent by*		
	Client	*Server*	*Fault message*
In-Out	*Request*	*Reply*	may replace *Reply*
In-Only	*Request*		no fault message
Robust In-Only	*Request*		may be sent
Out-In	*Reply*	*Request*	may replace *Reply*
Out-Only		*Request*	no fault message
Robust Out-Only		*Request*	may send fault

Which of these does a fire alarm service provide?

XSDL and WSDL

- XSDL (The XML Schema Definition Language) allows us to describe the structure of an XML message
- WSDL allows us to describe message exchanges

Notes from article by Aaron Skonnard. See the schedule for the URL of this article

WSDL

- A message exchange is called an *operation.*
- Related operations are grouped into *interfaces.*
- A *binding* specifies concrete details about what goes on the wire.

WSDL

- Describes the contract between applications.
- Can be automatically generated from a collection of Java or C# classes.
- Can be read by utilities that generate client side proxy code or server side skeletons.
- See wsimport (JDK 6.0) or wsdl.exe on the Microsoft side. In Netbeans just drag and drop a web reference.

WSDL Structure

<definition>
    <!– abstract definitions →
    <types>
    <messages>
    <portType>
    <!– concrete definitions →
    <binding>
    <service>
</definition>

WSDL Structure

<definition>
    <!– Terms found in application code →
    <types>
    <messages>
    <portType>
    <!– Handled by XML infrastructure →
    <binding>
    <service>
</definition>

WSDL Structure

<definition>

<types>

- a container for XSDL Type definitions
- element names may be defined here as well

WSDL Structure

<definition>

<types>

For example, in Google's WSDL, GoogleSearchResult is defined as a complex type with many elements.

WSDL Structure

<definition>
  <types>
  <message>

- May have more than one part (think parameters)
- Define the input or output of an operation
- RPC style messages associate a name with a type (defined above)
- Document style messages associate a name with an XML element

</definition>

WSDL Structure

<definition>

<types>

<message> Two examples:

- In Google's WSDL, a doGoogleSearch message is defined with many parts of basic xsd types.
- In Google's WSDL, a doGoogleSearchResponse message is defined as of type GoogleSearchResult

</definition>

WSDL Structure

<definition>
<types>
<messages>
<portType>

- The definition of an interface or group of operations
- The term “portType” will be replaced with the term “interface” in WSDL 1.2
- Each operation has a name and normally specifies both input and output messages

</definition>

WSDL Structure

<definition>
<types>
<messages>
<portType>

- For example, in Google's WSDL, GoogleSearchPort contains three operations.
- The operation doGoogleSearch has an input message (doGoogleSearch) and an output message (doGoogleSearchResponse.)

</definition>

WSDL Structure

<definition>
  <types> <messages> <portType>
<binding>

- Each binding has a unique name that is associated with a particular interface.
- The protocol used is specified.
- Details found here specify how the data will look on the wire.

</definition>

WSDL Structure

<definition>
<types> <messages> <portType>
<binding>

- For example, in Google's WSDL, the binding name GoogleSearchBinding is introduced and is associated with the interface GoogleSearchPort.
- Each operation within that interface is described as soap operations.

</definition>

WSDL Structure

<definition>
<types><messages><portType>
<binding>
<service>

- Defines a collection of ports (endpoints) that exposes a particular binding
- An address is associated with a binding

</definition>

WSDL Structure

<definition>
<types><messages><portType><binding>
<service>

For example, in Google's WSDL, the service name GoogleSearchService is introduced. The interface GoogleSearchPort is associated with the binding GoogleSearchBinding.

The service element holds the address of the service.

</definition>

Writing A Google Client

(1) Get the WSDL from http://www.google.com/apis/

(2) If using .NET run wsdl.exe on GoogleSearch.wsdl.

(3) If using Java and Axis run wsdl2java.bat on GoogleSearch.wsdl.

(4) wsdl2java.bat holds the line

```
java org.apache.axis.wsdl.WSDL2Java  %1
```

The WSDL2Java class is in axis.jar

A Google Client in Java

```
// Running a simple Google RPC client for spell checking

import GoogleSearch.*;                    // wsdl2java generated package

public class MyGoogleClient{

    private static String endpointAddress = "http://api.google.com/search/beta2";

   public static void main(String[] args) throws Exception {

        if(args.length != 1) {
              System.out.println("Usage1: java MyGoogleClient wordToSpellCheck");
              System.out.println(
              "Usage2: java MyGoogleClient \"a phrase to spell check\"");
              System.exit(0);
        }
```

```
        System.out.println("Contacting Google Web Service at  " + endpointAddress);
        System.out.println("Checking on spelling of  '" + args[0]+"'");

       GoogleSearchServiceLocator loc = new GoogleSearchServiceLocator();

       GoogleSearchPort gp = loc.getGoogleSearchPort();

        String answer = gp.doSpellingSuggestion(
                                "n6lHU/FQFHIHzpbzRTPFvrUP4Cw+/k+N",
                                args[0]);

       if(answer == null) System.out.println("Google likes the spelling of  '" + args[0]+"' );
       else System.out.println("Google suggests the spelling '" + answer +"' );
    }
}
```

GoogleSpring2005\java>java MyGoogleClient "Cornegi Melon Universeti"
Contacting Google Web Service at http://api.google.com/search/beta2
Checking on spelling of 'Cornegi Melon Universeti‘

Google suggests the spelling 'Carnegie Mellon University'

A Google Client in C#

```
// run a client against Google's web service
using System;

namespace ConsoleApp
{
    class GoogleClient
    {
        public static void Main(string[] args) {

                    try {
                            GoogleSearchService s =
                             new GoogleSearchService();
```

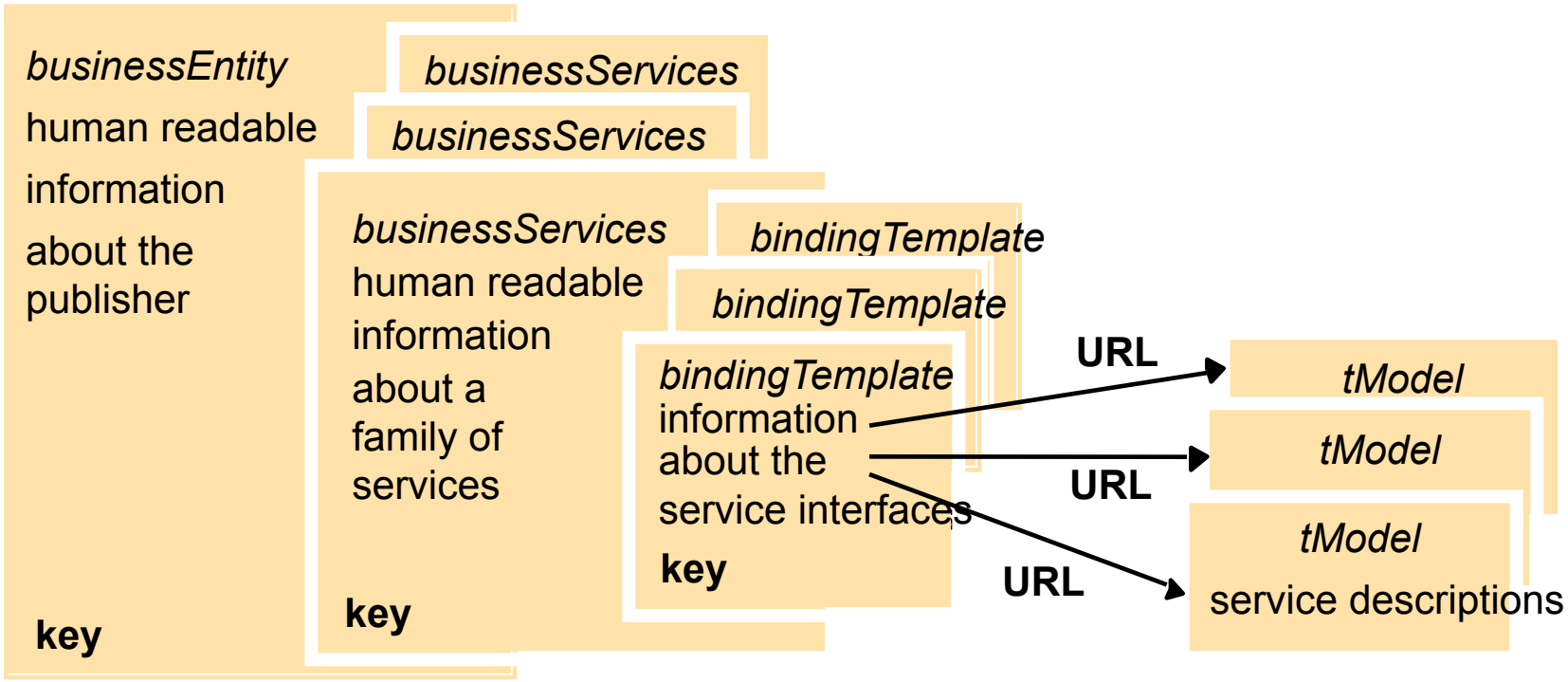
```
            Console.WriteLine("Enter word to spell check");
            String word = Console.ReadLine();
               String answer = s.doSpellingSuggestion(
                              "n6lHU/FQFHIHzpbzRTPFvrUP4Cw+/k+N", word);
               Console.WriteLine("Google returned " + answer);
            }
            catch(Exception ) {Console.WriteLine("Google threw an exception");}
         }
   }
}
```

UDDI

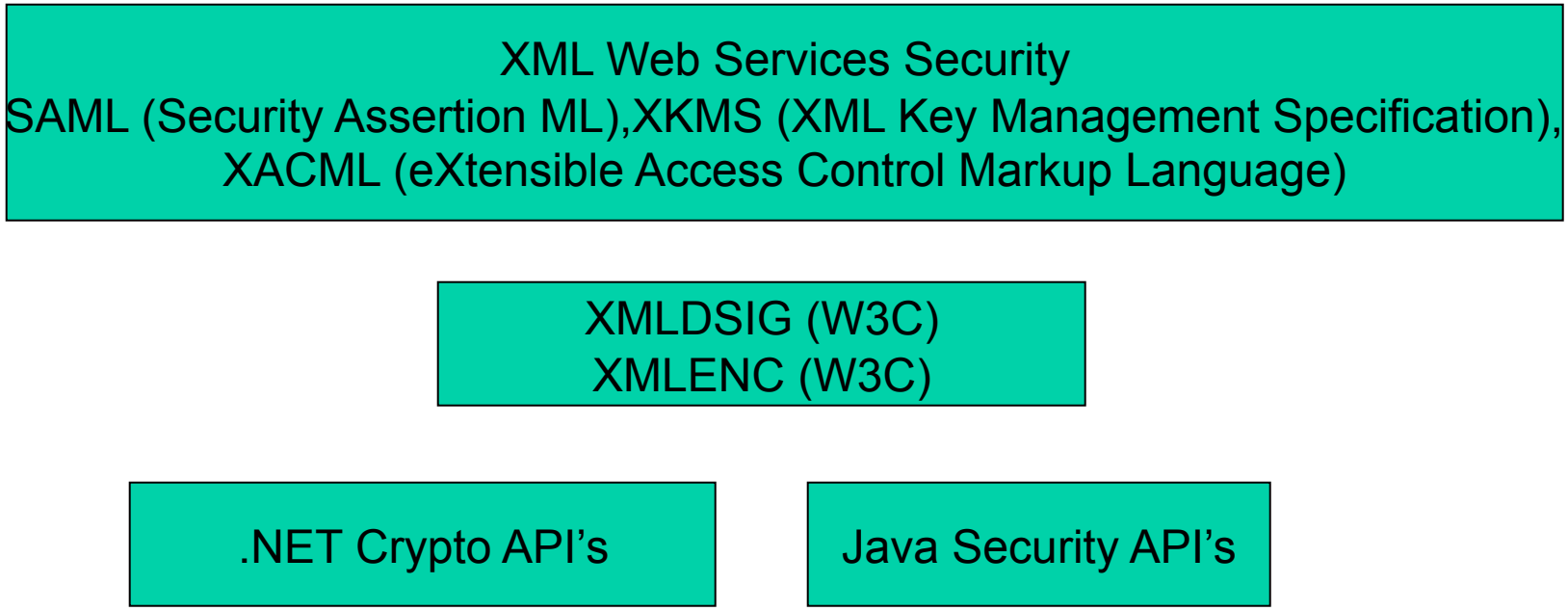
- An acronym for Universal Directory and Discovery Services.
- A directory service for use with web services.
- One way to obtain service descriptions.
- May be used within organizations to perform lookups for WSDL documents.
- Supports white pages (lookup by name) and yellow pages (lookup by attribute)
- Provides a publish/subscribe interface.
- Uses replication among many servers for scalability.
- JAXR (The Java API for XML Registries) may be used to interact with UDDI.

UDDI has not been as widely supported as was originally hoped.

UDDI Data Structures



Web Services Security Stack (Quick Overview)



We will dive deeper when we cover cryptography.

Travel Agent Scenario

1. The client asks the travel agent service for information about a set of services; for example, flights, car hire and hotel bookings.
2. The travel agent service collects prices and availability information and sends it to the client, which chooses one of the following on behalf of the user:
   (a) refine the query, possibly involving more providers to get more information, then repeat step 2;
   (b) make reservations;
   (c) quit.
3. The client requests a reservation and the travel agent service checks availability.
4. Either all are available;
   or for services that are not available;
   either alternatives are offered to the client who goes back to step 3;
   or the client goes back to step 1.
5. Take deposit.
6. Give the client a reservation number as a confirmation.
7. During the period until the final payment, the client may modify or cancel reservations

The Business Process Execution Language (BPEL) is used to write such scenarios in XML.

Many BPEL designer tools are based on BPMN. BPEL XML is the standard result.