Sistemi Distribuiti
Corso di Laurea in Ingegneria

Prof. Paolo Nesi

Ing. Antonio Cappuccio

Parte: 2a – Web Services & REST

Department of Systems and Informatics

University of Florence

Via S. Marta 3, 50139, Firenze, Italy

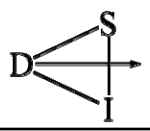
tel: +39-055-4796523, fax: +39-055-4796363

Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet

http://www.disit.dsi.unifi.it/

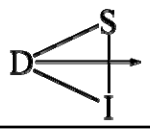
nesi@dsi.unifi.it paolo.nesi@unifi.it

http://www.dsi.unifi.it/~nesi, http://www.axmedis.org



Web Services and REST

- Introduzione e contesto
- WS aims
- Architetture e Protocolli
- Gli standard dei WS
- SOAP Call
- Costruzione applicazioni SOAP-based
- Architetture basate su WS
- Architettura REST

WEB Evolution

- From simple to frame, etc.. , GUI reengineering
- From simple to interactive models,.. active pages, etc.
- From two tier to n-tier architecture
- From single to multilingual
- From simple to multimodal for impaired people
- From HTML to multiformat: HTML, WAP, etc.
- From single server to cluster, performance problems
- From poor to certified User Interface

- Personalization, enforcing new technologies
- Hyperlink analysis, restructuring the web site, increasing the WEB efficiency on the basis of the goals
- WEB hosting, delegating maintenance, increasing band
- QoS, quality of service

WEB Classification 1/2

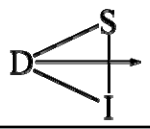
- static: (simple public pages)
  - HTML without or with frames
- interactive: (pages and fill in forms)
  - Forms, CGI, (slow and not flexible, sent the status)
  - Old staff and specific solutions such as NestScape-API, Microsoft ISAPI, etc…
  - Java Script, Visual Basic,
  - ASP, CORBA Plugin, Servlets
  - client server communication managed by the server
    - ➔ each action is managed by the HTML on the client and leads to call the server to exchange messages and commands among the object running on the client
- Dynamic: ……..

WEB Classification 2/2

- dynamic: pages created on the fly, from the database (active server)
  - changing on the basis of the login and user's actions, personalization logic engines
- dynamic technologies:
  - CGI, Common Gateway Interface
  - ASP, ASP.net
  - JavaScript, Java Servlets, Java Server Page
  - PHP, and other solutions, etc.
  - XML for data and communication, SOAP for communication
  - connection with ODBC, JDBC……(SQL)
  - CORBA, JAVA-RMI, Enterprise JavaBeans (Corba), DCOM (distrib COM), .net
  - on server: perl, python, java and C

Architetture WS, SOA

- **SOA: Service Oriented Architecture**
  - Soluzioni distribuite che mettono a disposizione servizi attraverso Web Application che espongono vari WS
- **Specificita'**
  - WS possono chiamare altri WS
  - Si vengono a creare dipendenze evidenti e nascoste fra WS
  - Applicazioni PHP, C++, Java possono chiamare WS
  - Si possono aprire connessioni dirette e passarci dentro SOAP
- **I WS possono** essere implementazioni di procedure con stato o senza stato
  - Lo stato deve essere tipicamente memorizzato all'interno di una database, file o che altro di persistente
  - E' meglio non avere stato sul protocollo ma di fatto alcuni WS lo hanno. Lo stato di protocollo significa avere svariate richieste collegate fra loro che implicano una seguenza di stati non espliciti nella chiamata
  - Si auspicano transazioni atomiche

Paradigma SOA

- Paradigma emergente per lo sviluppo e l'integrazione di applicazioni distribuite
  - L'obiettivo è quello di supportare l'interoperabilità tra componenti in esecuzione su piattaforme diverse
- Paradigma orientato ai servizi
  - L'enfasi è posta sull'interoperabilità tra componenti eterogenei sulla base di protocolli e tecnologie standard
  - Flessibilità dell'organizzazione dei suoi componenti
    - Le funzionalità vengono esposte agli utenti come servizi riusabili e condivisi in rete

Aims of Web Services

- Realizzare RPC per interfacce standard
  - Supportare l'interoperabilità tra diversi operatori residenti sulla rete
- Permettere l'utilizzo di servizi remoti
  - Attraverso una interfaccia software formalmente descritta
    - … e processabile da agenti automatici
- Comunicazione attraverso la rete internet
- Sostituzione delle tecnologie CORBA e RMI per l'invocazione di procedure remote
- Messaggi XML su HTTP → SOAP
- Paradigma orientato ai servizi: **SOA**

Web Service Architecture

- Architettura dei web service
  - Service Provider
  - Service Registry/Broker
  - Service Consumer/User

- **Operazioni principali dei web service**
  - Publishing – rendere un servizio disponibile
  - Finding/Discovering – localizzare un servizio
  - Binding – utilizzare un servizio

Gli standard alla base dei WS

- **UDDI:** Universal Description Discovery and integration, piattaforma per la descrizione di servizi, descrizione di business e integrazione di servizi. Proposto da IBM e Microsoft, www.uddi.org
- **WSDL:** Web Service Description Language, sviluppato da IBM, definisce un XML schema per descrivere metodi e parametri di un servizio WEB
- **SOAP** (1999), Simple Object Access Protocol, proposed by W3C (world Wide Web Consortium). SOAP utilizza XML sul protocollo HTTP per inviare richieste e ricevere delle risposte.
  - Nato da Microsoft e Userland come RPC
  - E' un protocollo RPC standardizzato da W3C
  - E' un protocollo XML based, basato su HTTP

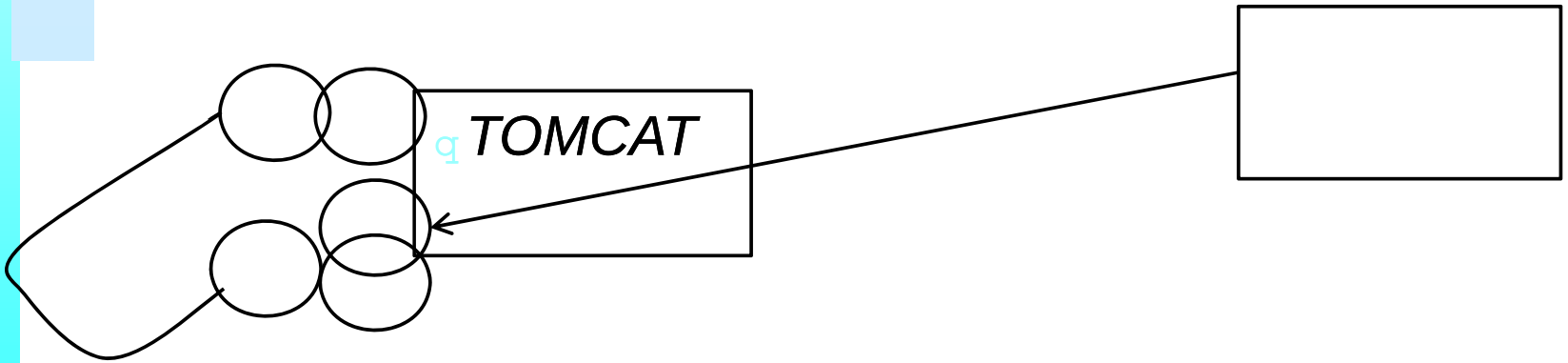
Web Services Protocols

Accessibility of WS

- To make it discoverable
  - Publishing a service in a registry (UDDI)
  - Publishing the service description of its interface (methods & arguments) in some manner
- To make it accessible by using a WSDL formalization of the Web Service.
  - IDL description
  - Platform independent formal description
  - Extensible language

Rendere un servizio accessibile

- Una volta che una descrizione del servizio e' stata pubblicata, ci deve essere un Web Server che metta a disposizione tale servizio
- Il WEB server puo' mettere a disposizione il servizio attraverso una applicazione custom, per esempio una Servlet, WAR (jsp based) deployed su un Web Application Server come Tomcat.
- Le funzionalita' dei WS vengono messe a disposizione
  - Le Web App possono essere:
    - Deployed, started, stopped, etc.

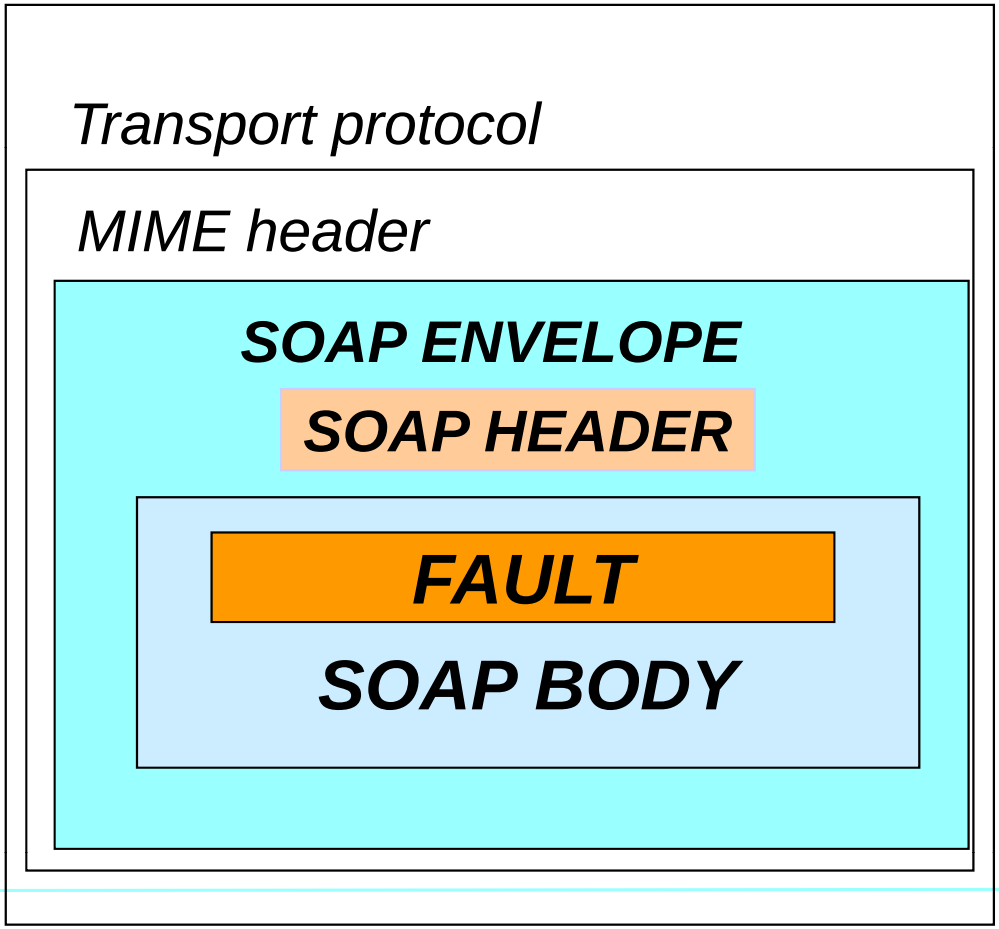


SOAP and WS

- Il corpo del messaggio è definito tramite WSDL
  - La grammatica del messaggio si basa su XML di W3C
- Il contenitore del messaggio è SOAP

- *The body element contains the core of the SOAP document – this will contain either the RPC call or the XML message itself*

- *The fault information may contain any exception information*



SOAP message

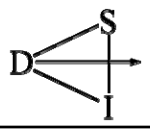

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP_ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3c.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3c.org/1999/XMLSchema">

        <SOAP-ENV:Header>

        </SOAP-ENV:Header>

        <SOAP_ENV:Body>


        </SOAP-ENV:Body>


</SOAP-ENV:Envelope>
```

SOAP as RPC on HTTP

- SOAP RPC message contains XML that represents a call or a response
- SOAP XML is converted into a call on the server
  - The response is encoded into SOAP XML to be returned to the client
- SOAP calls pass through firewalls since it is based on HTTP level protocol
- SOAP protocol supports different bindings of WSDL
- SOAP errors are handled using a specialised envelope known as a Fault Envelope
  - a SOAP Fault is a element which has to appear as an immediate child of the body element
  - <faultcode> and <faultstring> are required.

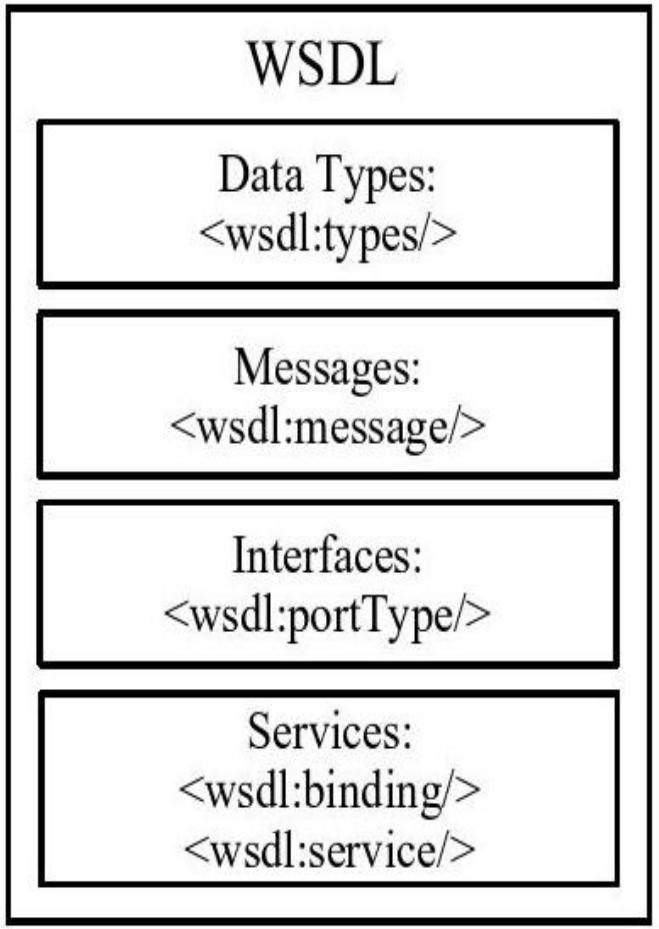
SOAP Fault

- Un fallimento SOAP e' un tipo speciale di messaggio mirato ad informare e descrivere un errore avvenuto in qualche fase del processo
- Le informazioni si possono dividere in 4 parti
- Fault code
  - Il valore del tipo di errore
  - 4 tipologie standard di errore
- Fault string
  - Descrizione dell'errore nel linguaggio corrente
- Fault actor
  - L'identidicatore del nodo sul quale si e' verificato l'errore
- Fault details
  - Dettagli relativi all'applicazione che ha riscontrato l'errore

WSDL:
Web Service Description Language

- An extended IDL
- Definition of standard interfaces
- Permitting advertising and publication of WS
- Permitting the definition of SLA, Service Level Agreements
- Publication of formal interface call on repositories, UDDI registry
- Requests on UDDI registry



Esempio di WSDL (1/4)

```
<?xml version="1.0"?>
<definitions name="StockQuote"
  targetNamespace= "http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd1="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
   <schema targetNamespace= http://example.com/stockquote.xsd
     xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="ticketSymbol"  type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
        </complexType>
      </element>
   </schema>
</types>
…….
```

Esempio di WSDL (2/4)

```
[…….]
<message name="GetLastTradePriceInput">
    <part name="body" element= "xsd1:TradePriceRequest"/>
  </message>
  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
  </message>
  <portType name="StockQuotePort">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
  </portType>
[…..]
```

Esempio di WSDL (3/4)

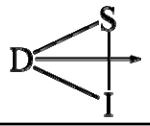

```
[…….]
<binding name="StockQuoteSoapBinding"
      type="tns:StockQuotePort">
      <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
      <soap:operation
        soapAction="http://example.com/GetLastTradePrice"/>
        <input>
          <soap:body use="literal"/>
        </input>
        <output>
          <soap:body use="literal"/>
        </output>
      </operation>
    </binding>
[…..]
```

Esempio di WSDL (4/4)

```
[…….]
<service name="StockQuoteService">
      <documentation>Il mio primo servizio web</documentation>
      <port name="StockQuotePort"
        binding="tns:StockQuoteBinding">
        <soap:address location="http://example.com/stockquote"/>
      </port>
    </service>
 </definitions>
[…..]
```

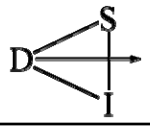
Client SOAP – JAX-WS (1/2)


```
public class MyFirstClient {
    @WebServiceRef(wsdlLocation="http://localhost:8080/
      StockQuoteService/stock?wsdl")

   static StockQuoteService service;

    public static void main(String[] args) {
        try {
            MyFirstClient client = new MyFirstClient();
            client.doTest(args);
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

public void doTest(String[] args) {
      [....]
    }
}
```

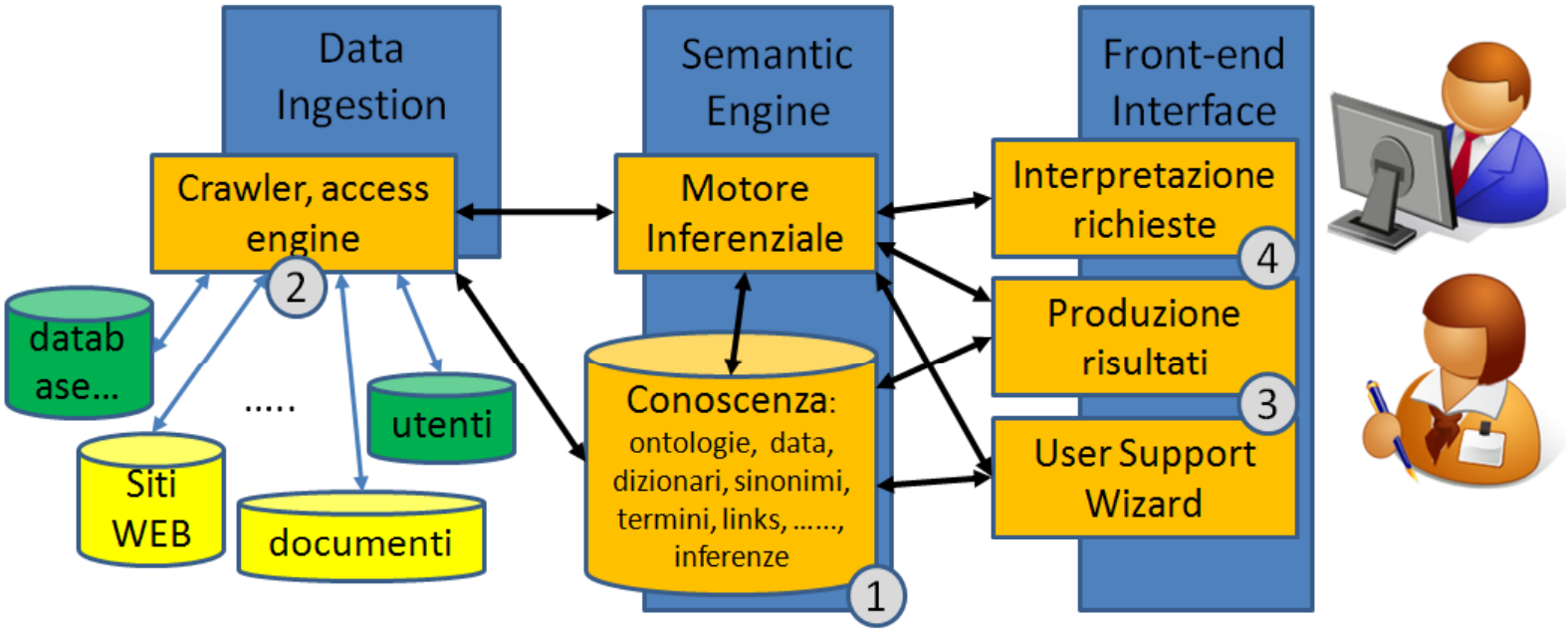




Client SOAP – JAX-WS (2/2)


```
public void doTest(String[] args) {
        try {
            System.out.println("Retrieving the port from
                    the following service: " + service);
            StockQuotePort port = service.getPort("StockQuotePort");
            System.out.println("Invoking the operation on the port");

            String tickerSymbol = "";
            if (args.length > 0) {
                tickerSymbol = args[0];
            } else {
                tickerSymbol = "No Symbol";
            }

            Float response = port.GetLastTradePrice(tickerSymbol);
            System.out.println(response);
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
```

Architetture basate su WS



- Crawling/ingestion dei dati di interesse attraverso servizi web dislocati sulla rete
- I servizi mettono a disposizione una interfaccia standard e permettono di reperire i dati di interesse

Generic SOAP Client (1/3)

Generic SOAP Client

This generic SOAP client allows you to access web services using a web browser. It performs dynamic bindings and executes methods at remote web services. Executing a SOAP service is a two-step process:

1. Enter the Web Service Description Language (WSDL) file, and click the retrieve button. Our SOAP Server will build HTML forms dynamically based on the description file.
2. Enter parameters in the HTML form and click the Execute button. This triggers the execution of the remote method. A SOAP client object will be created, which performs parameter binding, message construction/delivery, and finally response decoding. The result is then sent to your browser as a HTTP message.

WSDL File Address: siari-ws-unifi_preprod/ws/public/ConsultazioneCatalogo.wsdl

Default Array Size: 2 **Cache WSDL:** ☐ **No**

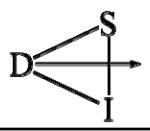
Retrieve

Tips for Users:

- If you know the WSDL file, you can setup a quick link to the client forms using
  http://www.soapclient.com/soapclient?template=/clientform.html&fn=soapform&SoapTemplate=none&SoapWSDL=Your_WSDL_File
  or
  http://www.soapclient.com/soapTest.html?SoapWSDL=Your_WSDL_File
- The server caches WSDL files in normal operations to improve performance. If you make any changes to a WSDL file, select the **No** checkbox.

Tips for Developers:

- Use <*documentation*> whenever possible in your WSDL file to provide instructions. It will be displayed in the client form.
- Use enumeration type if an element has fix number of values. They will be displayed as dropdown boxes.


Generic SOAP Client (2/3)

WSRIConsultazioneCatalogoService

Service Documentation : None

SOAP Method : getMetadataFormats

ServerAddress: https://test.unifi.u-gov.it:443/siari-ws-unifi_preprod/ws/public/ConsultazioneCatalogo

Show: Response Format: XML Invoke

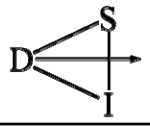
SOAP Method : getAuthorProductsByAB

ServerAddress: https://test.unifi.u-gov.it:443/siari-ws-unifi_preprod/ws/public/ConsultazioneCatalogo

getAuthorProductsByAB.authorID:	83324
getAuthorProductsByAB.metadataPrefix:	mods
getAuthorProductsByAB.filter.type:	Articolo su rivista
getAuthorProductsByAB.filter.from:	0
getAuthorProductsByAB.filter.to:	10
getAuthorProductsByAB.filter.orderingToken[0].field:	TYPE
getAuthorProductsByAB.filter.orderingToken[0].order:	ASC
getAuthorProductsByAB.filter.orderingToken[1].field:	YEAR
getAuthorProductsByAB.filter.orderingToken[1].order:	ASC

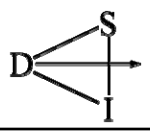
Show: Response Format: XML Invoke

SOAP Method : getAuthorProductsByURI

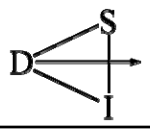
Generic SOAP Client (3/3)


```
- <S:Envelope>
  - <S:Body>
    - <ns2:getAuthorProductsByABResponse>
      - <return>
          <authorID>83324</authorID>
          <authorName>PAOLO</authorName>
          <authorSurname>NESI</authorSurname>
          <authorURI>urn:u-gov:unixx:AC_AB0:83324</authorURI>
          <pageProductsCount>2</pageProductsCount>
        - <products>
          - <metadataFormat>
              <name>mods</name>
              <namespace>http://www.loc.gov/mods/v3</namespace>
              <prefix>mods</prefix>
              <schema>http://www.loc.gov/standards/mods/v3/mods-3-3.xsd</schema>
            </metadataFormat>
            <productID>217953</productID>
          - <productMetadata>
            - <m:mods xsi:schemaLocation="http://www.loc.gov/mods/v3 http://www.loc.gov/standards/mods/v3/mods-3-3.xsd">
              - <m:titleInfo>
                  <m:title>Managing OO Projects Better</m:title>
                </m:titleInfo>
              - <m:name type="personal" xlink:href="urn:u-gov:unixx:AC_AB0:83324">
                  <m:namePart type="given">PAOLO</m:namePart>
                  <m:namePart type="family">NESI</m:namePart>
                - <m:role>
                    <m:roleTerm type="code">aut</m:roleTerm>
                  </m:role>
                - <m:role>
```


Generic SOAP Client

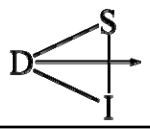
- Client Soap generico che permette di accedere ad un web service utilizzando un browser
- Abbiamo bisogno della descrizione del servizio sotto forma di file WSDL
- A seguito del processing del file WSDL il servizio identifica le operazioni messe a disposizione dal web service
- L'utente ha la possibilita' di accedere ai servizi offerti
  - Quello che serve e' solamente la descrizione del servizio web
  - … e alcuni dettagli delle operazioni
    - Valori dei parametri, valori di ritorno, parametri obbligatori,…
- Utilissimo per testing web service, di file WSDL, ….

Esempi di WS

- Tipicamente senza stato:
  - Dammi il tempo a Firenze Domani
  - Dammi I valori della borsa Milano riguardo a XX, YY
  - Dammi l''IP fisico e il GPS di http://WWW.pippo.it
  - Puoi consegnarmi questo tipo di lavoro <descrizione> entro XX
  - Dammi il mio profilo sono XX con password ZZZ

- Tipicamente con stato persistente ma non di protocollo:
  - Ho visto questo film
  - Ecco il mio profilo
  - Mi autorizzi a usare questo contenuto ?
  - Sono passato da questo punto
  - Ho rilevato questa temperatura alle coordinate GPS X,Y
  - Accetto la tua offerta <identificativo> per YY Euro

Esempi di WS non corretti

- Sono tipicamente sequenze, per esempio:
  - Call1: Sono Paolo Nesi
  - Call2: Dammi lo stato del mio conto
  - Call3: fai questa operazione

- Protocolli complessi portano a problemi di sicurezza
  - E' possibile proteggere la comunicazione tramite soluzioni come SSL, HTTPS, etc.
  - Lo stato implicito sul protocollo (come nell'esempio sopra) puo' portare a dover tenere conto di tempi di attesa per mantenere lo stato, etc.

REST

- Representational State Transfer (REST)
  - an approach for getting information content from a Web site by reading a designated (ok, you need a URI) Web page that contains an XML file that describes and includes the desired content.
  - To use HTTP (get,put,…) to a URI, with XML as the payload …
- REST is an architecture strategy and not a protocol
- REST+XML is appropriate for all of the applications that SOAP/WSDL are designed/used for.
- Suitable for exploiting/encapsulating legacy

REST design

- Components in a REST system obey the following constraints:
  - Servers are stateless and messages can be interpreted without examining history.
  - resources are identified through URIs
  - resources manipulated through representations
  - Uniform Interface for all resources:
    - GET (Query the state, idempotent, can be cached)
    - POST (Modify, transfer the state)
    - PUT (Create a resource)
    - DELETE (Delete a resource)
  - self-descriptive messages

REST design

- REST emphasizes the role of intermediaries: caches, proxies, gateways, etc.
  - The solution has to work in all these conditions.

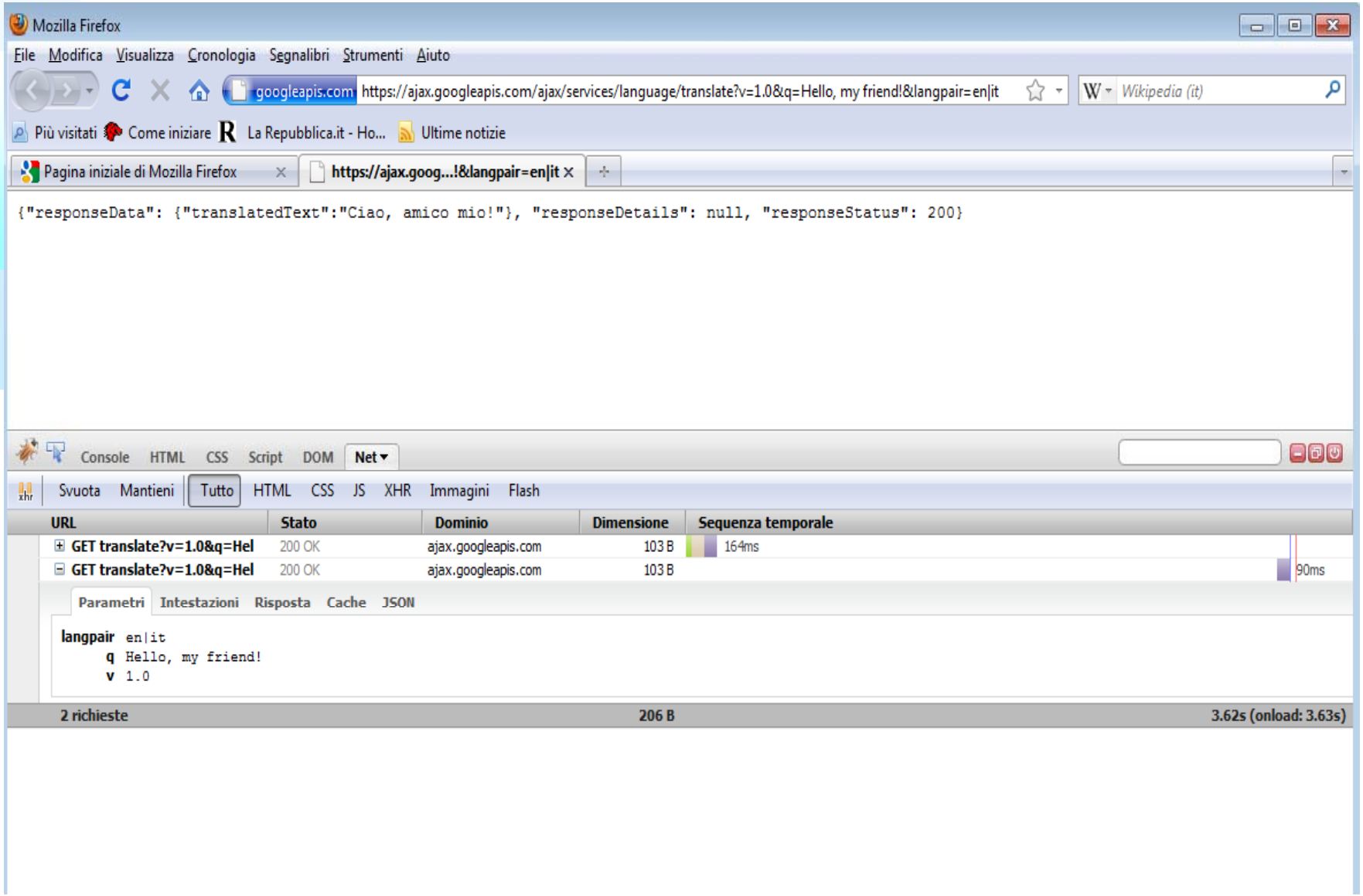
- Main correspondences

	SQL	REST
CREATE	Insert	PUT
READ	Select	GET
UPDATE	Update	POST
DELETE	Delete/Drop	DELETE

Google Translate

- Servizio web di traduzione automatica di Google
- Utilizza un approccio statistico per tradurre intere frasi o documenti in lingue diverse
  - Mette a disposizione 25 tipologie di lingue diverse
- Richiesta HTTP:
  - https://ajax.googleapis.com/ajax/services/language/translate?v=1.0&q=Hello, my friend!&langpair=en|it
- Risposta in formato JSON
  - {"responseData": {"translatedText":"Ciao, amico mio!"}, "responseDetails": null, "responseStatus": 200}

Esempio di Google Translate



Confronto: REST vs WS

- REST:
  - Maggiore dipendenza fra server e client
  - Minor costo di realizzazione per cose semplici
  - Minor espressività del modello di chiamata
  - Problemi con dati strutturati
- WS:
  - Maggior costo di realizzazione del server e del client
  - Maggiore formalizzazione
  - Pubblicazione formale dell'interfaccia, verifica formale di consistenza, gestione degli errori, vedi soap fault
  - Minor dipendenza fra server e client
  - Si possono sviluppare chiamate complesse con dati anche molto strutturati

References

- SOAP http://www.w3c.org/TR/soap
- WSDL http://www.w3c.org/TR/wsdl
- UDDI http://www.uddi.org
- REST Resources:

http://www.prescod.net/rest

http://internet.conveyor.com/RESTwiki

Realizzazione di una applicazione C++

- Nelle prossime due slide è mostrato come è possibile realizzare un'applicazione C++ composta da server e client che comunicano tramite WS via SOAP
- Questa soluzione è piu' statica di quando descritto in precedenza, cioe' prevede la presenza di un WSDL per produrre il Client compilato

- Si parte dal Server
  - Si decide di pubblicare una certa funzione F(int a, string b)

- Per arrivare ad avere tale funzione come RPC sul client

GSOAP server side

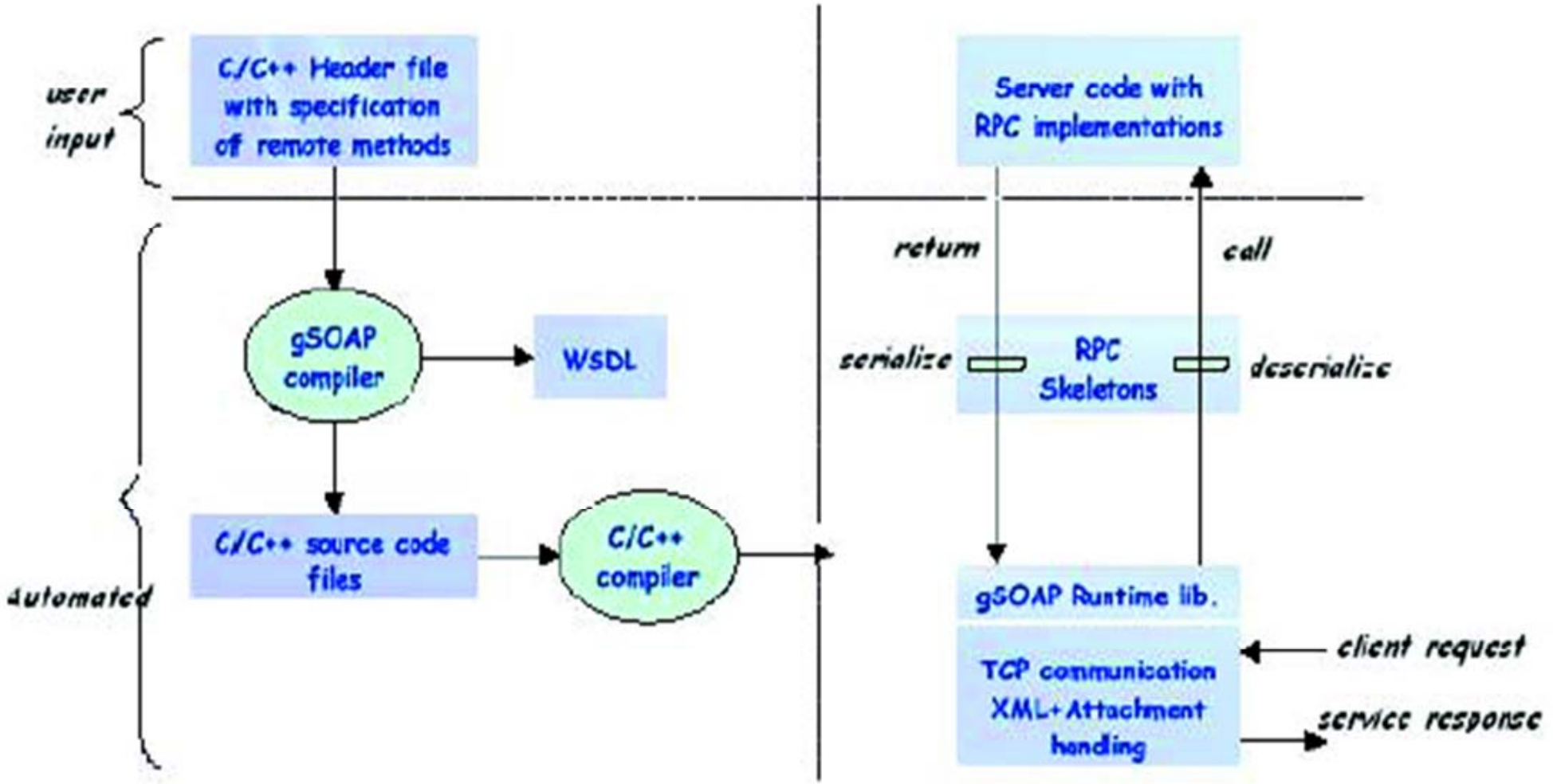


Fig.3- gSoap server-side

GSOAP client side

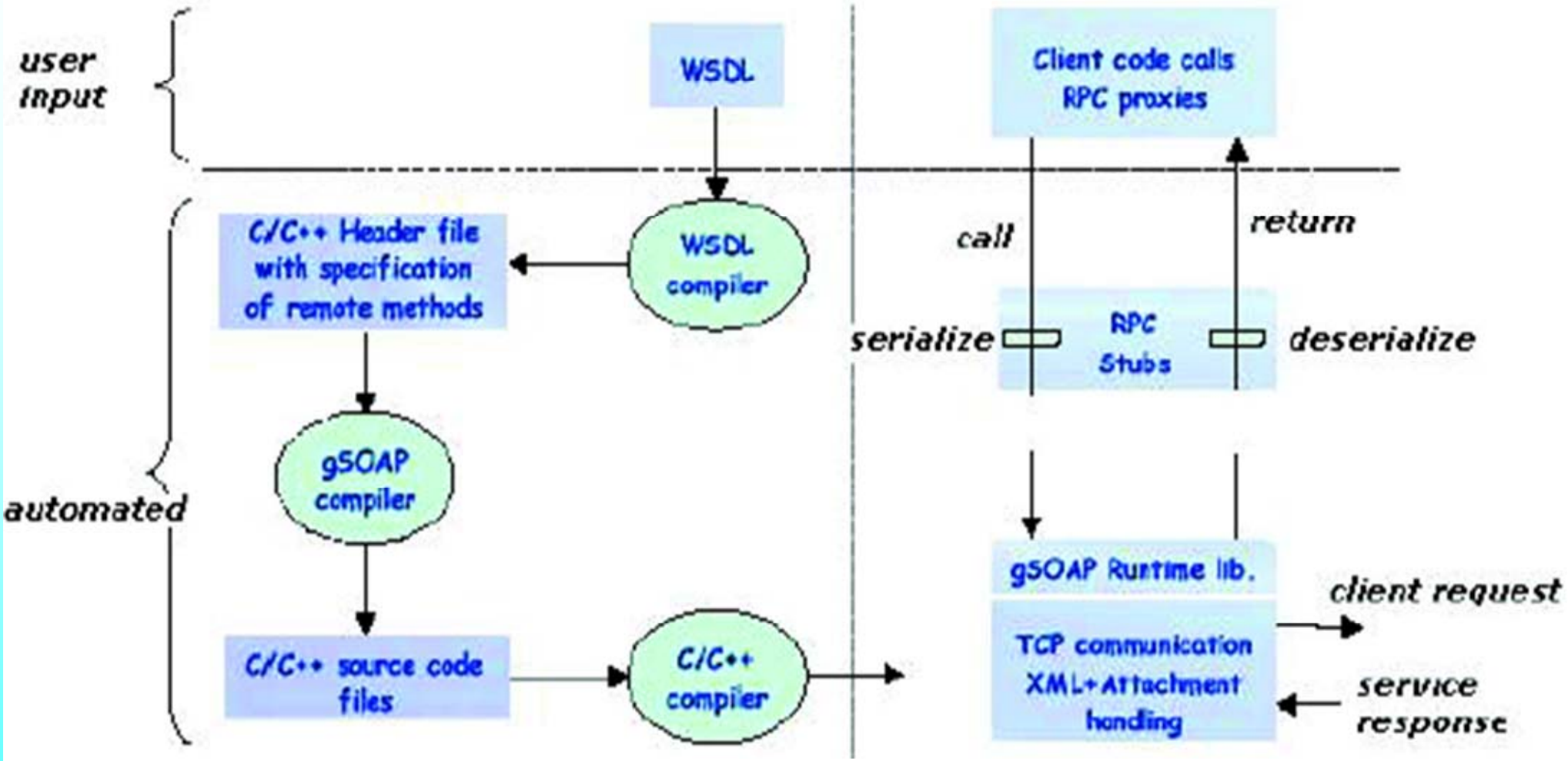


Fig.4- gSoap client-side

Critical Aspects

- Lack of defined methodologies (distributed architecture, heterogeneous languages and models)
- problems in
  - tracking requirements
  - maintainability, design for maintenance
  - portability, technology evolution
  - modularization, composability, design for reuse
  - Informal specification
  - requirement and context evolutions
  - short term development process
  - reliability, tracking defects
  - security problems
  - distributed responsibility among partners (COTS and partnerships)