Web Services and gSOAP

Computer Architecture Area (ARCOS)

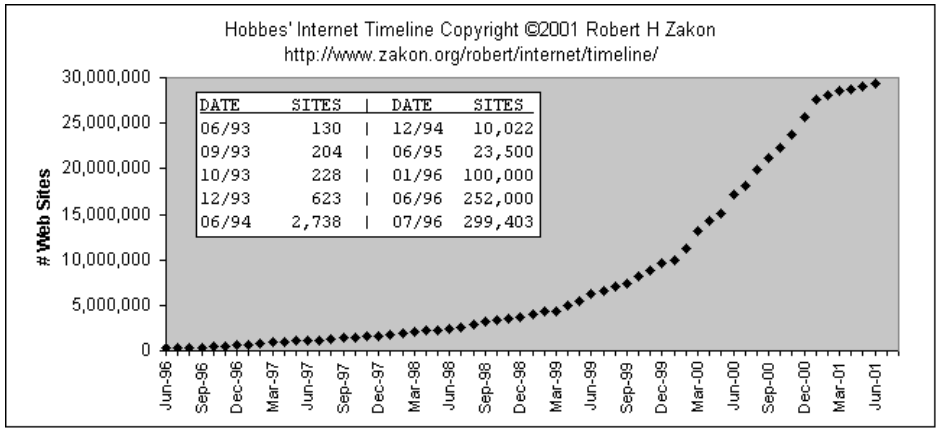
Distributed Systems

Bachelor in Informatics Engineering

Universidad Carlos III de Madrid

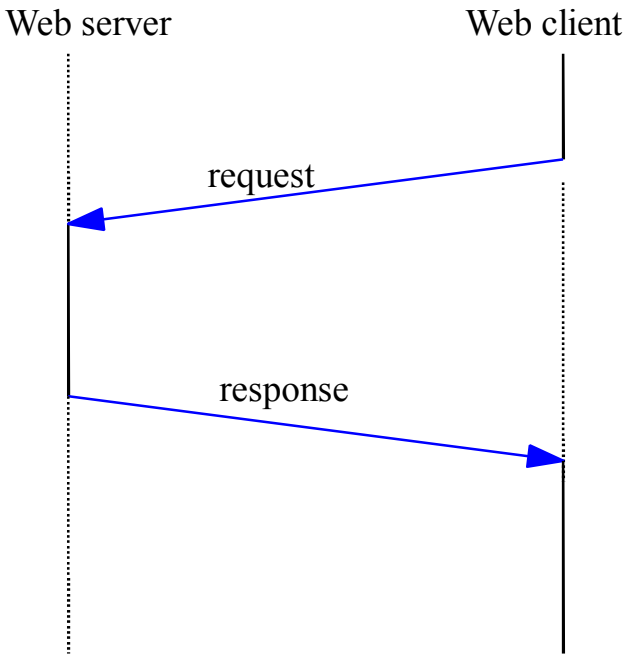
Introduction

- Elements that form the *World Wide Web:*
  - Hipertext documents
  - HTTP protocol
  - HTML language
  - Web server
  - Web browsers



- Web server and client.

HTTP Protocol



Request elements:
- <command> <document address> <HTTP version>
- optional header
- optional data

Response elements:
- state line with format <protocol><state code><description>
- header information
- document

HTTP Protocol: client

- Connection establishment.
  - Example: telnet www.uc3m.es 80

- Request line:

```
<HTTP method><espace><requested URI><espace><protocol>\r\n
```

  - HTTP methods:

```
GET:  requests a web page
HEAD: requests a web page header
POST: sends data to a web page
PUT:  sends a web page to be stored
```

  - URI: *uniform resource identifier*.

  - Protocol especification
    HTTP/1.0
    HTTP/1.1

  - Example: GET /dir1/index.html HTTP/1.0

HTTP Protocol : server

- Response contents:
  - Diagnosis:
    - `<protocol> <code> <description>`
      - Protocol understood by the server
      - Codes: 200 OK, 400 client error, 500 server error…
  - Metadata:
    - `<id>: <value>`
  - Requested URI content
    - `<data>`

HTTP Protocol : server

- Response contents:

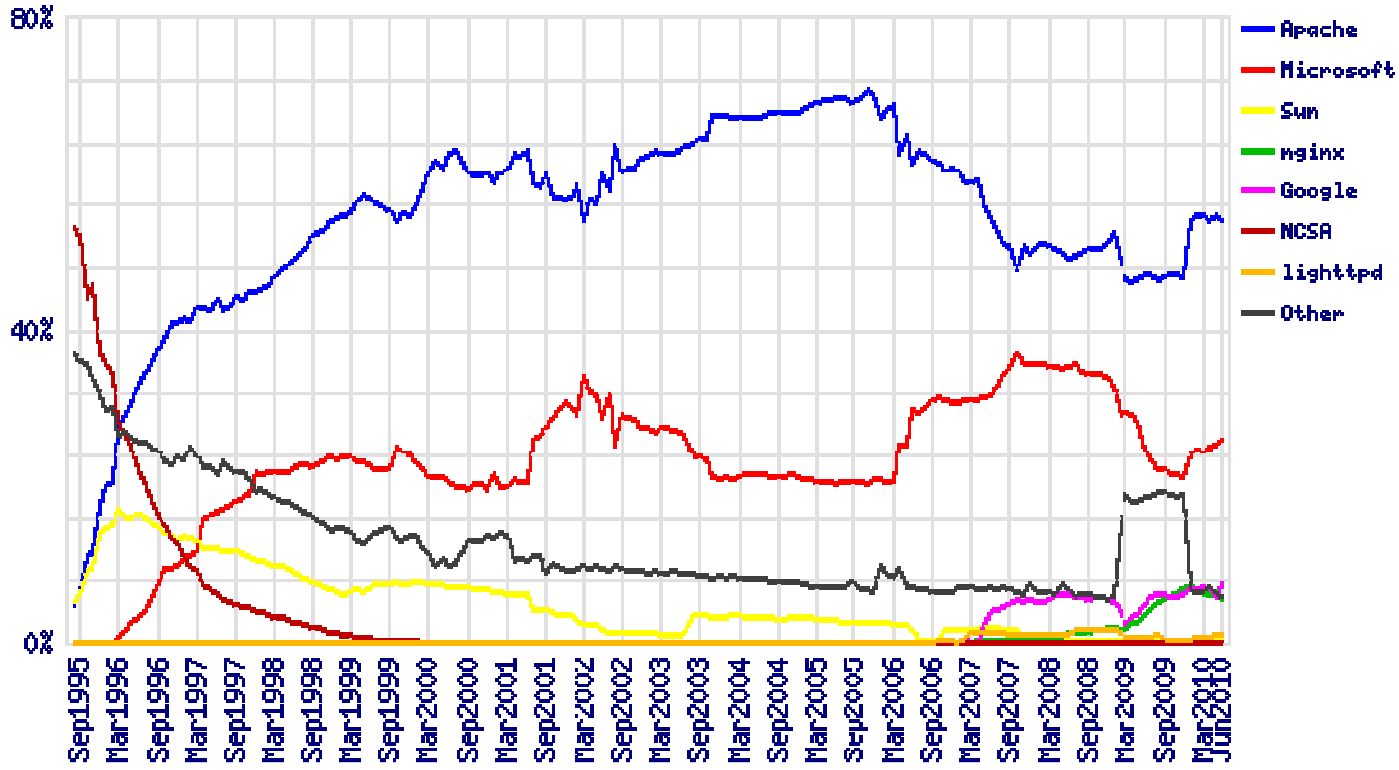
```
HTTP/1.1 200 OK

Date: Sat, 15 Sep 2001 06:55:30 GMT
Server: Apache/1.3.9 (Unix) ApacheJServ/1.0
Last-Modified: Mon, 30 Apr 2001 23:02:36 GMT
ETag: "5b381-ec-3aedef0c"
Accept-Ranges: bytes
Content-Length: 236
Connection: close
Content-Type: text/html

<html>
<head>
<title>My web page</title>
</head>
<body>
Hello world!
</body>
</html>
```

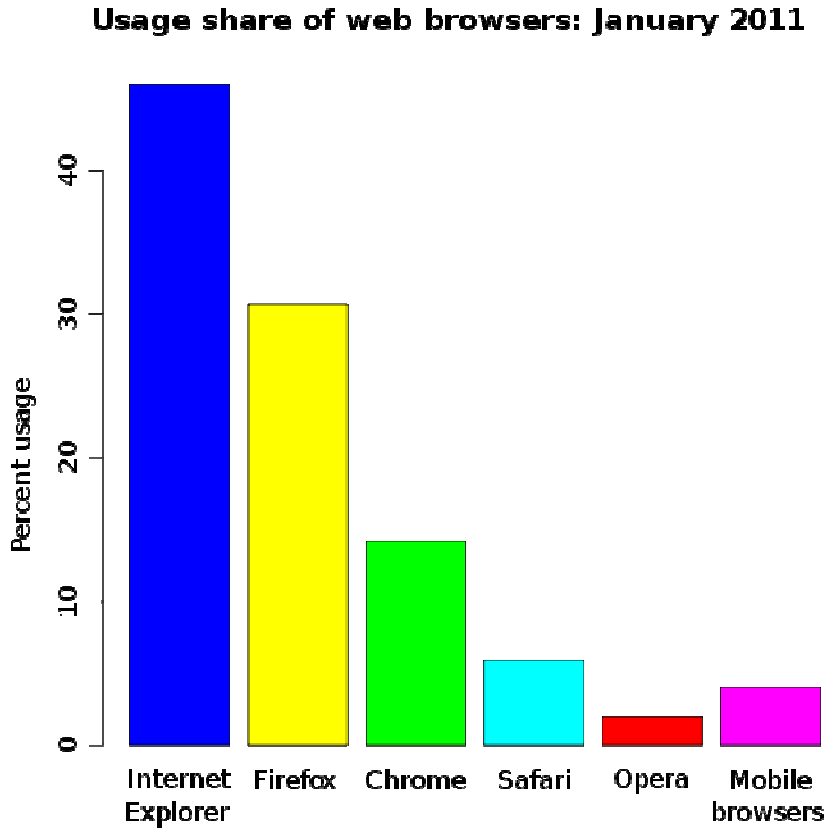
HTTP servers

- Source: netcraft.com



HTTP Browsers

- Source: en.wikipedia.org/wiki/Usage_share_of_web_browsers

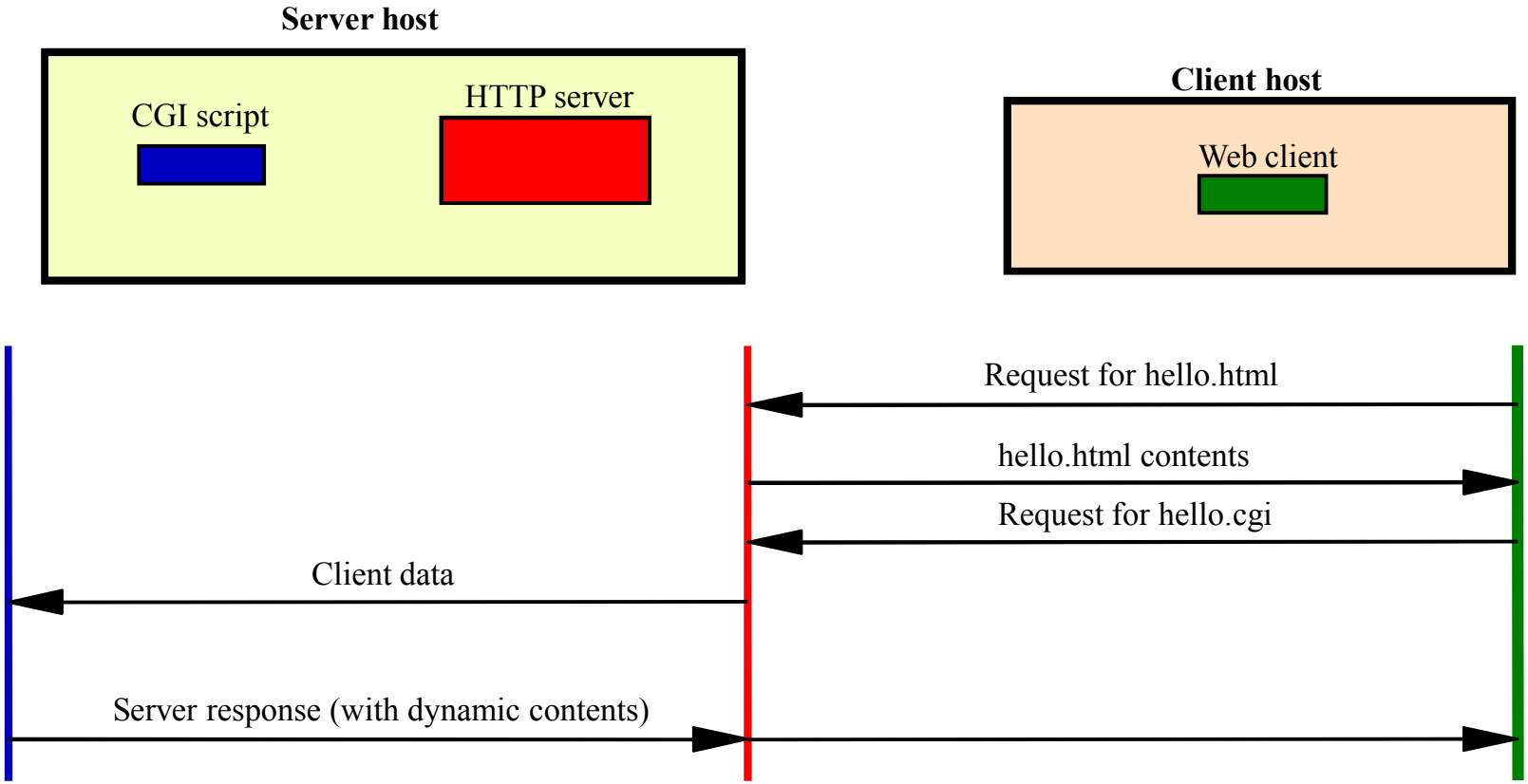


Types of web pages

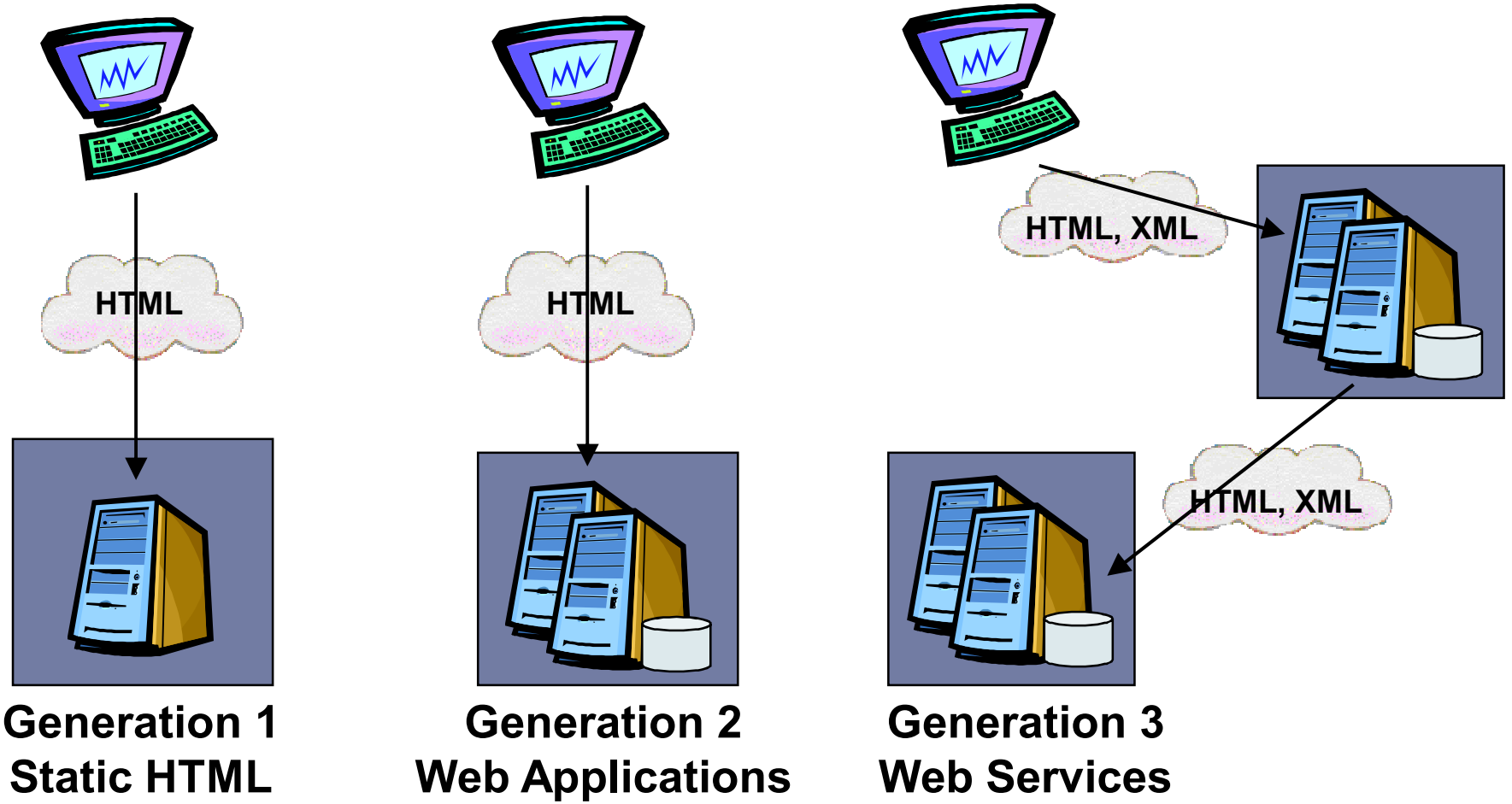
- Static documents
- Dynamic web pages
  - Created on client-side
    - Javascript
    - Applets
  - Created on server-side
    - CGI
    - PHP
    - ASP
    - Servlets

CGI (*Common Gateway Interface*)

- A CGI program can be written in a:
  - Programming language: C, Ada, C++, Fortran.
  - *Scripting* language: Perl, Tkl, Python.

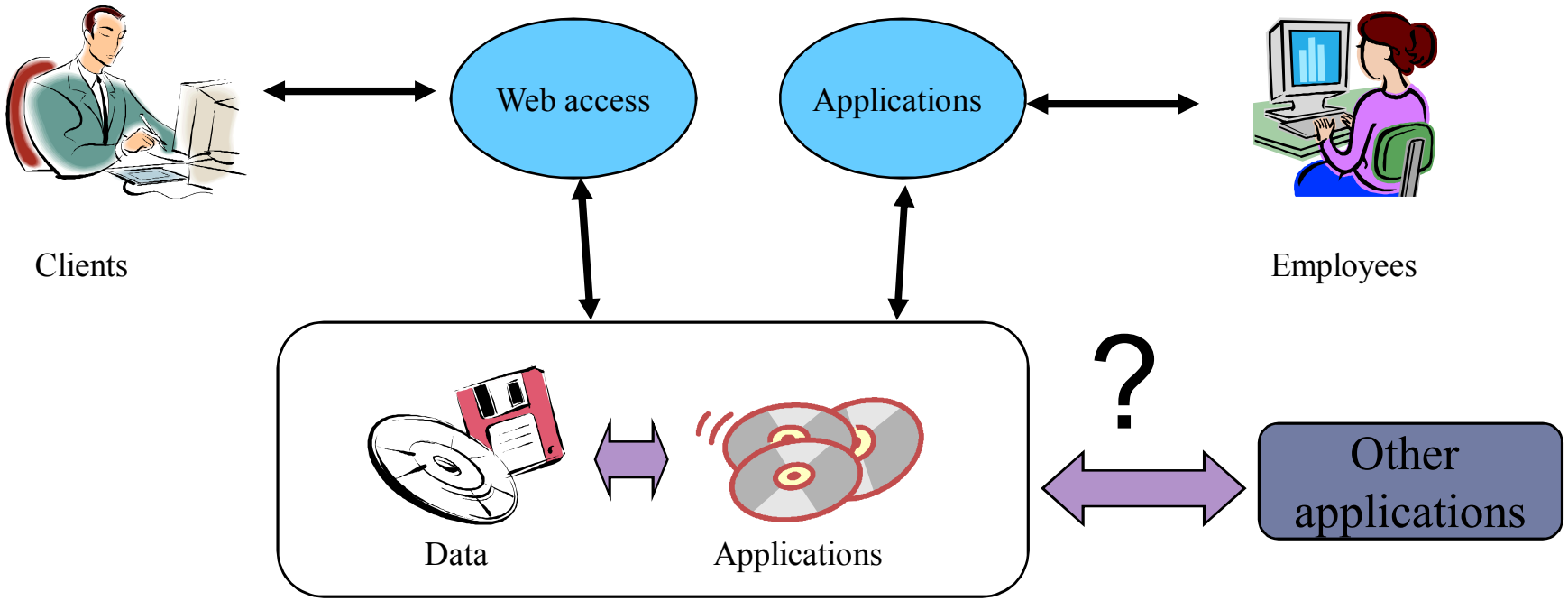


Web evolution



Problems of traditional web-based developments

- Several technologies:
  - Applets, CGI, Scripting languages, etc.
- Developments strongly oriented to user interaction

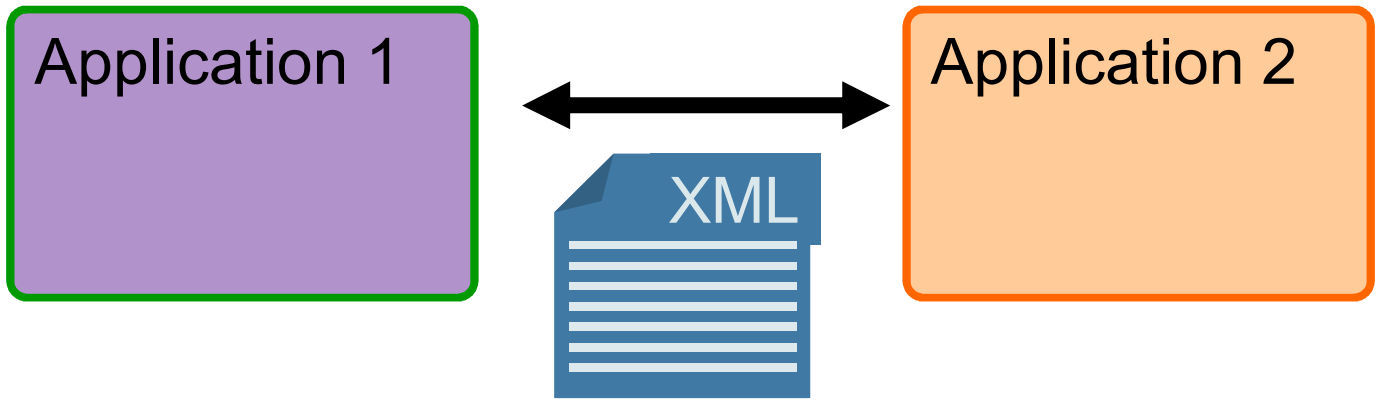


Web Services

- To adapt the web programming model (loosely coupled) to be used with applications not based on a web browser.
- The objetive is to offer a plataform for building distributed applications using software:
  - That executes in different OS and architectures
  - Written with different languages and programming tools
  - Developed independently

What are Web Services?

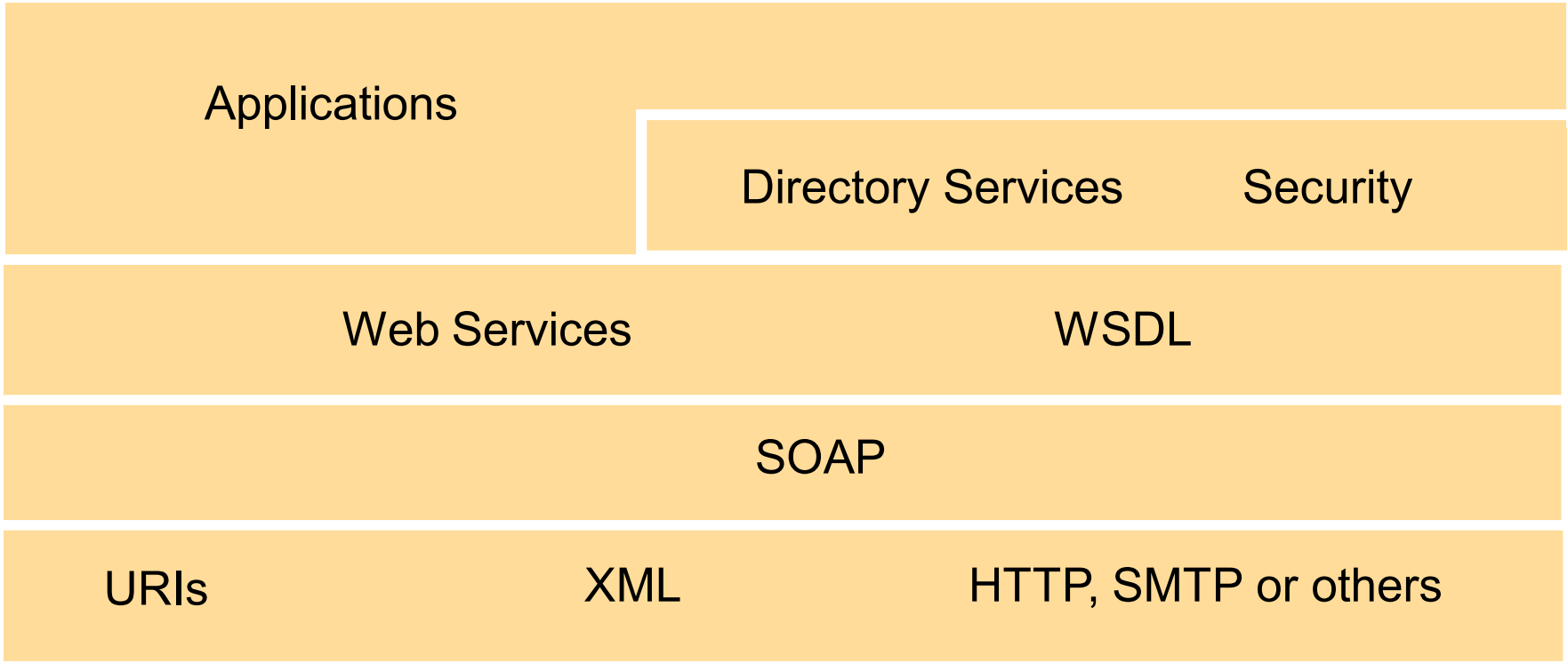
- Technology that allows accessing services (applications) on the Internet
- Interface and mechanism that allow clients to interact with servers
- Its definition can be discovered by other software systems.
- Standardization controlled by a W3C group:
  - http://www.w3.org/2002/ws/



Interface and operations

- A web service interface consists of a set of operations for being used by clients on the Internet
  - Web services are not specific of HTTP.
- Web services operations can be offered by:
  - Programs, objets, data bases
- A web service can be managed by:
  - A traditional web server
  - An independent server

Components and infrastructure



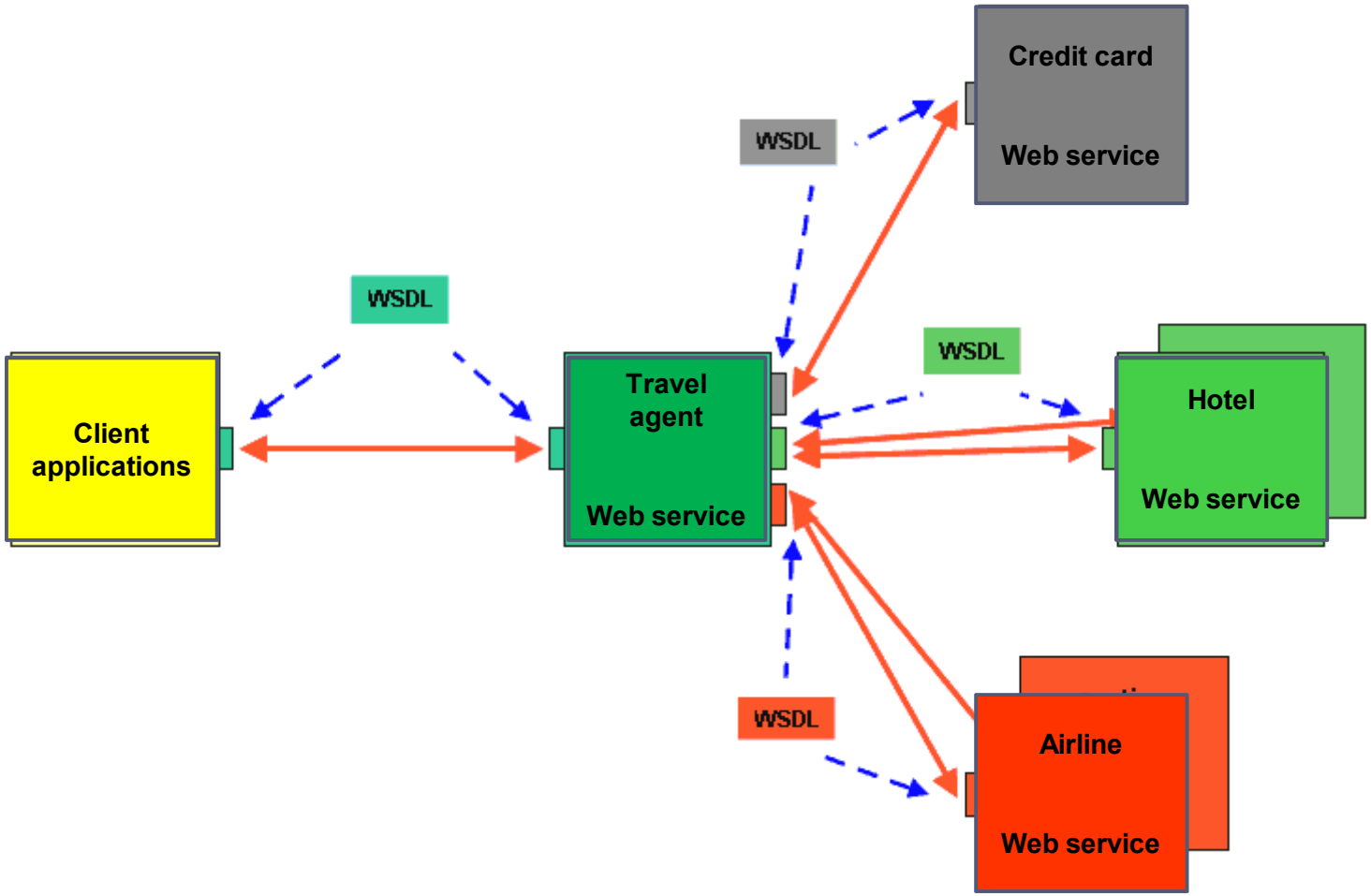
Basic design principles

- Interoperability in heterogeneous environments
- Combination of web services
- Different communication patterns
- Independency of programming model
- Representation of messages
- Referencies of services
- Activation of services
- Transparency

Interoperability in heterogeneous environments

- Services based on standard protocols and mechanisms
  - HTTP: transport used
  - SOAP: packs information and transmits it between client and service provider
  - XML: describes information, messages
  - UDDI: list of available services
  - WSDL: service description
- Advantages:
  - Crosses firewalls
    - Difficult in other environments like Java RMI or CORBA

Combination of Web Services



Source: www.w3c.es

Different communication patterns

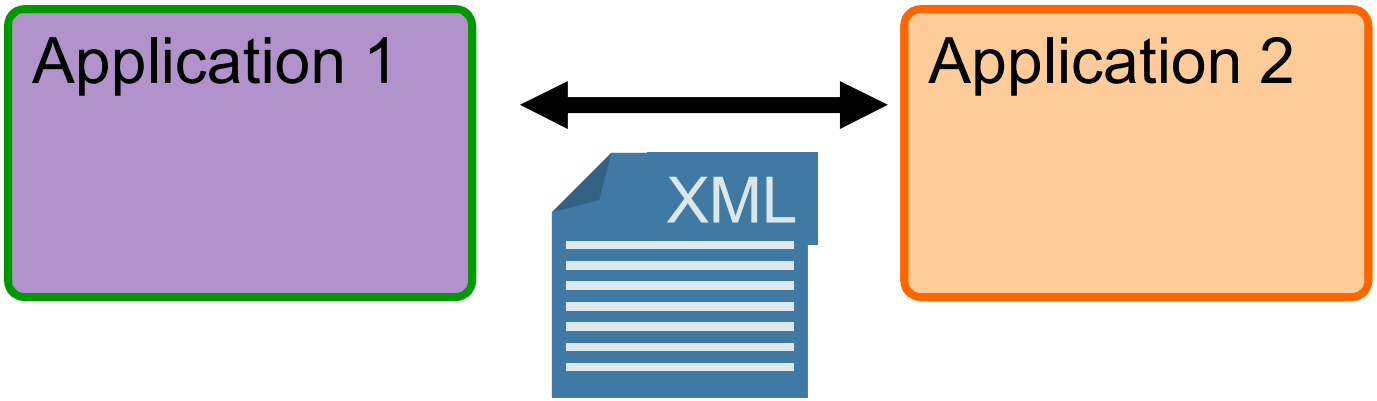
- Synchronous request-response pattern
  - RPC-based models
- Asynchronous messages

Independency of programming model

- Web Services offer a computation model on the Internet
- Independency of programming language
- Independency of programming model

Messages representation

- SOAP messages and data are represented in XML



Referencies of services

- Each Web Service has a URI (*Uniform Resource Identifier)*
  - URL (*Uniform Resource Locator*)
    - Includes resource localization (*hostname+pathname*)
  - URN (*Uniform Resource Name*)
    - Resource names that does not include localization
- Clients use URIs to refer to services

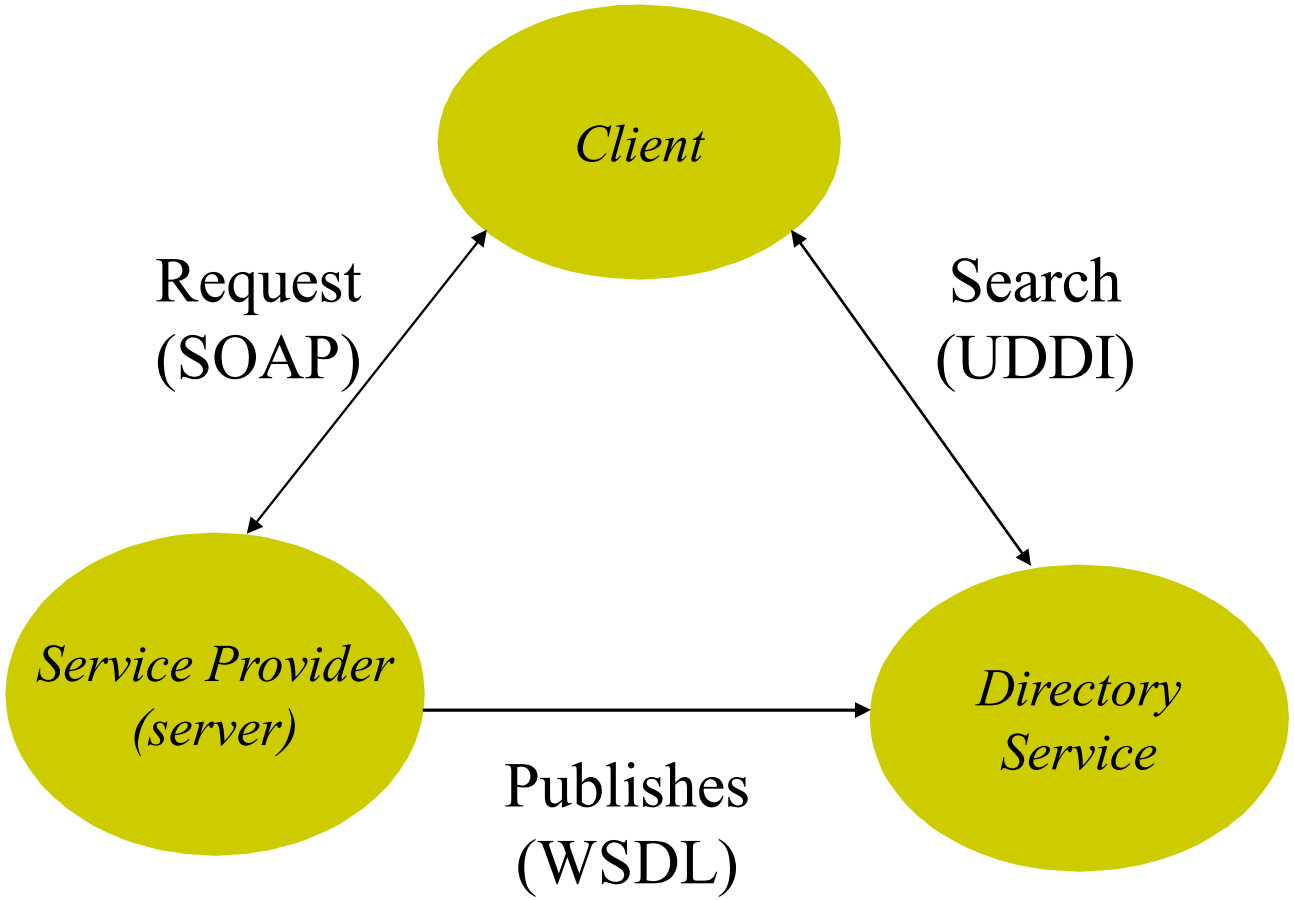
Activación de servicios

- El servicio web es solicitado al computador identificado en la URL
  - El servicio Web puede residir en ese computador
  - O en otro computador
    - Mejora las prestaciones
- Tipos de activación
  - El servicio web se activa bajo demanda
  - El servicio web ejecuta continuamente

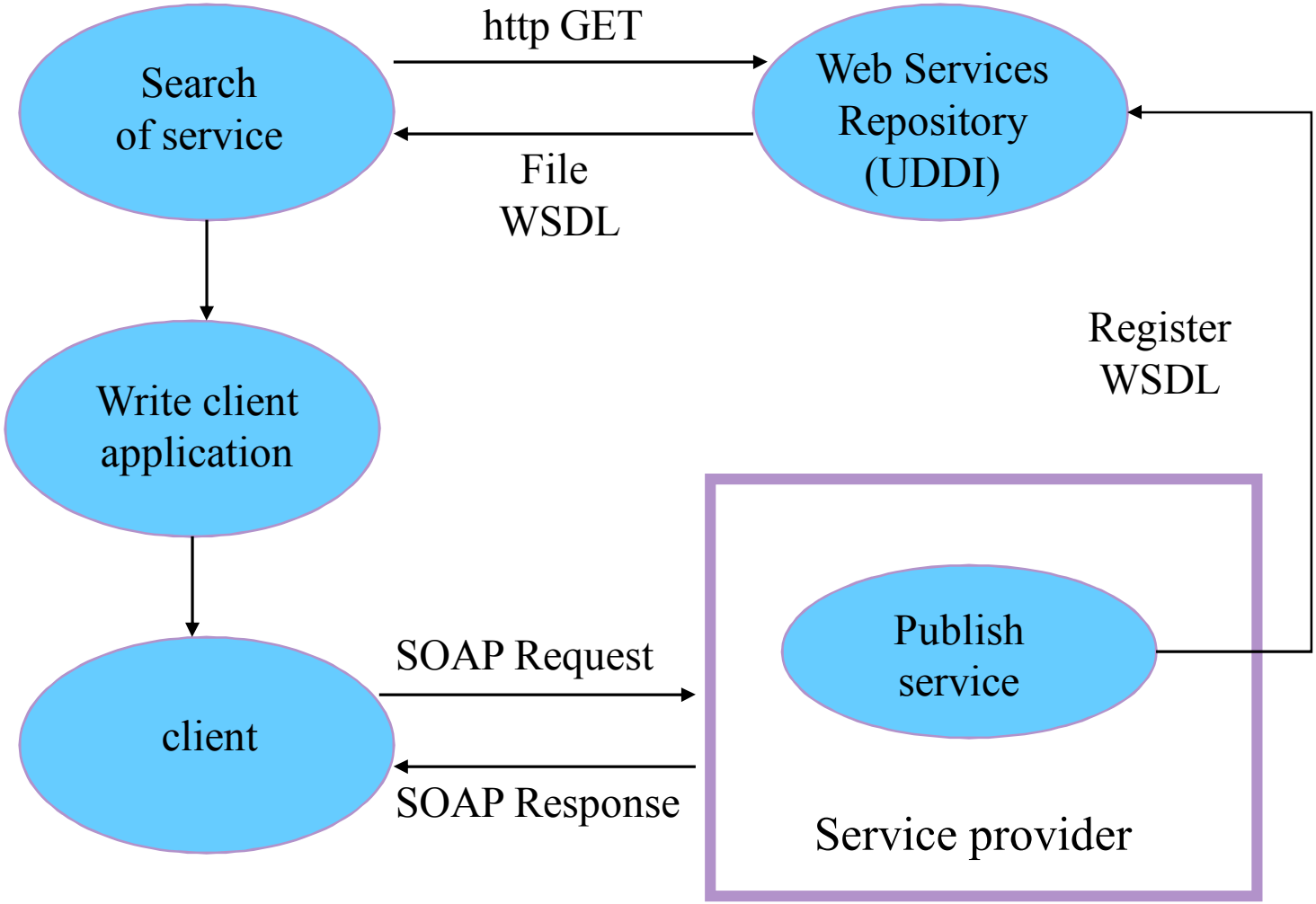
Use of Web Services

- Direct use in clients and servers of SOAP and XML (difficult to use)
- Use by means of an API that hides SOAP and XML details
- Ways of calling:
  - Static invocation by means of a *proxy* or *stub*
  - Dynamic invocation: generic operation that converts call and arguments to SOAP and XML dynamically

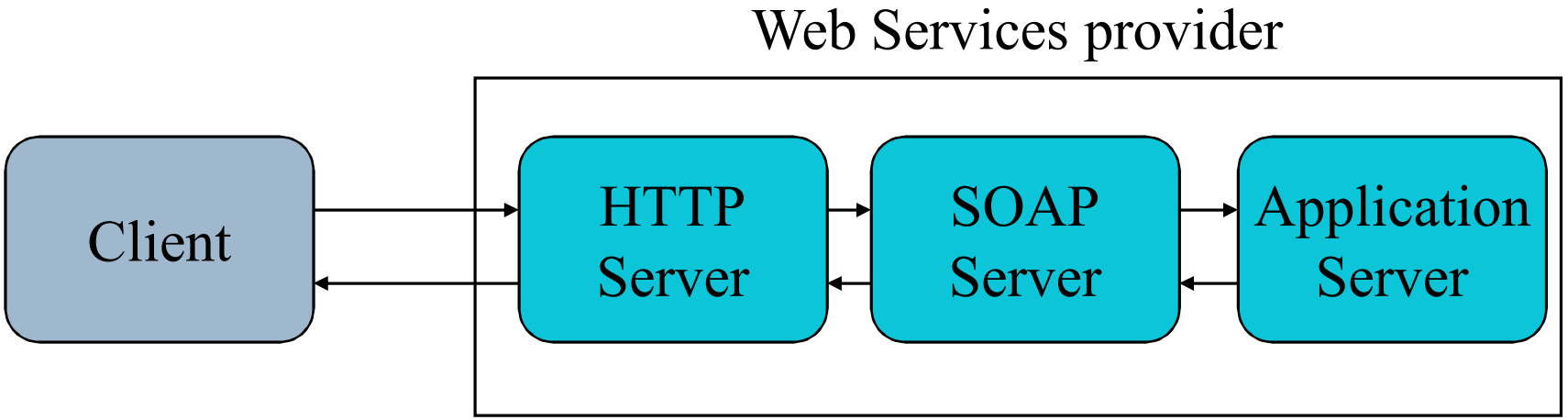
Web Services and SOA



Use case


Search
of service
http GET
File
WSDL
Web Services
Repository
(UDDI)
Register
WSDL
Write client
application
Publish
service
client
SOAP Request
SOAP Response
Service provider

Example of implementation



XML

- *eXtensible Markup Language*
  - Defined by W3C (www.w3c.org)
- XML is extensible, it allows users to define their own labels (not like HTTP)
- Components:
  - Elements and attributes
    - *<tag attr=value/>*
    - *<tag>value</tag>*
    - *Example:*

http://www.dataaccess.com/webservicesserver/numberconversion.wso?WSDL

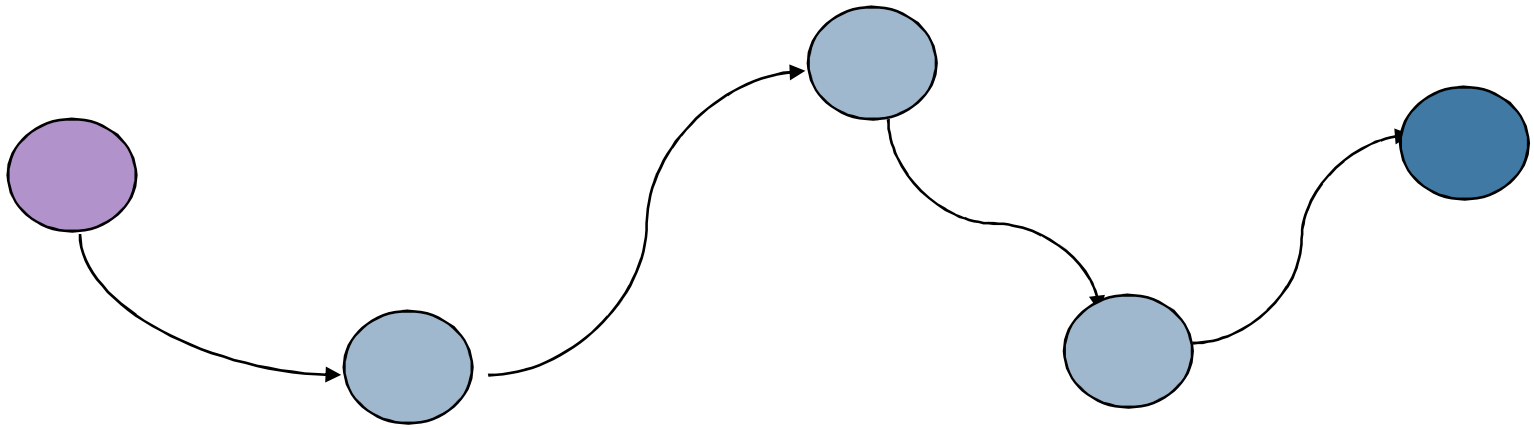
  - Namespace
    - *xmlns=http://www.w3.org/1999/xhtml*
  - Schemas
    - Elements and attributes that are allowed to appear in a document

SOAP

- *Simple Object Accces Protocol*
  - (*www.w3.org*)
- SOAP specifies:
  - How to represent messages in XML
  - How to combine SOAP messages for a request-response model
  - How to process message elements
  - How to use the transport (HTTP, SMTP, …) to send SOAP messages

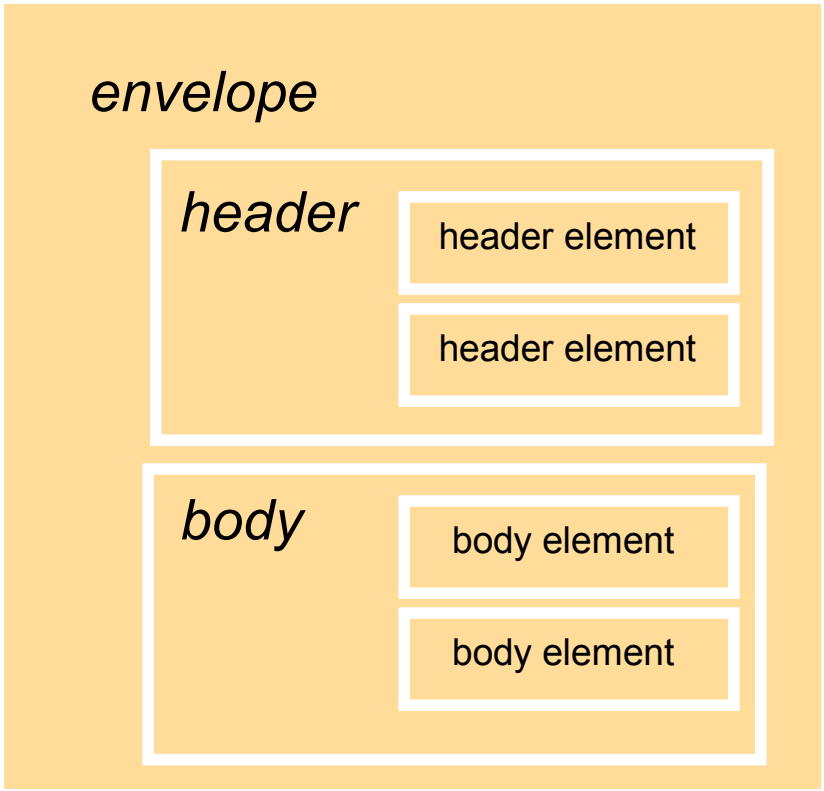
SOAP node

- Node that transmits, receives, processes and responds SOAP mssages
  - SOAP sender
  - SOAP receiver
  - Intermediary



SOAP messages

- Basic communication unit among SOAP nodes
- The message is transported in an *envelope*
  - Optional header
  - Body
- Previous XML elements are defined as a schema in the XML namespace
  - Schema defined in www.w3.org



Header

- Optional element
- Includes control information
  - Transaction Identifier to be used in a transactions service
  - Message identifier to relate massages among them (services are autonomous and independent among them)
  - User name, public key, etc.

Communication model

- Send of messages, the *body* element *includes:*
  - Message
  - Reference to the XML schema that describes the service
- Client-server comunication (RPC)
  - The *body* element contains a *request* or a *response*

Example

- Ex: *float GetPrice(string item);*

Request:

```
<GetPrice>
    <item>table</item>
</GetPrice >
```

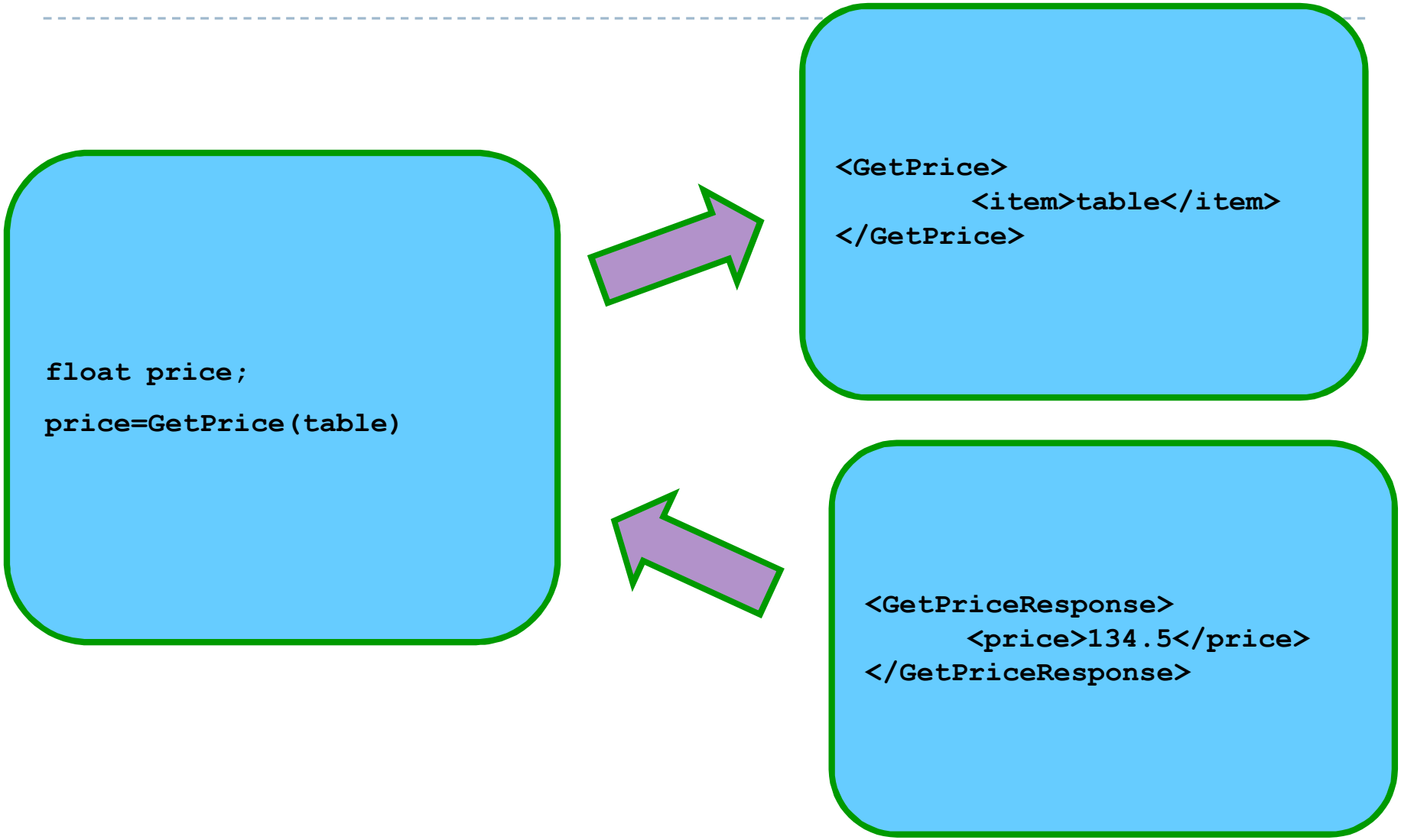
Respuesta:

```
<GetPriceResponse>
    <Price>134.5</Price>
</GetPriceResponse>
```

Schema:

```
<element name="GetPrice">
<complexType><all>
    <element name="item" type="string"/>
</all></complexType>
</element>
<element name="GetPriceResponse">
<complexType><all>
    <element name="Price" type="float"/>
</all></complexType>
</element>
```

Serialization



Transport of SOAP messages

- HTTP protocol
  - RPC style:
    - Request: in HTTP POST
    - Response: in the response to the POST
  - Sending of information:
    - With HTTP POST
    - With HTTP GET
- SMTP protocol
  - The specification indicates how to pack SOAP messages in SMTP messages

Example of request/response

```
POST /StockQuote HTTP/1.1
.....................
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
      <m:GetPrice xmlns:m="http://example.com/stockquote.xsd">
            <item>table</item>
      </m:GetPrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request

```
HTTP/1.1 200 OK
.............
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  <SOAP-ENV:Body>
    <m:GetPriceResponse xmlns:m="http://example.com/stockquote.xsd">
      <Price>134.5</Price>
    </m:GetPriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

More examples

- http://www.xmethods.com/

WSDL

- Web Services Description Language
  - IDL for Web Servicios in XML
- Written in XML
- WSDL is a XML document
- It is used for:
  - Describe Web Services
  - Localizate Web Services
- WSDL is not a W3C standard yet (*draft*)

Web Services Description

- Describes the Web Service
- Specifies the service localization
- Specifies the service operations and methods
- Usually automatically generated from services code

Web Services Description: example

- It is usually generated, it produces:

string NumberToWords (unsignedLong ubiNum)

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://www.dataaccess.com/webservicesserver/" name="Conversions"
                  targetNamespace="http://www.dataaccess.com/webservicesserver/">
  <types>
    <xs:schema elementFormDefault="qualified" targetNamespace="http://www.dataaccess.com/webservicesserver/">
      <xs:element name="NumberToWords">
<xs:complexType>
          <xs:sequence>
            <xs:element name="ubiNum" type="xs:unsignedLong"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="NumberToWordsResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="NumberToWordsResult" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </types>
<message name="NumberToWordsSoapRequest">
    <part name="parameters" element="tns:NumberToWords"/>
  </message>
  <message name="NumberToWordsSoapResponse">
    <part name="parameters" element="tns:NumberToWordsResponse"/>
  </message>
  <portType name="ConversionsSoapType">
    <operation name="NumberToWords">
      <documentation>Returns the word corresponding to the positive number passed as parameter. Limited to quadrillions.</documentation>
      <input message="tns:NumberToWordsSoapRequest"/>
      <output message="tns:NumberToWordsSoapResponse"/>
    </operation>
  </portType>
  <binding name="ConversionsSoapBinding" type="tns:ConversionsSoapType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="NumberToWords">
      <soap:operation soapAction="" style="document"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="Conversions">
    <documentation>The Conversion Visual DataFlex Web Service will provide different conversion functions. The function currently available will help you converting numbers into words.</documentation>
    <port name="ConversionsSoap" binding="tns:ConversionsSoapBinding">
      <soap:address location="http://www.dataaccess.com/webservicesserver/conversions.wso"/>
    </port>
  </service>
</definitions>
```

Web Services Description: example

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://www.dataaccess.com/webservicesserver/" name="Conversions"
        targetNamespace="http://www.dataaccess.com/webservicesserver/">
 <types>
  <xs:schema elementFormDefault="qualified" targetNamespace="http://www.dataaccess.com/webservicesserver/">
   <xs:element name="NumberToWords">
   <xs:complexType>
     <xs:sequence>
      <xs:element name="ubiNum" type="xs:unsignedLong"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="NumberToWordsResponse">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="NumberToWordsResult" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:schema>
 </types>
```

```
<message name="NumberToWordsSoapRequest">
  <part name="parameters" element="tns:NumberToWords"/>
 </message>
 <message name="NumberToWordsSoapResponse">
  <part name="parameters" element="tns:NumberToWordsResponse"/>
 </message>
 <portType name="ConversionsSoapType">
  <operation name="NumberToWords">
   <documentation>Returns the word corresponding to the positive number passed as parameter. Limited to
    quadrillions.</documentation>
   <input message="tns:NumberToWordsSoapRequest"/>
   <output message="tns:NumberToWordsSoapResponse"/>
  </operation>
 </portType>
```

```
<binding name="ConversionsSoapBinding" type="tns:ConversionsSoapType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="NumberToWords">
    <soap:operation soapAction="" style="document"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
<service name="Conversions">
  <documentation>The Conversion Visual DataFlex Web Service will provide different conversion functions. The function
    currently available will help you converting numbers into words.</documentation>
  <port name="ConversionsSoap" binding="tns:ConversionsSoapBinding">
    <soap:address location="http://www.dataaccess.com/webservicesserver/conversions.wso"/>
  </port>
</service>
</definitions>
```

Structure of a WSDL document

<definitions>
  <types>
    types definition (independent of language)
  </types>
  <message>
    messages definition (to be exchanged)
  </message>
  <portType>
    ports definition (function interfaces, including parameters, etc.)
  </portType>
  <binding>
    bindings definition (messages and data formats)
  </binding>
  <services>
    services definition (service name and one or more ports where they are)
  </services>
</definitions>

Elements

- **Types:** definition of language-independent types
- **Messages:** definition of kind of messages to be exchanged
- **PortTypes (interfaces):** definition of function interfaces (operation name, input and output parameters)
- **Bindings:** specification of messages and data to be used formats
- **Services:** specification of service name and one or more places *(ports) to find the service.*

Namespaces

- Objetive: to avoid conflicts
  - Two different web services A and B that have a common element f.
- Each instance of f can be refered to by A:f or B:f
- http://www.w3.org/2001/XMLSchema.

Web Service development

- Service library programming
  - In some environments it is necessary to include specific information
    - In VisualStudio .Net: label *[WebMethod]* over exported methods
- Automatic generation of WSDL file
  - Usually inside service application generation
    - In VisualStudio .Net: proyect of type *Web Service*
- In server: WSDL file tells about how to activate the service
  - It is usually done by a web server with web services support
- Client development:
  - Obtain WSDL file and generate a proxy for client application
    - In VisualStudio .Net: "*Add Web Reference*"

UDDI

- *Universal Description, Discovery, and Integration*
  - Not standard: initial proposal by Microsoft, IBM y Ariba
- Distributed register of web serviecs offered by companies
- Information classified in three cathegories (guides):
  - White pages: Company's data
  - Yellow pages: Clasification by activity type
  - Green pages: Description of web services (WSDL)
- It is also accessed as a web service
- Can be checked in development-time or even dynamically in run-time
- Allows searches against different criteria
  - Type of activity, type of service, geographical location

Web Services examples

Google: http://www.google.com/apis

- For applications that perform searches on the Internet

- Amazon:
  - http://simplest-shop.com/camera
- Mapquest
- UPS, Fedex, etc.

Web services from Java

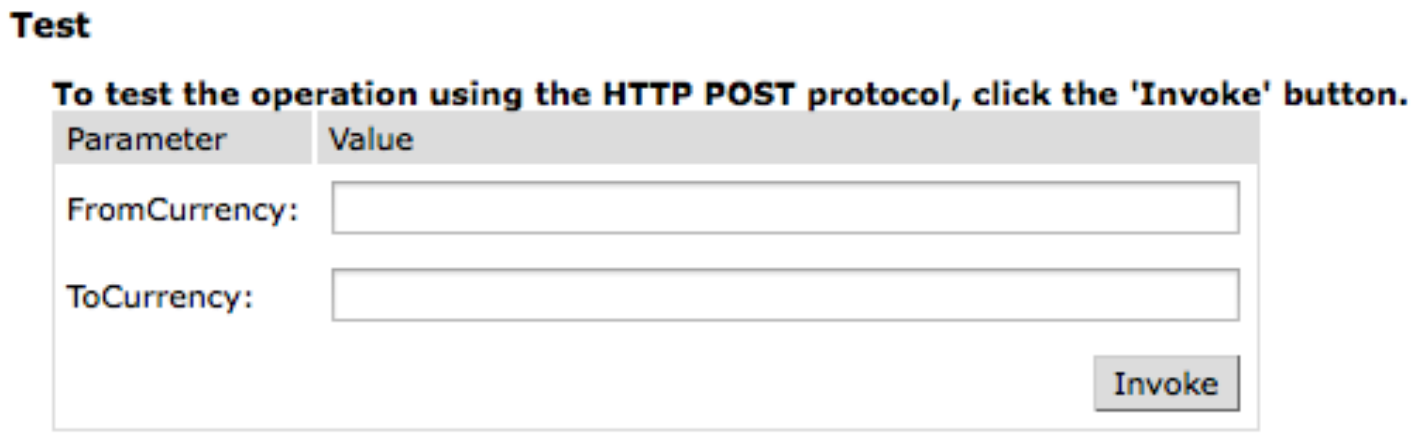
- A lot of alternatives
- Wsimport
  - Usage: wsimport –keep file.wsdl
- Create all the needed classes for invoking remote web services (stubs)

Generated classes

- Class PortType. A class that has the same name as the name attribute of the wsdl porttype element and contains a method for each operation defined with operation elements.
- Class Service. A class that has the same name as the name attribute of the wsdl service. This class accesses the web service and allows PortType instantiate the class.
- For each operation defined in the portType label:
  - As many classes as necessary to fill Input
  - As many classes as necessary to return the result of the operation
- Class ObjectFactory. Esta clase facilita la instanciación interna de las clases input y response.
- Class package-info. Annotate the Java package for each object created from the xsd of the wsdl (package metadata).

Example

- CurrencyConvertor service from http://www.webservicex.net
- WSDL:
- http://www.webservicex.net/CurrencyConvertor.asmx?WSDL
- Function:
- ConversionRate



Stubs creation

- wsimport –keep http://www.webservicex.net/CurrencyConvertor.asmx?WSDL
- Creates a folder net/webservicex with the following files:
  - ConversionRate.java
  - ConversionRateResponse.java
  - CurrencyConvertor.java
  - CurrencyConvertorSoap.java
  - Currency.java
  - ObjectFactory.java
  - package-info.java

Client example in Java

```
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import  net.webservicex.*;

public class test{
        public static void main(String[] args) throws Exception {
        CurrencyConvertor myservice= new CurrencyConvertor();
        CurrencyConvertorSoap port =  myservice.getCurrencyConvertorSoap();

        Currency from    = Currency.fromValue("USD");
        Currency to      = Currency.fromValue("EUR");

        double d = port.conversionRate(from, to);
        System.out.println("de USD a ERU " + d );
    }
 }
```

Example

- Servicio GlobalWeather from http://www.webservicex.net
- WSDL:
- http://www.webservicex.net/globalweather.asmx?WSDL

- Functions
- GetCitiesByCountry
- GetWeather

Example in Java

```
import java.util.List;
import net.webservicex.*;

public class WeatherClient {
    public static void main (String[] args) {

        GlobalWeather myservice =  new GlobalWeather();
        GlobalWeatherSoap port = myservice.getGlobalWeatherSoap();

        String data = port.getWeather("Madrid", "Spain");

        System.out.println(data);
    }
}
```

Example in Java

```
import j
import n

public c
    publ

    }
}
```

```
<?xml version="1.0" encoding="utf-16"?>
<CurrentWeather>
  <Location>Madrid / Cuatro Vientos, Spain (LEVS) 40-23N 003-47W 687M</Location>
  <Time>Apr 09, 2015 - 09:30 AM EDT / 2015.04.09 1330 UTC</Time>
  <Wind> from the ENE (060 degrees) at 6 MPH (5 KT) (direction variable):0</Wind>
  <Visibility> 4 mile(s):0</Visibility>
  <SkyConditions> overcast</SkyConditions>
  <Temperature> 44 F (7 C)</Temperature>
  <DewPoint> 42 F (6 C)</DewPoint>
  <RelativeHumidity> 93%</RelativeHumidity>
  <Pressure> 30.27 in. Hg (1025 hPa)</Pressure>
  <Status>Success</Status>
</CurrentWeather>
```

More information

- Main Page http://www.w3.org/
- UDDI Page http://www.uddi.org/
- Tutorials about SOAP, WSDL, and other web technologies:
  - http://www.w3schools.com/

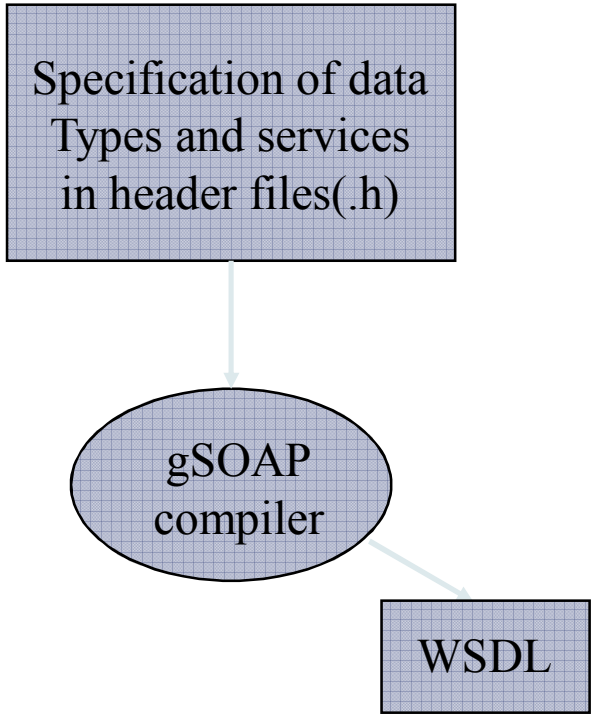
Development environments

- Growing number of development environments
- Some interesting implementations:
  - gSOAP
  - .Net de Microsoft
  - *Web Services Project* from Apache
  - *Java Web Services Developer Pack*
  - *IBM WebSphere SDK for Web services* (WSDK)
  - WASP from Systinet
  - JOnAS
  - AXIS

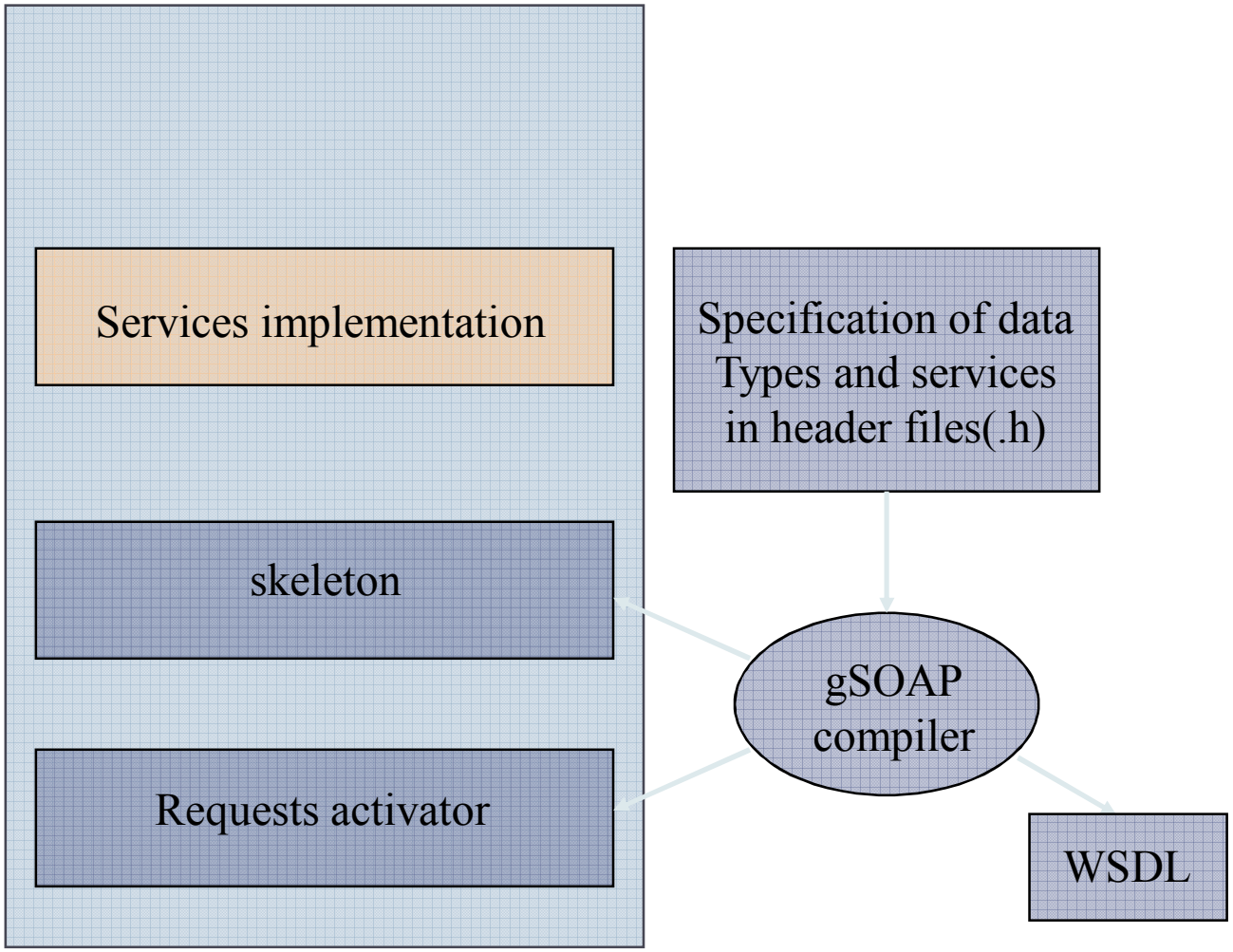
gSOAP

- Set of tools to develop applications in C/C++ based on web services
  - http://www.cs.fsu.edu/~engelen/soap.html

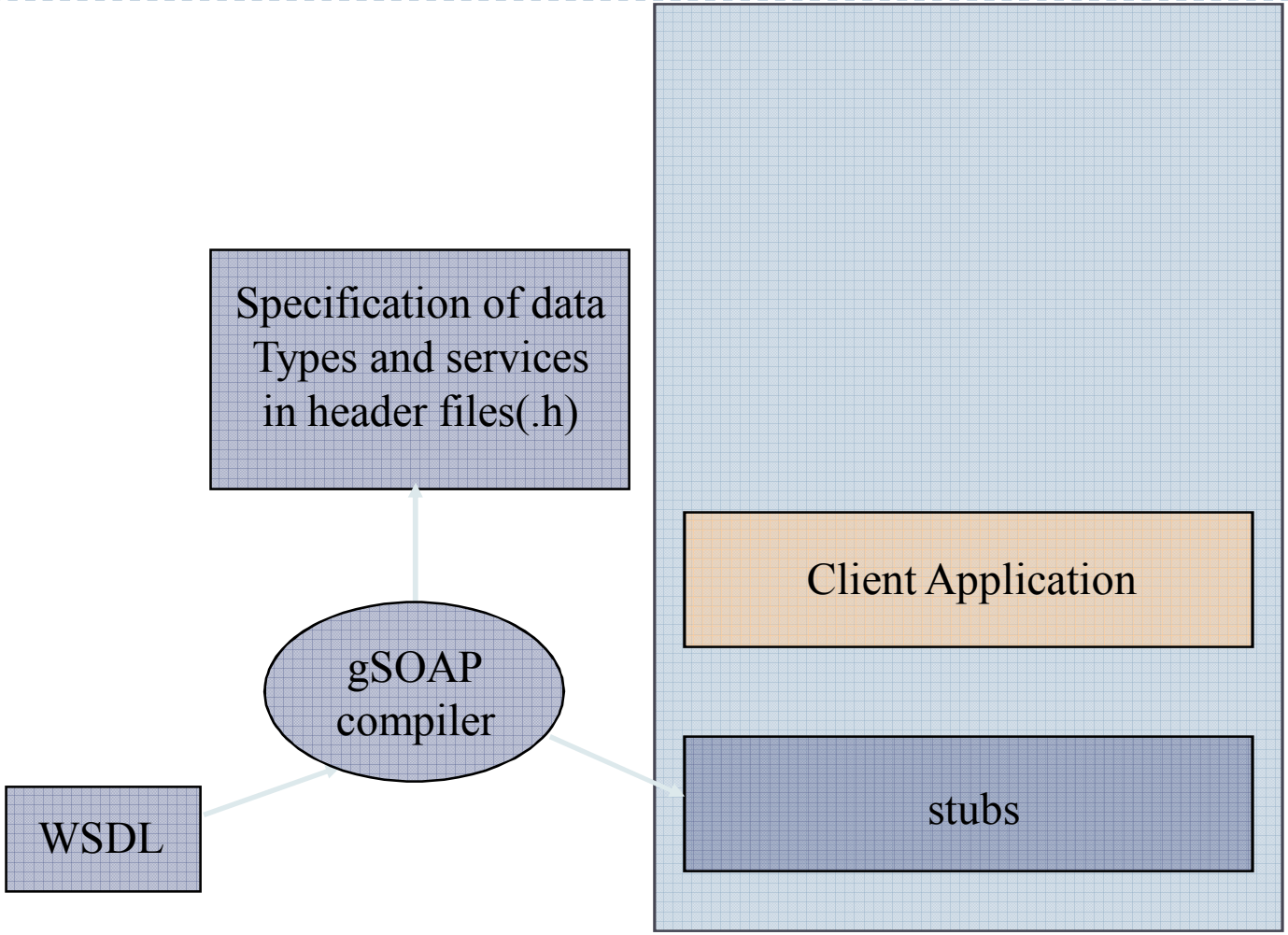
Application developmnt



Server development



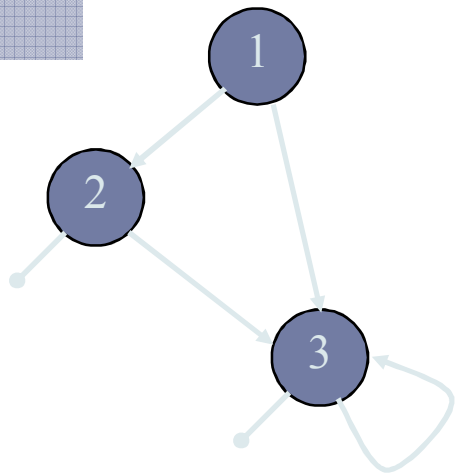
Client development



Serialization/deserialization

- gSOAP is responsible of all the process of XML serialization and deserialization

```
struct BG
{ int val;
  struct BG *left;
  struct BG *right;
};
```




```
<BG>
 <val>1</val>
 <left>
  <val>2</val>
  <right href="#X"/>
 </left>
 <right href="#X"/>
</BG>
<id id="X">
 <val>3</val>
 <right href="#X"/>
</id>
```

Example

- WSDL:
  - http://www.dataaccess.com/webservicesserver/numberconversion.wso?WSDL
  - Accesible from http://www.xmethods.com/

- Two methods

  string **NumberToWords**(unsignedLong ubiNum)
  // Returns the word corresponding to the positive number passed as parameter. Limited to quadrillions.

  string **NumberToDollars**(decimal dNum)

  // Returns the non-zero dollar amount of the passed number

WSDL elements

```
<operation name="NumberToWords">
<documentation>Returns the word corresponding to the positive
 number passed as parameter. Limited to quadrillions.
 </documentation>
<input  message="tns:NumberToWordsSoapRequest" />
<output message="tns:NumberToWordsSoapResponse" />

<message name="NumberToWordsSoapRequest">
<part name="parameters" element="tns:NumberToWords" />
</message>

<message name="NumberToWordsSoapResponse">
<part name="parameters"
 element="tns:NumberToWordsResponse" />
</message>
```

WSDL elements

```
<xs:element name="NumberToWords">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ubiNum" type="xs:unsignedLong" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="NumberToWordsResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="NumberToWordsResult"
  type="xs:string" />
    </xs:sequence>
  </xs:complexType>
 </xs:element>
```

Client generation

- Step 1: Obtain the header file from a WSDL:
  - *wsdl2h -o conversions.h http://www.dataaccess.com/webservicesserver/numberconversion.wso?WSDL*

```
int soap_call___ns1__NumberToWords(
    struct soap *soap,
    NULL, // char *endpoint = NULL selects default endpoint for this operation
    NULL, // char *action   = NULL selects default action for this operation

    // request parameters:
    struct _ns1__NumberToWords           * ns1__NumberToWords,

    // response parameters:
    struct _ns1__NumberToWordsResponse * ns1__NumberToWordsResponse
 );
```

Client generation

```
struct _ns1__NumberToWords
{
    /// Element ubiNum of type xs:unsignedLong.
    ULONG64          ubiNum     1;      ///< Required element.
};


struct _ns1__NumberToWordsResponse
{
   /// Element NumberToWordsResult of type xs:string.
   char*  NumberToWordsResult          1;  ///< Required element.
};
```

Client generation

- Step 2: stub and skeletons generation
  - *soapcpp2 –C -c conversions.h*
  - Generates stub for client

Client development

```
#include "soapH.h" // obtain the generated stub
#include "ConversionsSoapBinding.nsmap"

main(int argc, char **argv)
{
   int err;
   struct _ns1__NumberToWords  arg1;
   struct _ns1__NumberToWordsResponse arg2;

   struct soap *soap = soap_new();

   arg1.ubiNum = atoi(argv[1]); // argument

   err = soap_call___ns1__NumberToWords (soap, NULL, NULL,
                                         &arg1, &arg2)
  if (err == SOAP_OK)
      printf("Result = %s\n", arg2.NumberToWordsResult);
   else // an error occurred
      soap_print_fault(soap, stderr);
}
```

Compilation and execution

- Files to compile:
  - client.c
  - soapC.c
  - soapClient.c

```
./client 34
> Result = thirty four
```

Server and client development

- calc.h

```
int ns__add (int a, int b, int *res);
int ns__subtract (int a, int b, int *res);
```

- Step 1: stubs and skeletons generation:
  - *soapcpp2 -c calc.h*

Step 2: server development

```
#include "soapH.h"
#include "ns.nsmap"

int main(int argc, char **argv)
{ int m, s; /* master and slave sockets */
  struct soap soap;
  soap_init(&soap);
  if (argc < 2)
    soap_serve(&soap);  /* serve as CGI application */
  else
  { m = soap_bind(&soap, NULL, atoi(argv[1]), 100);
    if (m < 0)
    { soap_print_fault(&soap, stderr);
      exit(-1);
    }
    fprintf(stderr, "Socket connection successful: master
  socket = %d\n", m);
```

Step 2: server development

```
for ( ; ; )
{
  s = soap_accept(&soap);
      fprintf(stderr, "Socket connection successful:
                      slave socket = %d\n", s);

       if (s < 0)
      { soap_print_fault(&soap, stderr);
        exit(-1);
      }

      soap_serve(&soap);
      soap_end(&soap);
    }
  }
  return 0;
}
```

Step 2: server development

```
int ns__add (struct soap *soap, int a, int b, int *res)
{   *res = a + b;
  return SOAP_OK;
}
```

```
int ns__subtract (struct soap *soap, int a, int b, int *res)
{   *res = a - b;
  return SOAP_OK;
}
```

Step 3: client development

```
#include "soapH.h"
#include "ns.nsmap"

//const char server[] =
   "http://websrv.cs.fsu.edu/~engelen/calcserver.cgi";
//const char server[] = "http://localhost:9000";

int main(int argc, char **argv)
{
  struct soap soap;
  char *server;
  int a, b, res;

  if (argc != 2) {
    printf("Use: calcClient   http://server:port\n");
    exit(0);
  }

  soap_init(&soap);
```

Step 3: client development

```
 server = argv[1];

  a = 5;      b = 7;

 soap_call_ns__add (&soap, server, "", a, b, &res);

 if (soap.error)
  {
        soap_print_fault(&soap, stderr);
        exit(1);
  }
  printf("Result = %d \n", res);

  soap_destroy(&soap);
  soap_end(&soap);
  soap_done(&soap);
  return 0;
}
```

Step 4: compilation and execution

- Client:
  - calcClient: calcClient.c soapC.c soapClient.c
- Server:
  - calcServer: calcServer.c soapC.c soapServer.c

- >./calcServer 9000

- > ./calcClient http://localhost:9000

Web Services and gSOAP

Computer Architecture Area (ARCOS)

Distributed Systems

Bachelor in Informatics Engineering

Universidad Carlos III de Madrid