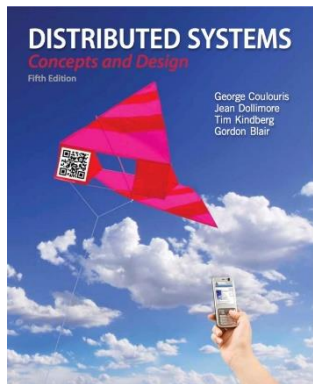




# Web Services



Most concepts are  
drawn from Chapter 9

Slides are revised from  
Dr. Michael McCarthy at CMU

**Dr. Minxian Xu**

**Associate Professor**

**Research Center for Cloud Computing**

**Shenzhen Institute of Advanced Technology, CAS**

**<http://www.minxianxu.info/dcp>**

天网疏难漏，世网密莫通。我心久不动，一脱二网中  
高竹漱清泉，长松迎清风。此时逢此景，正与此心同。

——（宋）邵雍

# Review

---

- Q1: Name and explain three transparencies that should be addressed by distributed file systems.

# Review

---

## **Access transparency**

- Client programs don't know if the file is local or remote

## **Location transparency**

- Client programs don't know where the file is stored
- Files can be relocated without changing their pathname

## **Mobility transparency**

- Neither client programs nor system administration tables in client nodes need to be changed when files are moved

## **Performance transparency**

- Maintain acceptable performance while the load on the service varies within a specified range

## **Scaling transparency**

- Service can be expanded without loss of performance

# Review

---

- Q2: What are the advantages and disadvantages of using absolute names as a naming strategy?

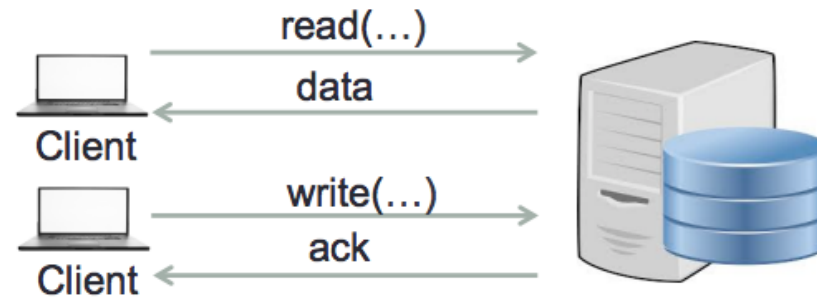
# Review

---

- **Absolute name**
- provides a complete address to a file including both the server and path names: <machine name: path name>
- **Advantages**
- Trivial to find a file once the name is given
- No additional state must be kept since each name is self contained (No global state)
- Greater scalability
- Easy to add and delete new names
- **Disadvantages**
- No location transparency
- File is location dependent and cannot be moved
- Less resilient to failure

# Review

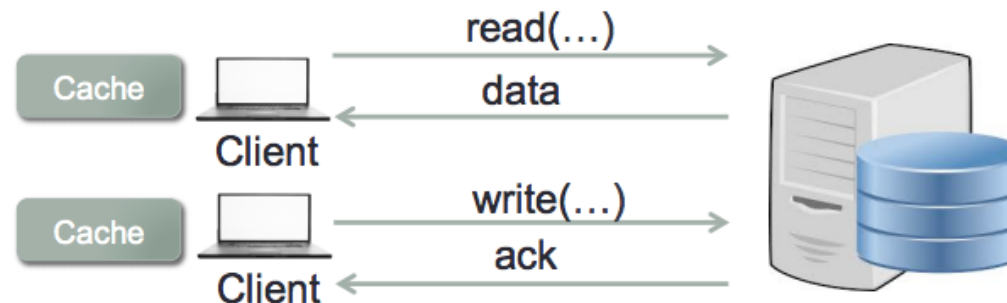
- Q3: What are the advantages and disadvantages of the following simple distributed file system?
- RPC to access file system calls
  - No client/local caching



- **Advantages**
  - Consistent view of file system
- **Disadvantages**
  - Performance (network access, server becomes potential bottleneck)

# Review

- Q4: What are the advantages and disadvantages of the following simple distributed file system?
- Cache files on clients and perform local file system operations



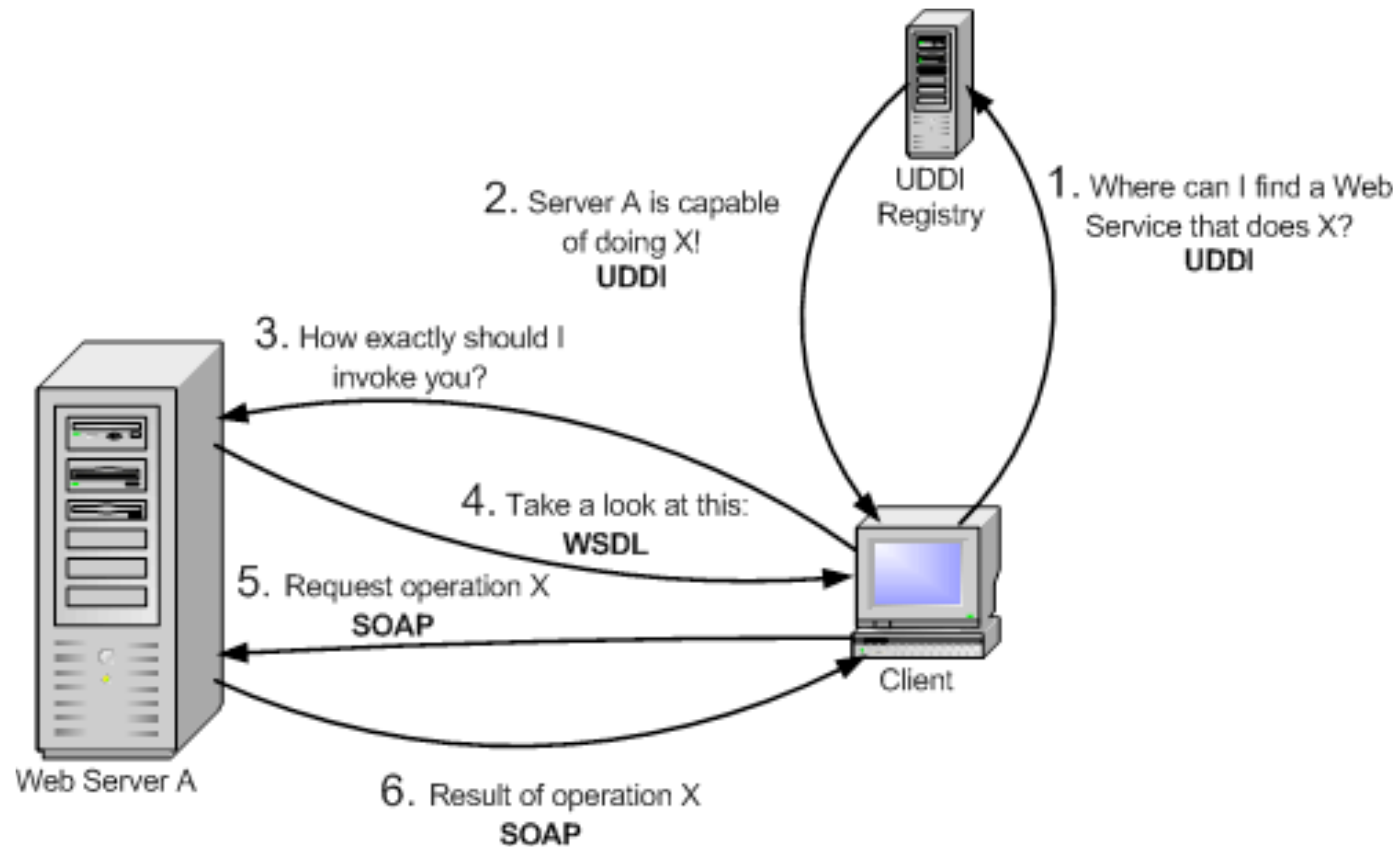
- **Advantages**

- Local operations = better performance

- **Disadvantages**

- What happens when the client fails?
- Where should the client store the cached files?
- Difficult to keep local copy consistent with remote copy

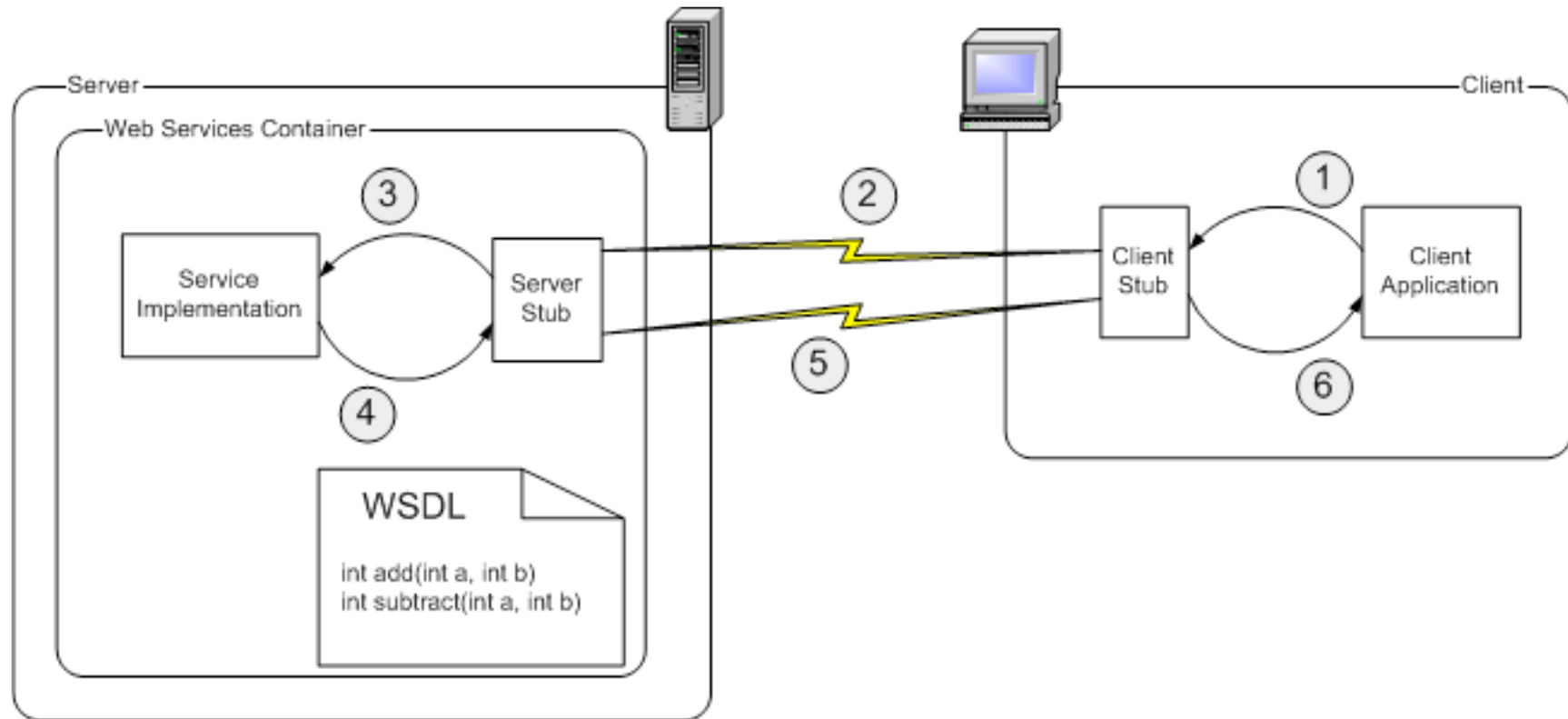
# What is a web service?



From Globus.org (Grid Computing)



# With Subs



From Globus.org (Grid Computing)

# Some Important Standards

## Web Services and Grid Computing

SOAP (W3C), WSDL (W3C), UDDI (OASIS), WS Interop (WS-I), Grid (GGF)

**SQL/XML**  
(ANSI & ISO)

**XML Transformations (W3C)**  
XPath, XSL, XSLT, XQuery

**XML APIs**  
DOM (W3C), SAX

**XML Vocabularies (OASIS, etc)**

**Basic XML Constructs (W3C)**

Canonical XML, XML Fragments, XInclude, XLink, XPointer, XPath

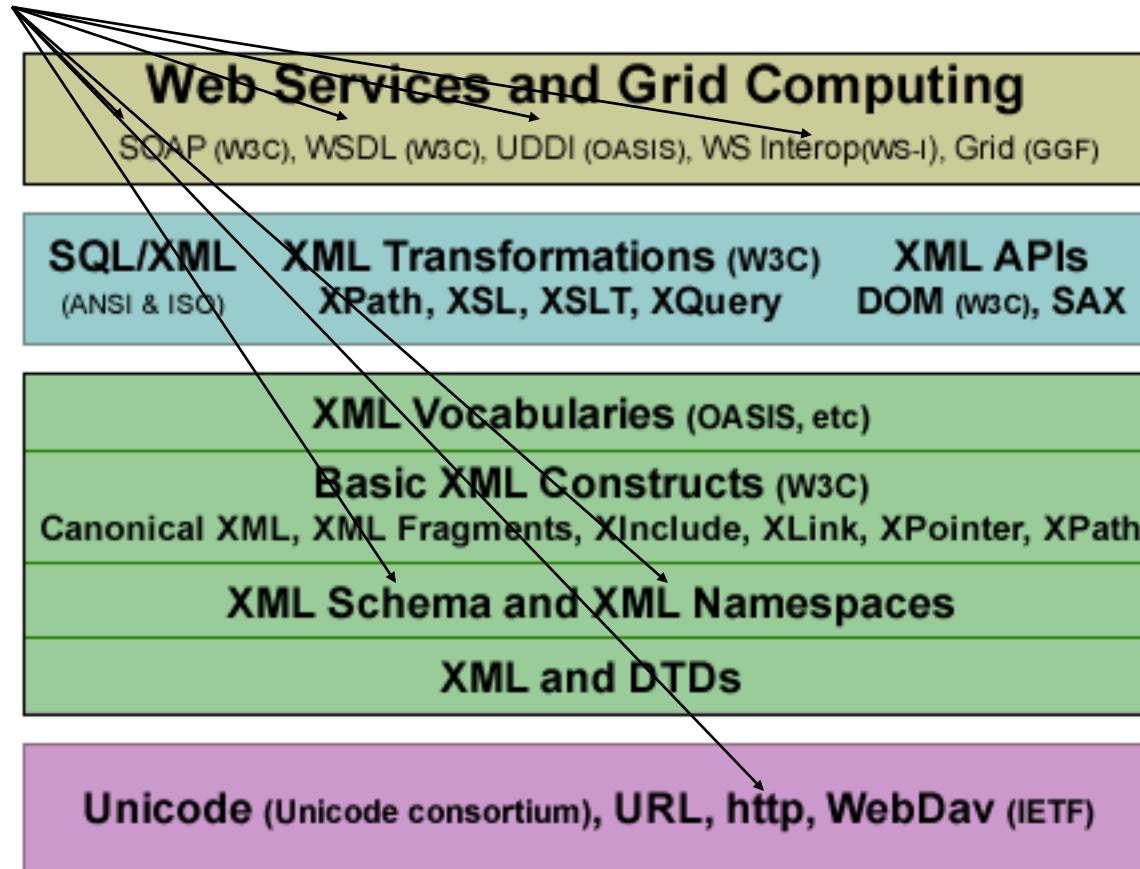
**XML Schema and XML Namespaces**

**XML and DTDs**

**Unicode (Unicode consortium), URL, http, WebDav (IETF)**

# Some Important Standards

Very important  
with respect to  
XML web services.

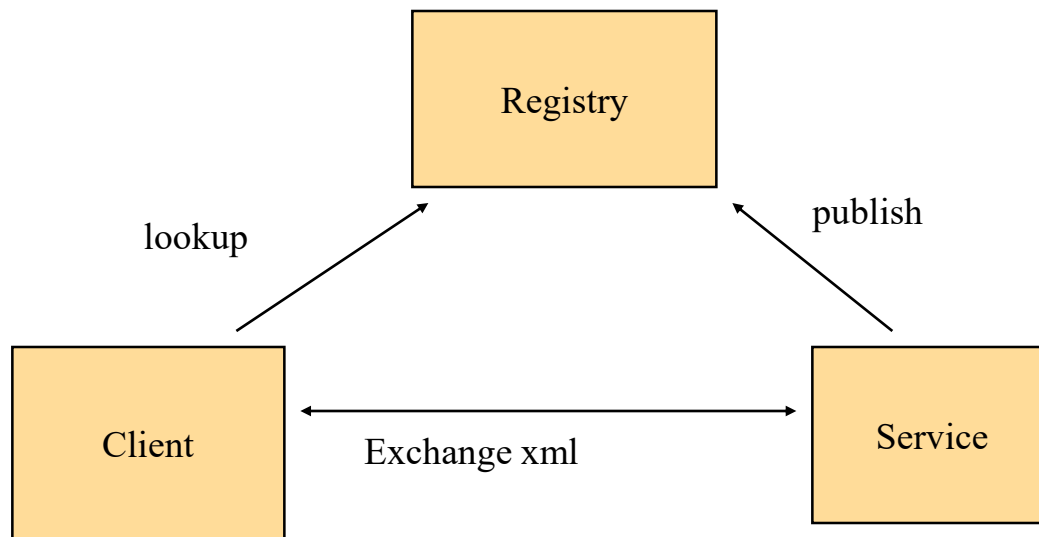


# Web Services

---

- 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



Applications

Directory service   Security   Orchestration

Web Services

Service descriptions (in WSDL)

SOAP

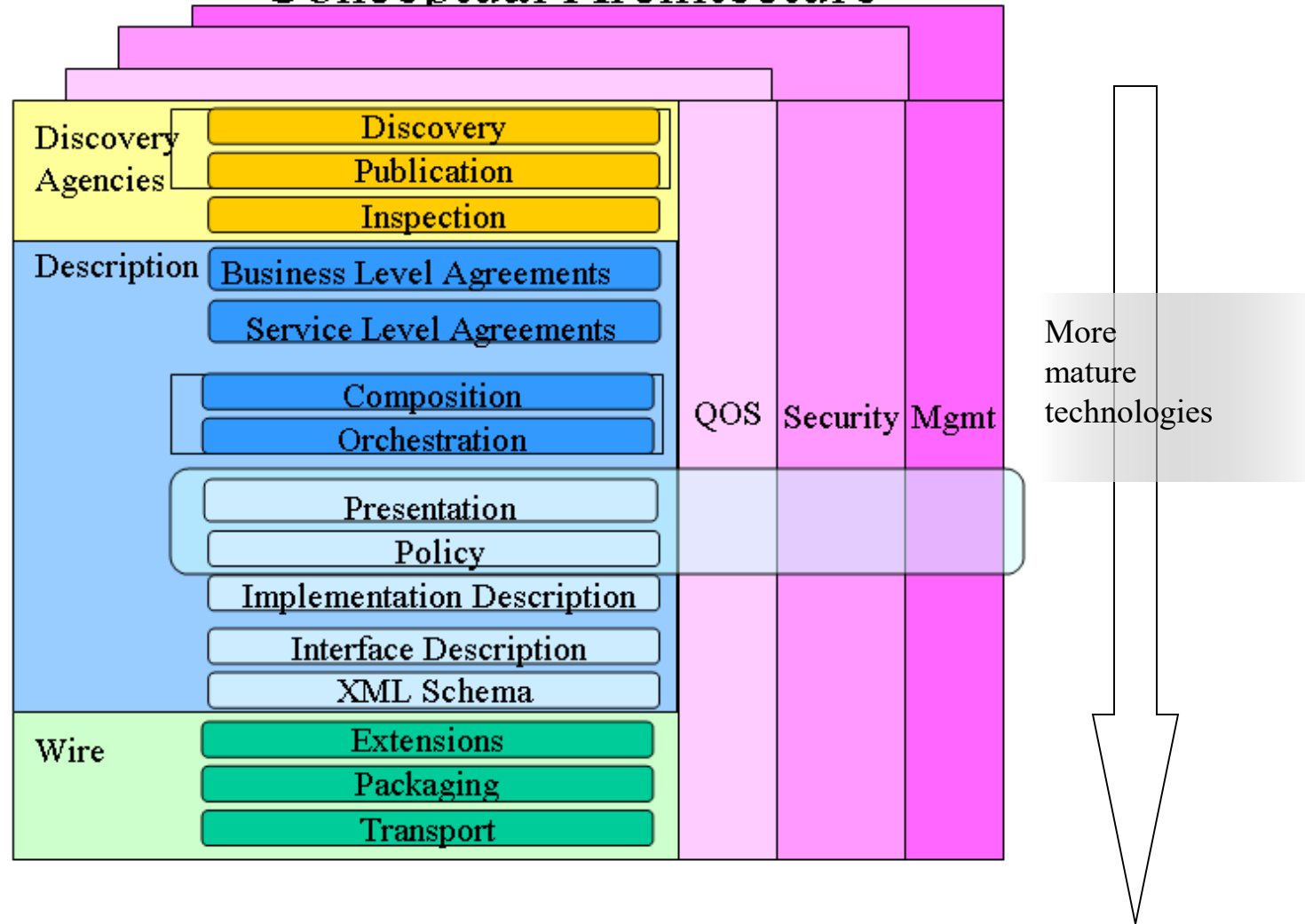
URIs (URLs or URNs)

XML

HTTP, SMTP or other transport

# The Complete Web Services “Stack”

## Conceptual Architecture



# 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

---

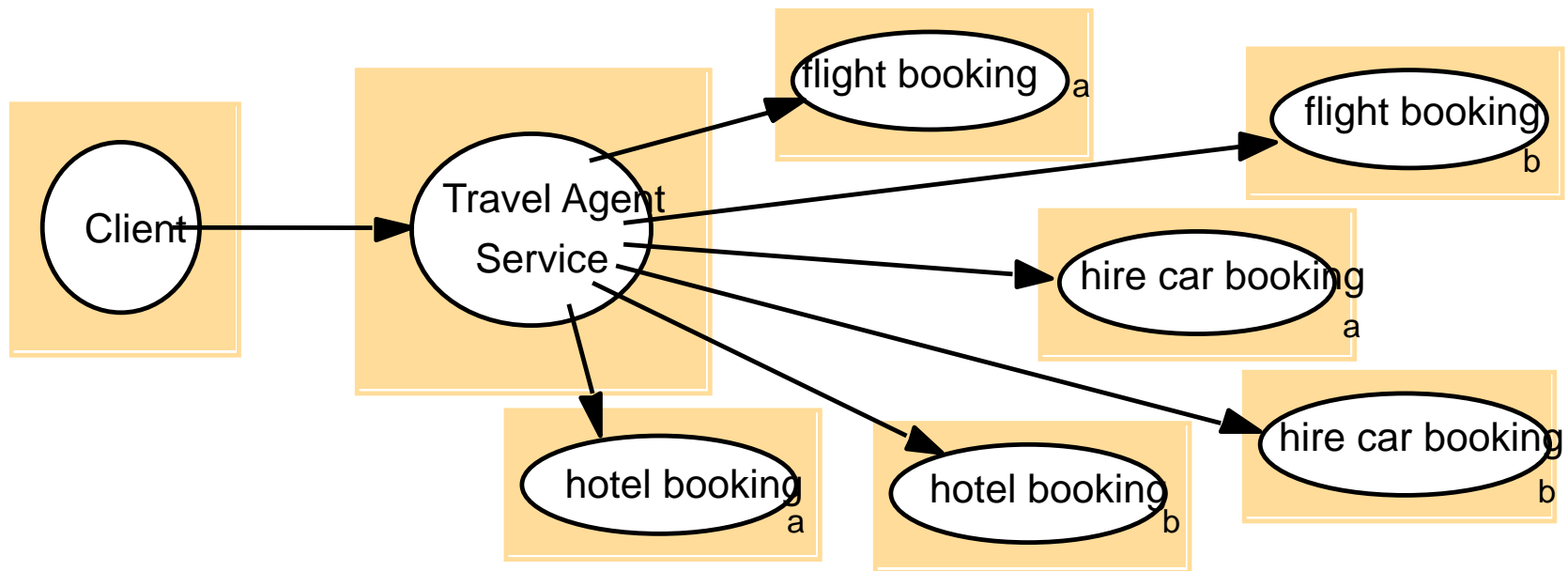
- URI's are Uniform Resource Identifiers.
- URL's are Uniform Resource Locator URI's that include location information. Thus, resources pointed to by URL's are hard to move.
- URN's are Uniform Resource Name URI's that include no location information.
- A URN lookup service can be employed to determine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

Examples:

URL: <http://www.siat.ac.cn>

URN: <urn:ISBN:0-111-2345-6>

# Web Service Composition



- 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.

*envelope*

*header*

header element

header element

*body*

body element

body element

# Request Without Headers

*env:envelope* xmlns:env = namespace URI for SOAP envelopes

*env:body*

*m:exchange*

xmlns:m = namespace URI of the service description

*m:arg1*  
Hello

*m:arg2*  
World

In this figure and the next, each XML element is represented by a shaded box with its name in italic followed by any attributes and its content

# Corresponding Reply

*env:envelope* xmlns:env = namespace URI for SOAP envelope

*env:body*

*m:exchangeResponse*

xmlns:m = namespace URI for the service description

*m:res1*  
World

*m:res2*  
Hello

# HTTP POST Example

<i>POST /examples/stringer</i>	← endpoint address	HTTP header
<i>Host: www.cdk4.net</i>		
<i>Content-Type: application/soap+xml</i>		
<i>Action: http://www.cdk4.net/examples/stringer#exchange</i>	← action	
 <i>&lt;env:envelope xmlns:env=namespace URI for SOAP envelope&gt;</i> <i>&lt;env:header&gt; &lt;/env:header&gt;</i> <i>&lt;env:body&gt; &lt;/env:body&gt;</i> <i>&lt;/env:Envelope&gt;</i>		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.**

# 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.



# Distributed Objects?

---

At first glance, the interaction between client and server seems like RMI.

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

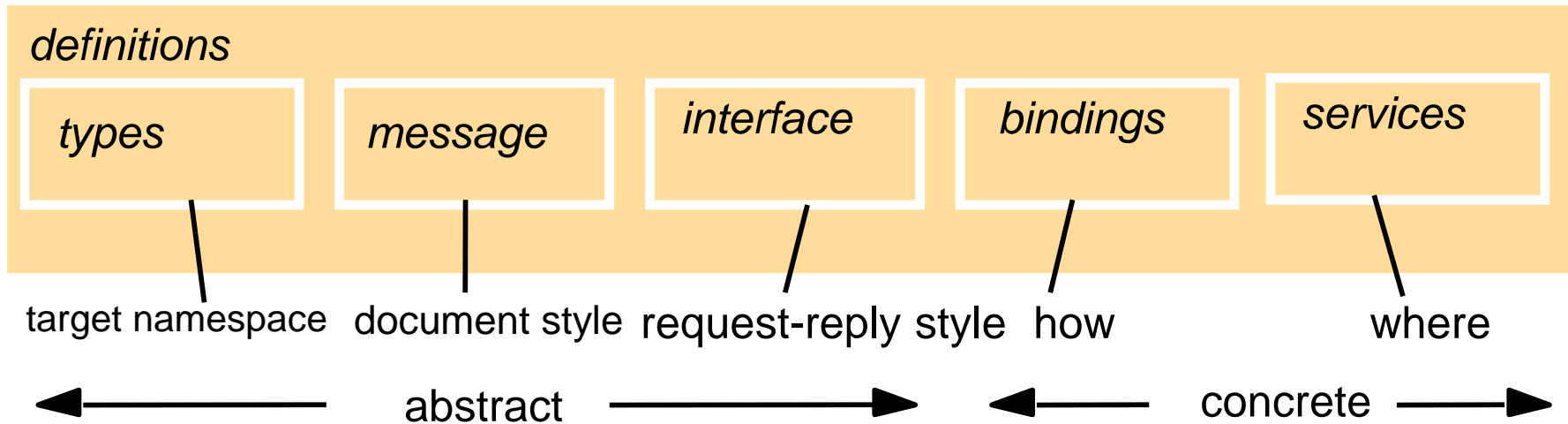
Web services may create and use objects but never return a remote reference to a remote object. **A web service is a single object that offers a set of procedures.**

# 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 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).

<i>Name</i>	<i>Messages sent by</i>			
	<i>Client</i>	<i>Server</i>	<i>Delivery</i>	<i>Fault message</i>
In-Out	<i>Request</i>	<i>Reply</i>		may replace <i>Reply</i>
In-Only	<i>Request</i>			no fault message
Robust In-Only	<i>Request</i>		guaranteed	may be sent
Out-In	<i>Reply</i>	<i>Request</i>		may replace <i>Reply</i>
Out-Only		<i>Request</i>		no fault message
Robust Out-Only		<i>Request</i>	guaranteed	may send fault

# 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

# 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

# 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>`

<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>

<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>`



`<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>`

`<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();
```



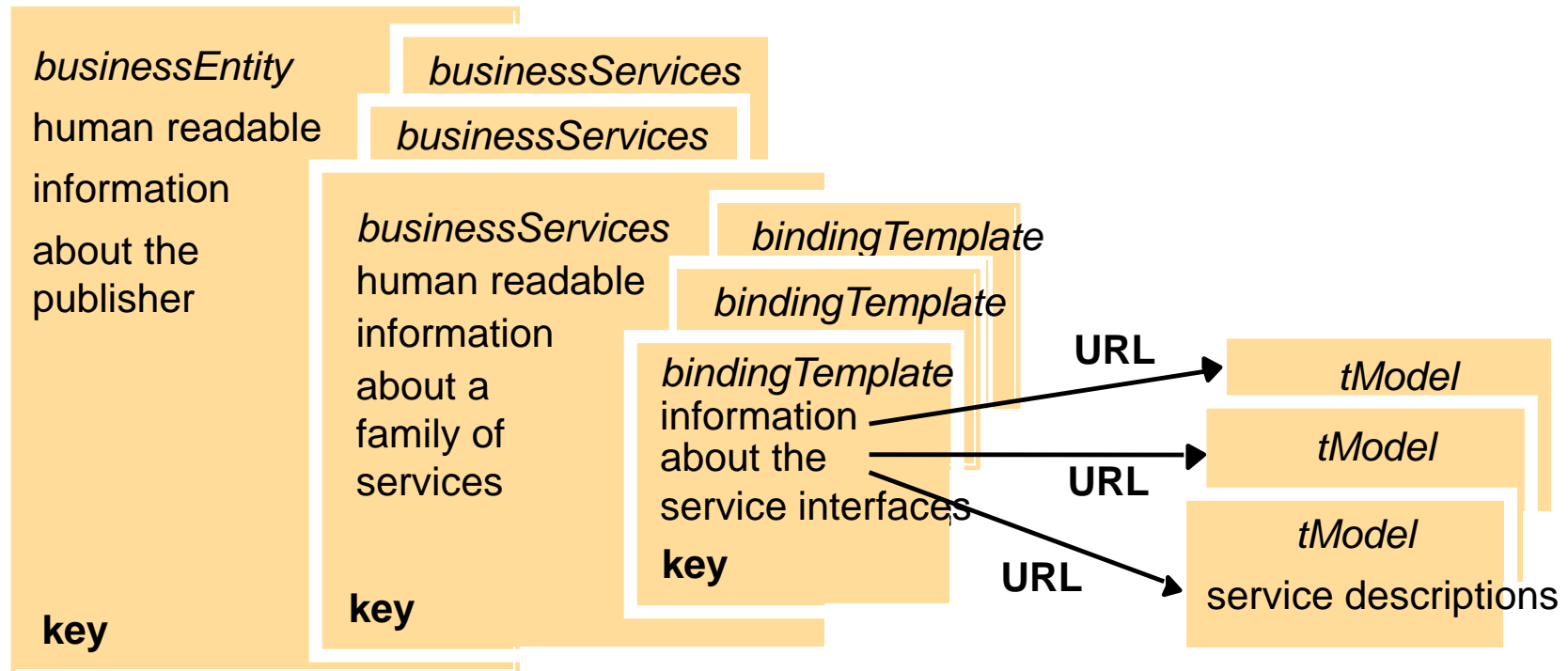
```
Console.WriteLine("Enter word to spell check");
String word = Console.ReadLine();
    String answer = s.doSpellingSuggestion(
        "n6lHU/FQFHIHzpbzRTPFvrUP4Cw+/k+N", word);
    Console.WriteLine("Google returned " + answer);
}
catch(Exception ) {Console.WriteLine("Google threw an exception");}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

# 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 Data Structures



# Web Services Security Stack

---

XML Web Services Security  
SAML (Security Assertion ML), XKMS (XML Key Management Specification),  
XACML (eXtensible Access Control Markup Language)

XMLDSIG (W3C)  
XMLENC (W3C)

.NET Crypto API's

Java Security API's

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.**

# Case Study: The Grid

- Grid refers to middleware that is designed to allow for sharing of resources such as data and CPU cycles on a very large scale.
- Provides for heterogeneity, management, and security.
- Latest version runs over web services.
- The open source Globus Toolkit implements the grid architecture.
- The immense quantity of data in archives makes ftp or web access infeasible.





American Astronomical Society

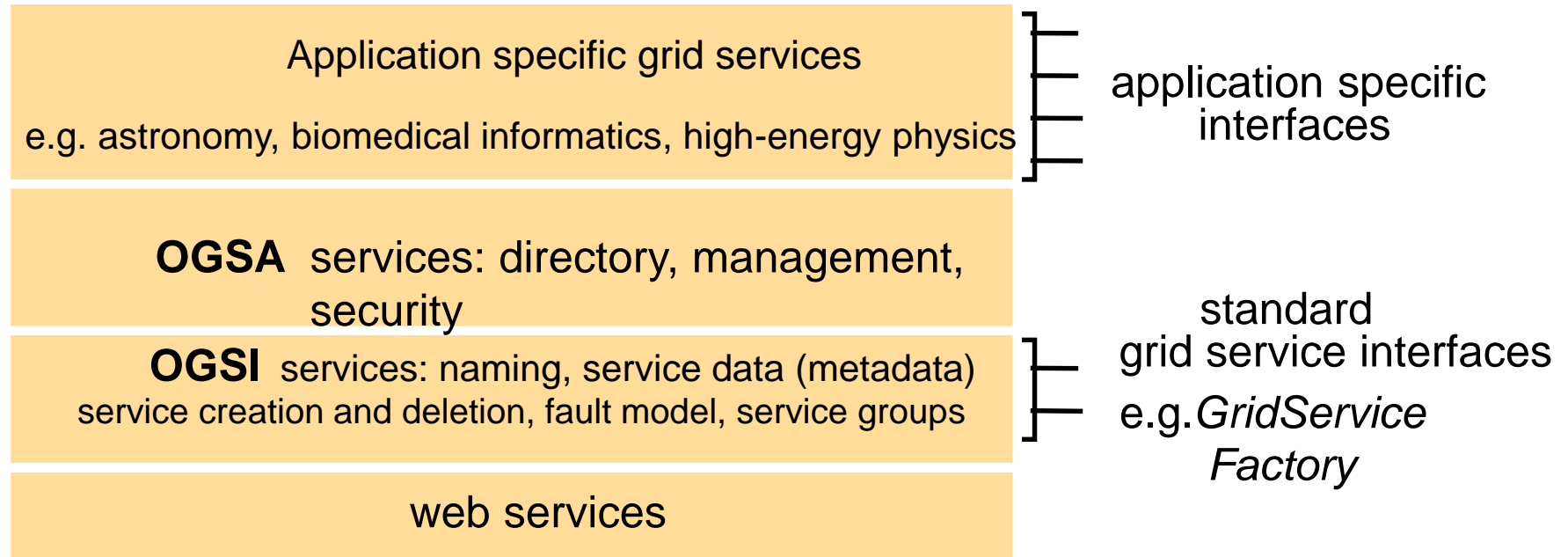
**WorldWide Telescope**



Welcome to AAS WorldWide Telescope!

- Move around the sky by clicking and dragging.
- Zoom in/out by scrolling the mouse wheel or pressing +/-.
- Right-click in the view to display the Finder Scope for more information.
- Menu tabs ("Explore", "Guided Tours", etc.) have two parts. Click the tab's top to open a pane; click the tab's bottom to open a submenu.

# Open Grid Services Architecture



# Some Grid Projects



---

| <i>Description of the project</i>   | <i>Reference</i>   |
|---|--|
| 1. <i>Aircraft engine maintenance using fault histories and sensors for predictive diagnostics</i>  | <a href="http://www.cs.york.ac.uk/dame">www.cs.york.ac.uk/dame</a> |
| 2. <i>Telepresence for predicting the effects of earthquakes on buildings, using simulations and test sites</i>   | <a href="http://www.neesgrid.org">www.neesgrid.org</a>             |
| 3. <i>Bio-medical informatics network providing researchers with access to experiments and visualizations of results</i>  | <a href="http://nbc.sdsc.edu">nbc.sdsc.edu</a>                     |
| 4. <i>Analysis of data from the CMS high energy particle detector at CERN by physicists world-wide over 15 years</i>  | <a href="http://www.uscms.org">www.uscms.org</a>                   |
| 5. <i>Testing the effects of candidate drug molecules for their effect on the activity of a protein, by performing parallel computations using idle desktop computers</i> | [Taufers et al. 2003]<br>[Chien 2004]                              |
| 6. <i>Use of the Sun Grid Engine to enhance aerial photographs by using spare capacity on a cluster of web servers</i>  | <a href="http://www.globexplorer.com">www.globexplorer.com</a>     |
| 7. <i>The butterfly Grid supports multiplayer games for very large numbers of players on the internet over the Globus toolkit</i>   | <a href="http://www.butterfly.net">www.butterfly.net</a>           |
| 8. <i>The Access Grid supports the needs of small group collaboration, for example by providing shared workspaces</i>   | <a href="http://www.accessgrid.org">www.accessgrid.org</a>         |

---