

Arhitecturi orientate pe servicii

Web Services

Arhitecturi Orientate pe Servicii

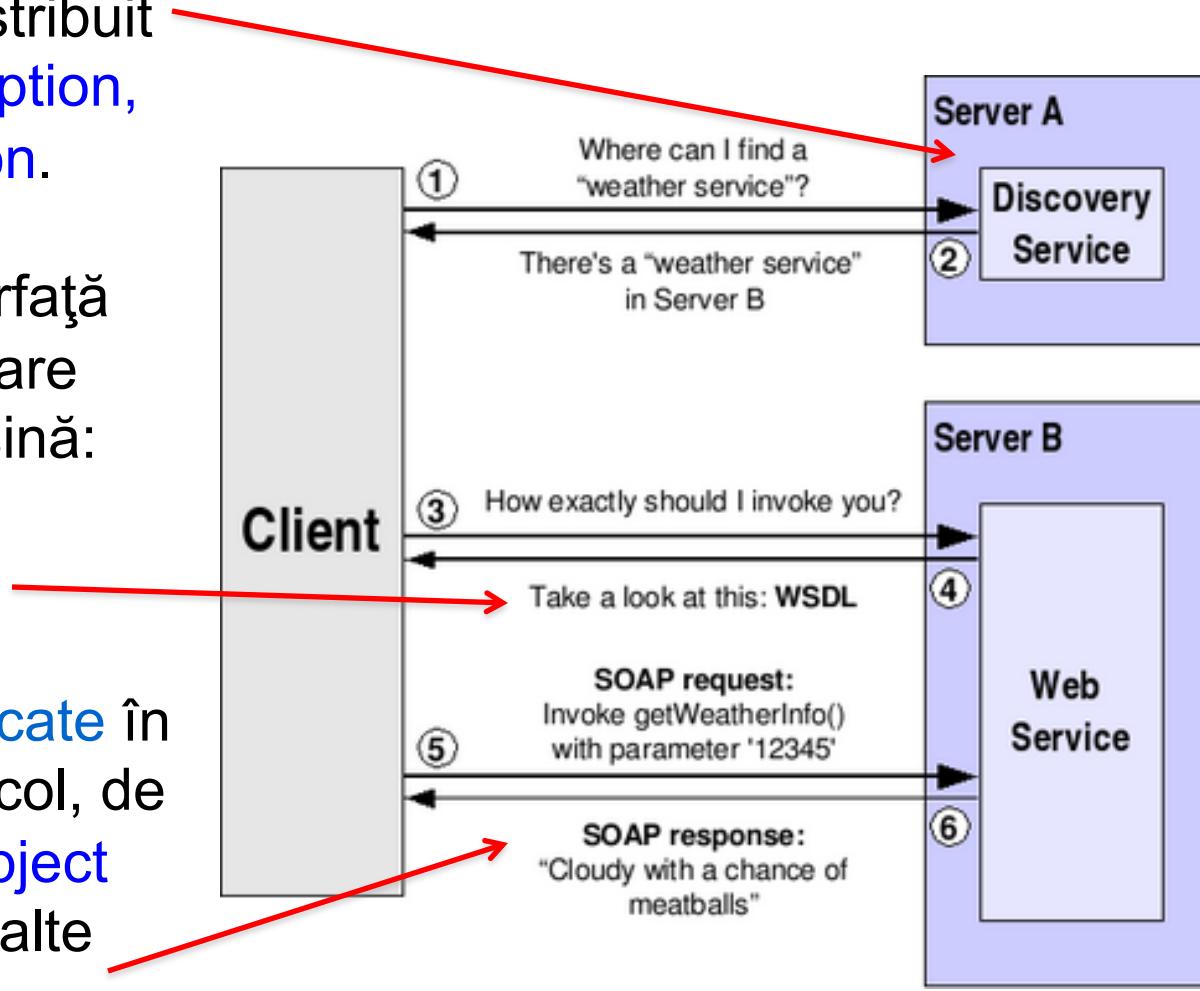
- Servicii Web
- Reprezentarea mesajelor, SOAP
- Descrierea serviciilor, WSDL
- Publicarea si cautarea serviciilor, UDDI
- Utilizarea serviciilor, WSDL - JAX-RPC

Elemente ale arhitecturii serviciilor Web

Descoperirea serviciilor Web se face printr-un director distribuit UDDI – Universal Description, Discovery, and Integration.

Serviciul WEB are o interfață descrisă într-un format care poate fi procesat de mașină: WSDL - Web Service Description Language.

Serviciile WEB sunt invocate în conformitate cu un protocol, de obicei SOAP - Simple Object Access Protocol (posibil alte protocoale).



Arhitectura orientata pe servicii

Descoperire (UDDI)

Descriere (WSDL)

Mesagerie XML (XML-RPC, SOAP, XML)

Transport (HTTP, SMTP, FTP, BEEP)

Stiva de protocoale include functionalitati pentru

- transportul mesajelor intre aplicatii
 - BEEP (Blocks Extensible Exchange Protocol) – ofera securitate, tratarea erorilor, dialogul initial intre entitati (handshake)
- codificarea mesajelor intr-un format comun XML
- descrierea interfetei publice a unui serviciu
- evidenta serviciilor si mecanismul de publicare/subscriere

SOAP – Simple Object Access Protocol

- SOAP defineste o **schema** de utilizare a XML pentru
 - **mesaje asincrone**
 - ex. procesarea unei cumparaturi – dureaza mai mult si implica un schimb mai bogat de mesaje, inclusiv mesaje periodice de stare, detalii la finalul operatiilor etc.
 - **cerere-raspuns – RPC**
 - ex. verificarea online a unui card bancar
- Specifica
 - regulile de reprezentare a diferitelor **tipuri de date** din mesaje
 - alcatuirea **mesajelor** respectiv a **cererilor / raspunsurilor**
 - cum se folosesc HTTP, SMTP ... pentru **transportul** mesajelor
- API-uri SOAP au fost implementate in diverse limbaje: Java, Perl, JavaScript, Python, C++, C#, Visual Basic

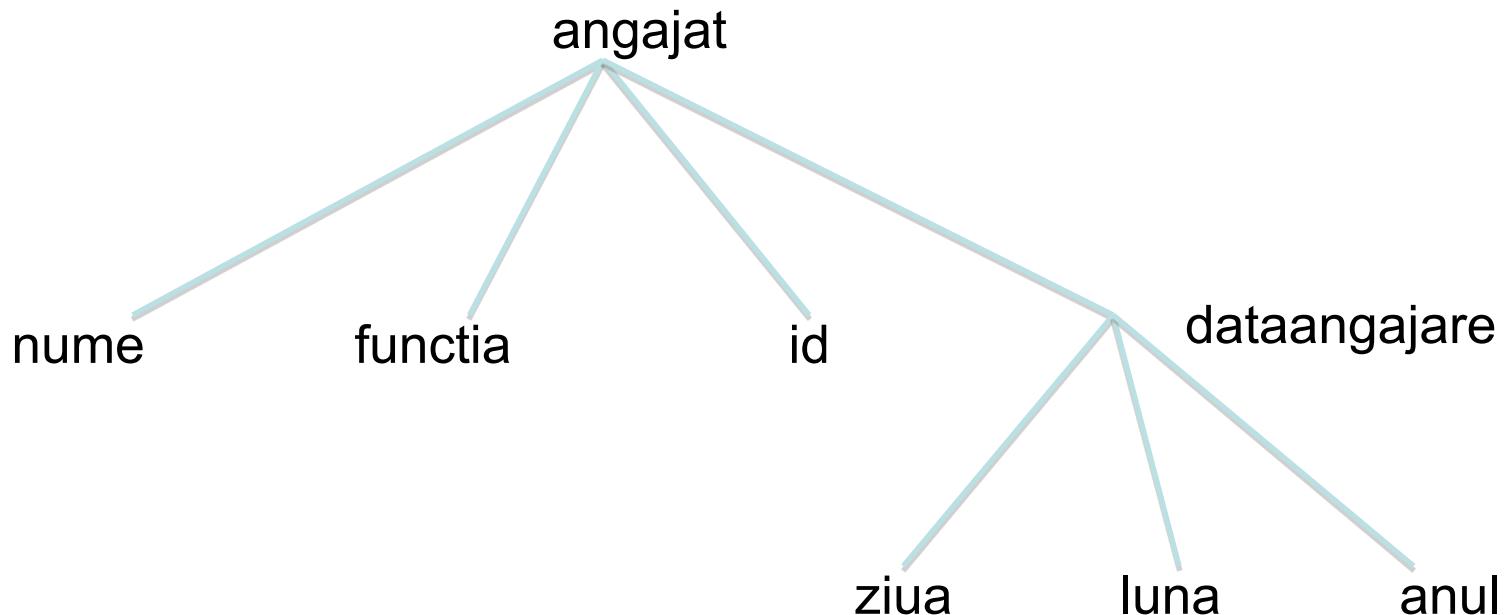
Cateva notiuni XML folosite in SOAP

Datele, reprezentate ca **text**, sunt intercalate cu **marcaje** (de asemenea textuale); marcajele permit aflarea **tipului si structurii** datelor

```
<?xml version="1.0"?>                                prolog – versiunea XML  
<angajat>                                         element radacina (unic)  
  <nume>Nicolae Ionescu</nume>  
  <functia>contabil</functia>  
  <id>123456789</id>  ← element inclus  
  <dataangajare>                                         nume = id  
    <ziua>5</ziua>                                         valoare 123456789  
    <luna>iunie</luna>  
    <anul>2013</anul>  
  </dataangajare>  
</angajat>
```

Modelul de date

- Descrierea unui **angajat** este **structurata** conform unui tipar (un arbore) si include mai multe elemente: **nume**, **functia**, un **identificator unic** si **data angajarii**
- **dataangajare** include trei elemente: **ziua**, **luna**, **anul**



Document XSD

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/
          XMLSchema">
  <element name="angajat">
    <complexType>
      <sequence>
        <element name="nume" type="string"/>
        <element name="functia" type="string"/>
        <element name="id" type="string"/>
        <element name="dataangajare">
          <complexType>
            <sequence>
              <element name="ziua" type="integer"/>
              <element name="luna" type="string"/>
              <element name="anul" type="integer"/>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```

Schema de grupare a **elementelor** si **tipurile** lor sunt definite de un document **XSD (XML Schema Document)**

Defineste

- **numele** componentelor
 - **tipul** lor
 - ordinea elementelor,
 - numarul minim si maxim de aparitii etc.
- care permit “intelegerea” unui document XML de un program si de un operator uman

Spatiul de nume (XML namespace)

`xmlns="http://www.w3.org/2001/XMLSchema"`

In plus, nume de **tipuri** si **attribute** folosite pentru descrierea elementelor dintr-un document sunt definite in **spatii de nume XML**

O colectie de **nume** de **tipuri** si **attribute** (XML namespace) este **referita** printr-un URI (al fisierului care contine colectia)

Un **namespace** este specificat intr-un document XML printr-o pereche **atribut=valoare**

un document XML poate folosi mai multe spatii de nume

un acelasi nume poate apare in spatii de nume diferite
pentru evitarea ambiguitatilor se folosesc **nume calificate**

Două spații de nume cu marcajul comun “title”

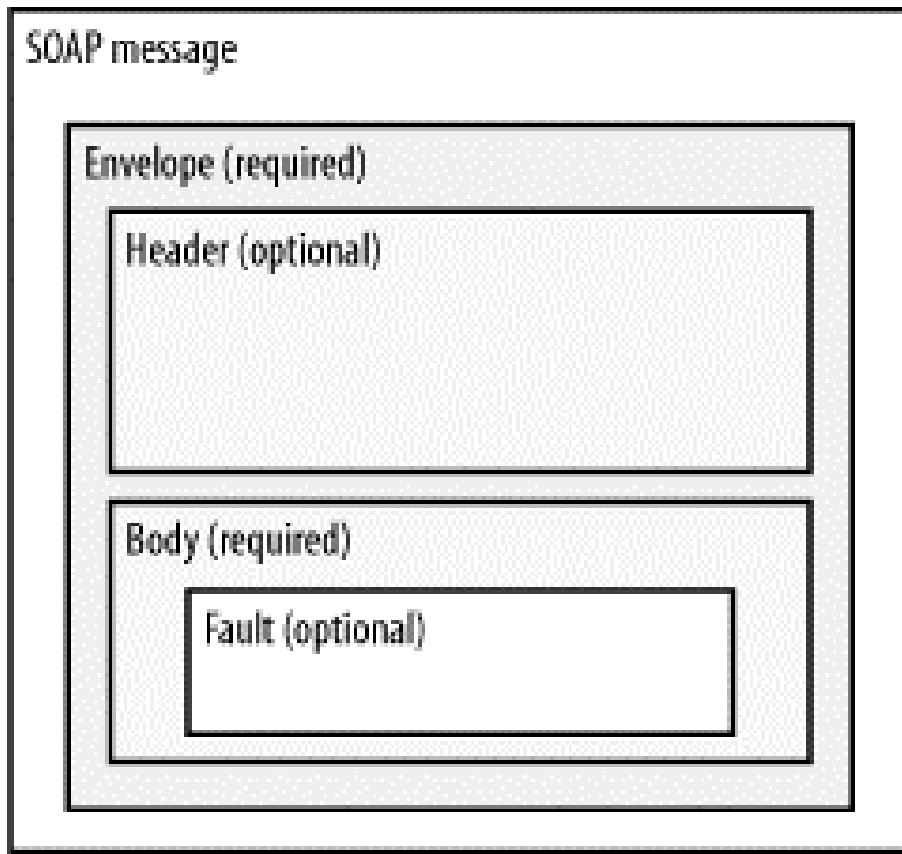
```
<?xml version="1.0"?>
<library-entry xmlns:authr="http://www.w3.org/TR/html4/"
               xmlns:bk="http://www.w3schools.com/furniture">

  <book>
    <bk:title> Mythical Man-Month: Essays on Engineering </bk:title>
    <pages>336</pages>
    <isbn>0-201-83595-9</isbn>
    <author>
      <firstname>Frederick</firstname>
      <middleinitial>P</middleinitial>
      <lastname>Brooks</lastname>
      <suffix>Jr</suffix>
      <authr:title>Dr</ authr:title>
    </author>
  </book>
</library-entry>
```

numele calificate **bk:title** si **authr:title** au ca prefix identificatorii spatiilor de nume

Elementele principale ale unui mesaj XML SOAP

- Doua componente obligatorii: o *anvelopă* și *corpu* mesajului
 - envelopă este singurul element al cererii; include
 - header – contextul pentru serviciu
 - body – documentul continand mesajul



O cerere SOAP

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <ns1:getTemp
            xmlns:ns1="urn:xmethods-Temperature"
            SOAP-
            ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <zipcode xsi:type="xsd:string">10016</zipcode>
        </ns1:getTemp>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Cererea are un singur element obligatoriu **Envelope**
care include elementul obligatoriu **Body**

Anvelopa

XML namespaces

`xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"`
mentioneaza versiunea folosita SOAP 1.1
diferita de 1.2: <http://www.w3.org/2001/09/soap-envelope>

`xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`
`xmlns:xsd="http://www.w3.org/2001/XMLSchema">`

specifica scheme de codificare a datelor si le leaga la
prefixele xsi , xsd

`xmlns:ns1="urn:xmethods-Temperature"`

dezambiguizare identificatori din aplicatie specifici serviciilor
disponibile public XMethods si leaga la prefix ns1

Corpuș

Element Body

- ✧ Folosit de regula pentru cerere si raspuns RPC
- ✧ Cererea

```
<ns1:getTemp  
xmlns:ns1="urn:xmethods-Temperature"  
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">  
    <zipcode xsi:type="xsd:string">10016</zipcode>  
</ns1:getTemp>
```

- apeleaza metoda `getTemp`
- cu parametrul `zipcode`
 - avand tipul `string`
 - si valoarea `10016`

Raspunsul SOAP

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <ns1:getTempResponse
            xmlns:ns1="urn:xmethods-Temperature"
            SOAP-
            ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
            <return xsi:type="xsd:float">71.0</return>
        </ns1:getTempResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Elementul **getTempResponse** intoarce o valoare de tipul **float**. Valoarea (temperatura) este **71.0** grade Fahrenheit.

Raspuns la eroare

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode xsi:type="xsd:string">
                SOAP-ENV:Client</faultcode>
            <faultstring xsi:type="xsd:string">
                Failed to locate method (ValidateCreditCard) in class
                (examplesCreditCard) at /usr/local/ActivePerl-5.6/lib/
                site_perl/5.6.0/SOAP/Lite.pm line 1555.
            </faultstring>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Raspunsul are un singur element, **Fault**

cu doua sub-elemente care specifică:
faultcode – clasa de erori **Client** – cererea
contine o eroare (ex. nume metoda inexistentă)
faultstring – descrierea erorii este în limbaj
natural

Tipuri de date

SOAP are un set de reguli pentru codificarea datelor

Pentru specificarea tipurilor de date, SOAP foloseste

- specificatiile elaborate de W3C (XML Schema)
- propriile conventii SOAP (ex. pentru tablouri si referinte)

Stilul de codificare este setat prin atributul SOAP-ENV:[encodingStyle](#)

SOAP 1.1 - <http://schemas.xmlsoap.org/soap/encoding/>

SOAP 1.2 - <http://www.w3.org/2001/09/soap-encoding>

Tipuri scalare

- includ string, float, double, integer
- sunt tipuri definite de XML Schema
- in ex. **atributul** xsi:type este setat la **tipul** xsd:double
`<return xsi:type="xsd:double">54.99</return>`

Tipuri compuse

Tablou

- Tipul **xsi:type** pentru tablou este un **Array**
- elementul **arrayType** specifica **tipul elementelor si dimensiunile**
 - ex. tablou de 10 elemente double
`arrayType="xsd:double[10]"`
 - ex. tablou bi-dimensional de string-uri
`arrayType="xsd:string[5,5]"`

Structuri

Pentru fiecare element este specificat un **nume** de acces si un **tip**

SOAP peste HTTP

- SOAP nu este legat de un protocol specific dar **HTTP** este cel mai popular
- Cererile sunt transmise prin **HTTP POST** (modificare resursa)

POST /perl/soaplite.cgi HTTP/1.0

Host: services.xmethods.com

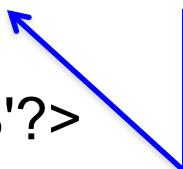
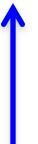
Content-Type: text/xml; charset=utf-8

Content-Length: 538

SOAPAction: "urn:xmethodsBabelFish#BabelFish"

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope ...
```

continutul este comanda sau
raspunsul SOAP transportate



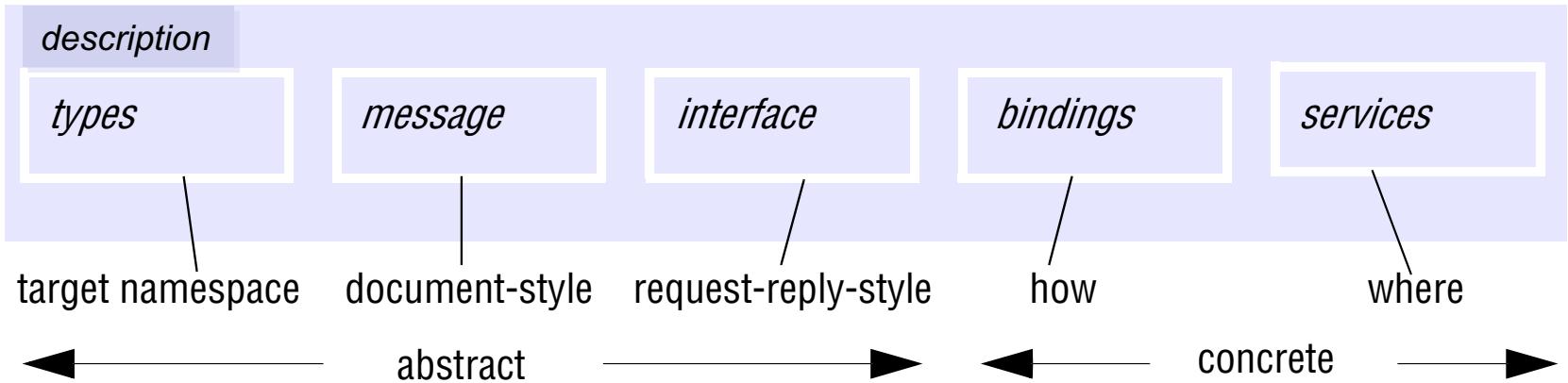
valoarea specificata aici
depinde de implementarea
serverului;
in ex. urn necesar pentru
accesul la serviciul
Translation de la **AltaVista**
BabelFish

Servicii Web - Definirea interfetelor

- Programarea cu **interfete** separa interfata de implementare si ascunde heterogeneitatea
- O **interfata** de serviciu Web consta intr-o **colectie de operatii** ce pot fi executate de programe, obiecte, baze de date ...
- Un document WSDL descrie un serviciu web precizand
 - **operatiile** oferite
 - **tipurile** datelor de intrare si rezultatelor fiecarei operatii
 - **protocolele** folosite pentru comunicarea client-server
 - **adresele** la care serviciul poate fi accesat
- Se folosesc mai multe **paradigme de comunicare** – serviciile pot fi descrise in termenii
 - **mesajelor** schimbatе intre client si server – stilul **document**
 - sau al **operatiilor** suportate de server – stilul **request-reply**

Elemente WSDL

The main elements in a WSDL description



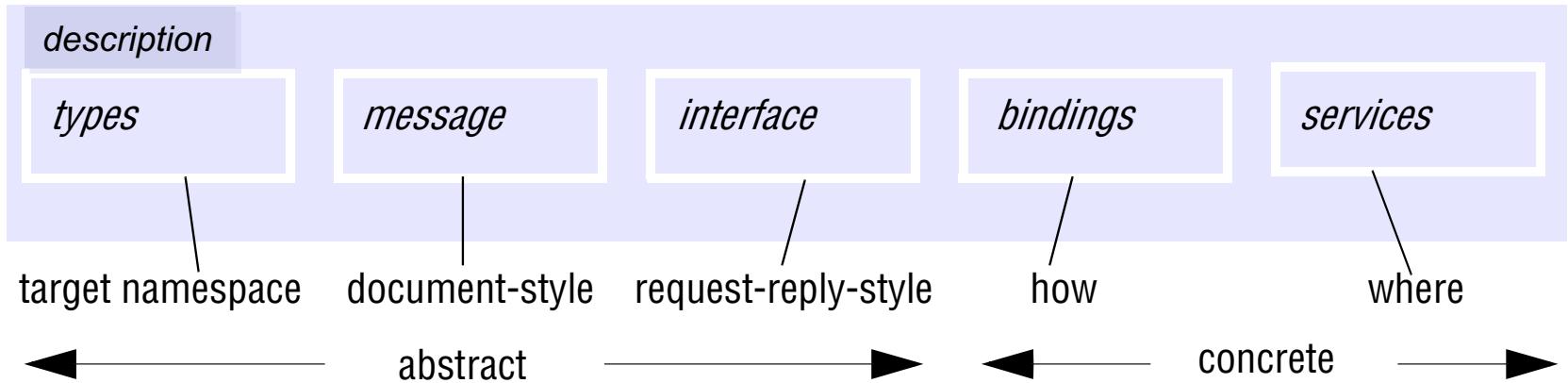
Descrierea WSDL este **exhaustiva**

- Include **toate elementele serviciului** care sunt **relevante pentru client**
- sunt folosite pentru a **genera codul proxy** la client

Descrierea are un **element radacina (description)** si elemente **incluse**, grupate in doua parti

Elemente WSDL – partea abstractă

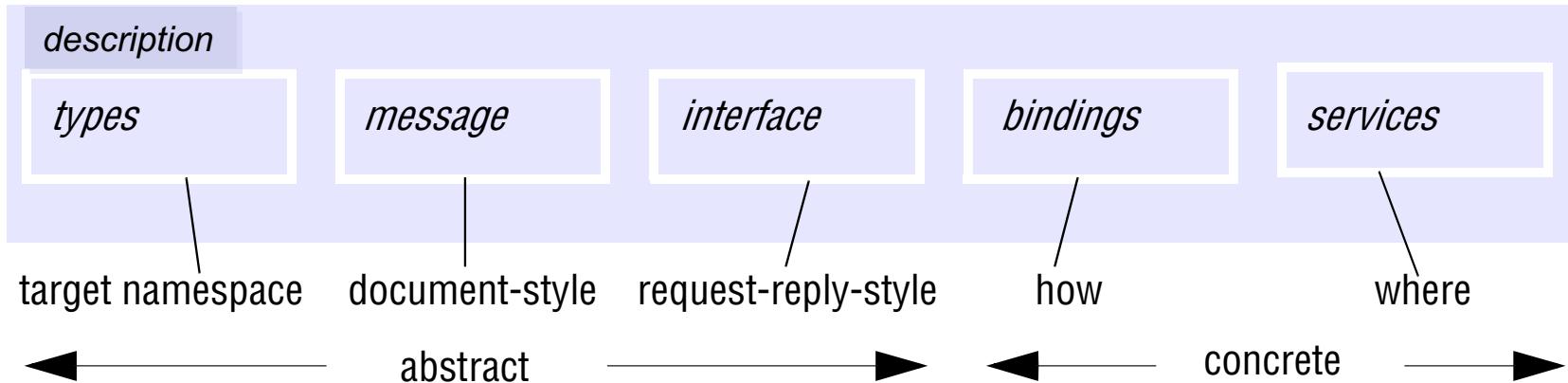
The main elements in a WSDL description



- **types** – definește tipurile folosite de serviciu
 - setul de nume definit în secțiunea **types** formează un spațiu de nume distinct, *target namespace*
- **message** - descrie mesajele schimbate specificând numele mesajelor, parametrii sau valorile returnate
- **interface** - **operatiile** serviciului Web

Elemente WSDL – bindings

The main elements in a WSDL description

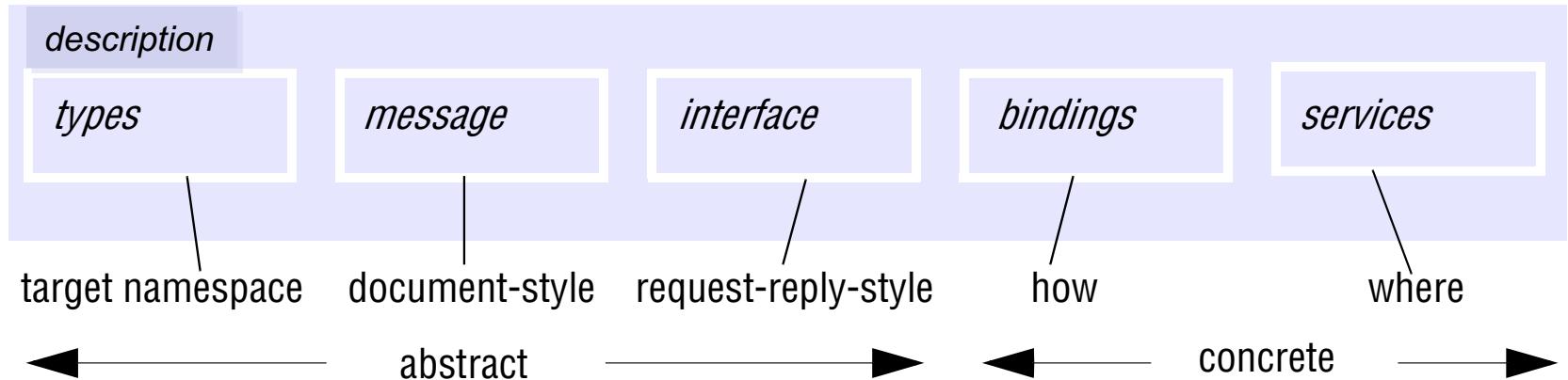


Specifica

- formatul mesajelor,
- protocolul de transport,
- tiparul schimbului de mesaje
 - in-out = cerere-raspuns
 - sau schimb de documente

Elemente WSDL – services

The main elements in a WSDL description



Specifica

- numele serviciului
- portul (endpoint) serviciului – care se referă la
 - binding-ul folosit
 - URL-ul serverului

Exemplu: serviciu Hello

La inceput sunt specificate spatiile de nume folosite in descrierea serviciului

```
<?xml version="1.0"?>
```

```
<wsdl:description
```

```
    xmlns:wsdl="http://www.w3.org/ns/wsdl"
```

- namespace pentru toate numele de elemente folosite in limbajul de baza WSDL

```
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
```

- nume si atribute din extensia SOAP

```
    xmlns:hy="http://www.herongyang.com/Service/"
```

- introduce prefixul hy: care permite referirea spatiului de nume targetNamespace

```
targetNamespace="http://www.herongyang.com/Service/">>
```

- namespace pentru toate numele definite in acest document

Definirea tipurilor

<types>

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
targetNamespace="http://www.herongyang.com/Service/">
```

- ❑ foloseste XML Schema in definirea tipurilor

```
<xsd:element name="Hello" type="xsd:string"/>
```

```
<xsd:element name="HelloResponse" type="xsd:string"/>
```

- ❑ definesc tipurile elementelor Hello si HelloResponse, ce vor fi incluse in targetNamespace

```
</xsd:schema>
```

</types>

- **Hello si HelloResponse** sunt parametri de intrare / iesire ai operatiei Hello definite in sectiunea **interface** – slide urmator!)

- tipurile trebuie definite ca elemente singulare (nu ca secventa de elemente)

- tipul poate fi complex (elementul poate include sub-elemente)

Definirea interfetei

```
<wsdl:interface name="helloInterface" >
    <wsdl:operation name="Hello"
        pattern="http://www.w3.org/ns/wsdl/in-out"
             defineste operatia, "Hello" cu tiparul in-out
        style="http://www.w3.org/ns/wsdl/style/iri">
             mesajele pot fi serialize, conform stil IRI - Internationalized
                Resource Identifier
    <wsdl:input messageLabel="In"
        element="hy>Hello" />
    <wsdl:output messageLabel="Out"
        element="hy>HelloResponse" />
             operatia are un parametru Hello si un raspuns HelloResponse
                ale caror tipuri se gasesc in spatiul de nume hy (slide precedent)
</wsdl:operation>
</wsdl:interface>
```

Binding

```
<wsdl:binding name="helloBinding"
```

```
    interface="hy:helloInterface"
```

❑ *binding-ul helloBinding se referă la interfața helloInterface*

```
    type="http://www.w3.org/ns/wsdl/soap"
```

❑ *formatul mesajului corespunde regulilor SOAP 1.2*

```
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"/>
```

❑ *protocolul de transport folosit este SOAP 1.2 peste HTTP*

```
<wsdl:operation ref="hy>Hello"
```

```
    wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
```

❑ *în operația Hello, tipul schimbului de mesaje (MEP – Message Exchange Pattern) este in-out*

```
</wsdl:binding>
```

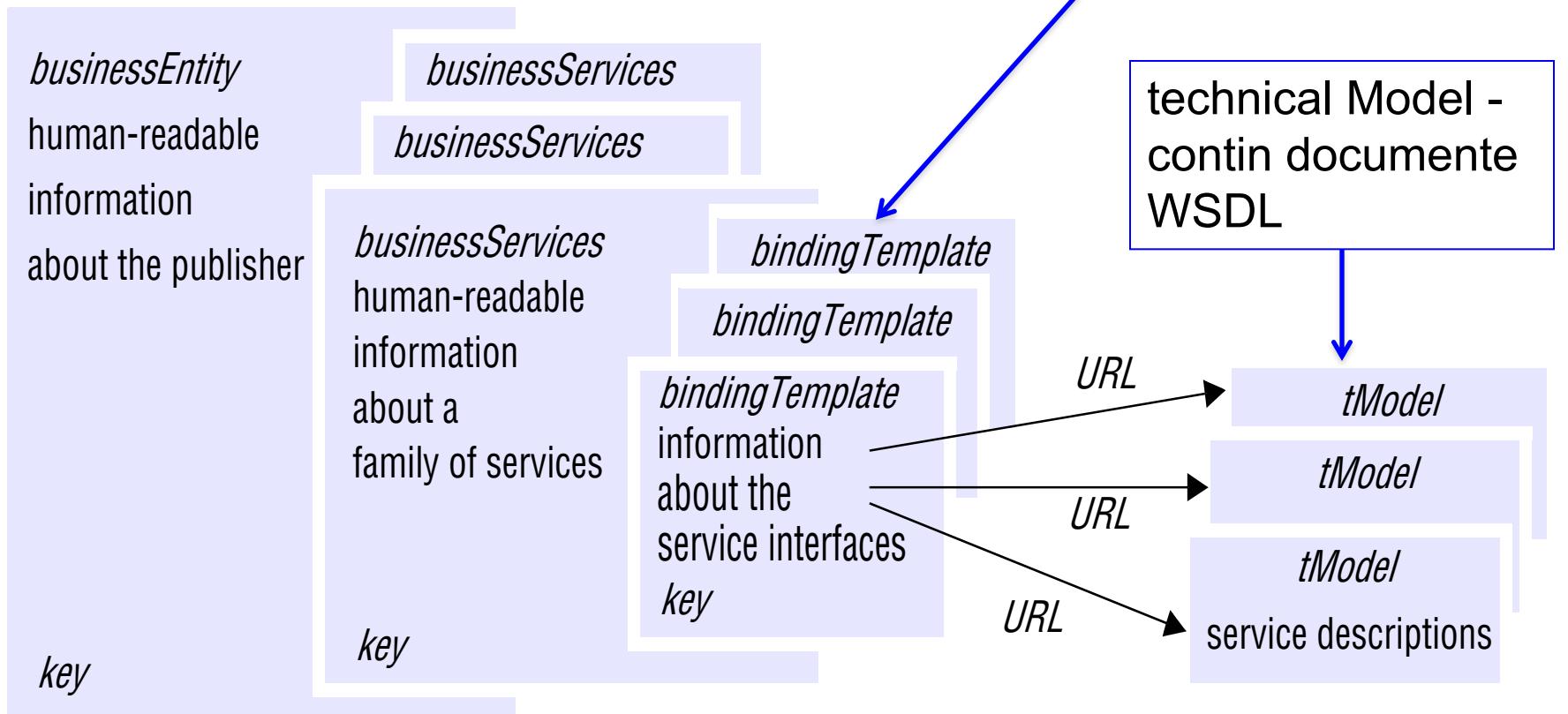
Descrierea serviciului

```
<wsdl:service name="helloService"
  interface="hy:helloInterface">
  <wsdl:endpoint name="helloEndpoint"
    binding="hy:helloBinding"
    address="http://hello.example.com/2004/Service/HelloService"/>
</wsdl:service>
</wsdl:description>
```

Defineste numele serviciului **helloService**
cu interfata **helloInterface**
cu endpoint avand numele **helloEndpoint**
avand binding-ul curent **helloBinding**
si adresa **http://hello.example.com/2013/Service/HelloService**

Universal Description, Discovery and Integration service (UDDI)

- Ofera servicii de nume si director
 - cautare dupa **numele** serviciului
 - cautare dupa **attribute**: categoria serviciilor
- Datele sunt organizate ca **patru structuri**



Cautarea

- UDDI are un **API** pentru servicii de cautare
- doua seturi de **query**
 - gasirea **unei entitati** ce corespunde unei **chei** (`get_xxx`)
`get_BusinessDetail`, `get_ServiceDetail`, `get_bindingDetail`,
`get_tModelDetail`
 - gasirea unui **set de entitati** ce corespund unor **criterii**
`find_business`, `find_service`, `find_binding`, `find_tModel`
- folosite in invocari successive care ingusteaza aria de cautare
 - `find_business` – afla lista cu furnizorii
 - `find_service` – gaseste servicii care se potrivesc atributelor
 - se alege **cheia** pentru un *bindingTemplate*
 - se regaseste URL-ul documentul WSDL corespunzator
- UDDI ofera o **interfata subscribe/notify** care furnizeaza abonatilor informatii despre entitatile de interes

Publicarea

- UDDI are o interfata pentru publicarea si actualizarea serviciilor Web
- La **prima publicare** a unei structuri de date, i se asociaza o cheie in forma unui URI
 - ex. uddi:abcd1.net:123
- serverul UDDI devine **proprietarul** acesteia
- el **difuzeaza** informatia altor servere
- **raspunsul** la o cautare in UDDI este dat de **un singur server**, fara a interactiona cu alte servere din acelasi **registru**

Registre

- Un **registru** UDDI consta din unul sau mai multe servere, fiecare avand o copie a aceluiasi set de date
- **Modificarile** unei structuri de date se fac la **serverul proprietar**, cel la care aceasta a fost publicata prima data
 - proprietarul poate ceda acest drept unui alt server
- Schema de replicare
 - **serverul** care face modificarea **notifica** celelalte servere
 - acestea cer modificarile pentru actualizare
 - **propagarea** modificarilor foloseste **vectori de timp**

Model simplu – folosire usoara

- toate modificarile unei structuri de date se fac la **acelasi server** care **propaga** apoi actualizarile
- ca urmare, actualizarile sunt primite de un server in **ordinea secventiala** corecta
- nu se impune vreo negociere intre actualizarile facute de servere diferite

Utilizarea WSDL

- documentele WSDL pot fi **accesate** prin **URI**-uri de clienti si servere
- **generarea** definitiilor WSDL se poate face prin **instrumente** cu interfata grafica
 - ocolesc structurile complexe ale WSDL
 - usureaza efortul utilizatorilor
- ex. Web Services Description Language for Java Toolkit (**WSDL4J**) permite crearea, reprezentarea si manipularea documentelor WSDL
- definitiile WSDL pot fi generate in API-uri pentru diverse limbaje
 - ex. Java in **JAX-RPC** (**Java API for XML based RPC**)

JAX-RPC

API Java care permite crearea de servicii Web si accesul lor folosind standarde XML

- implementarea de **referinta** foloseste **SOAP** (ca protocol de aplicatie) si **HTTP** (ca protocol de transport)
- **cererile** si **raspunsurile** sunt transmise ca **mesaje XML**
- Dezvoltatorul poate ignora detalii XML despre serviciu (totusi, cunoasterea lor este **recomandata** pentru facilitatile avansate ale JAX-RPC – de ex. folosirea unor tipuri de date in afara celor suportate transparent de JAX-RPC)

Servicii si clienti

Dezvoltarea unui serviciu web in JAX-RPC incepe cu

- crearea unei **interfete Java** cu o metoda pentru fiecare operatie
- crearea unei **clase Java** care implementeaza interfata

Pentru **invocarea serviciului**

- JAX-RPC permite **conversia automata** de la descrierea **Java** a interfetei serviciului la descrierea **WSDL**
- din documentul **WSDL**, JAX-RPC poate genera **codul Java** pentru legarea clientului la server si apelul serviciului

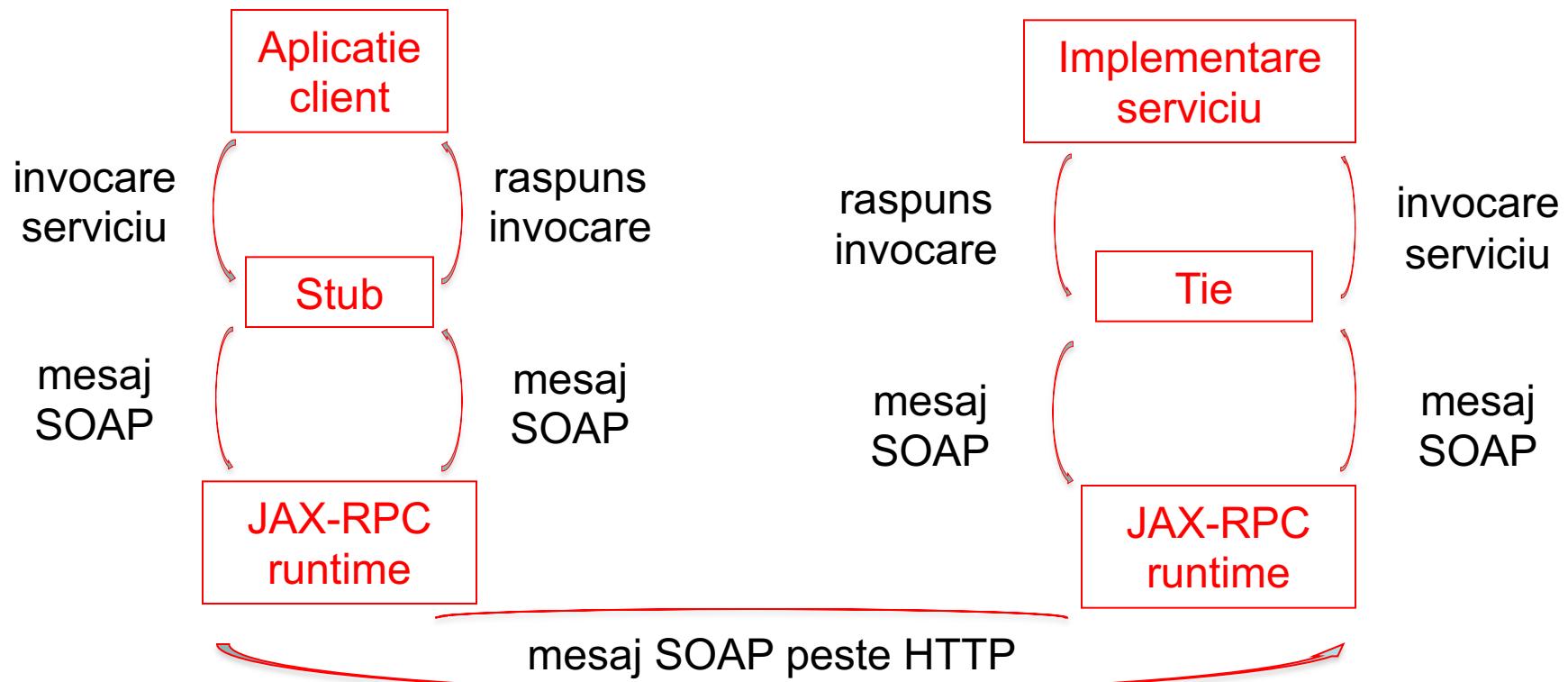
JAX-RPC runtime system

Invocarea serviciului de catre **clienti** ruland in alta masina virtuala (de regula in alta masina) reclama mecanisme speciale oferite de **JAX-RPC runtime system**



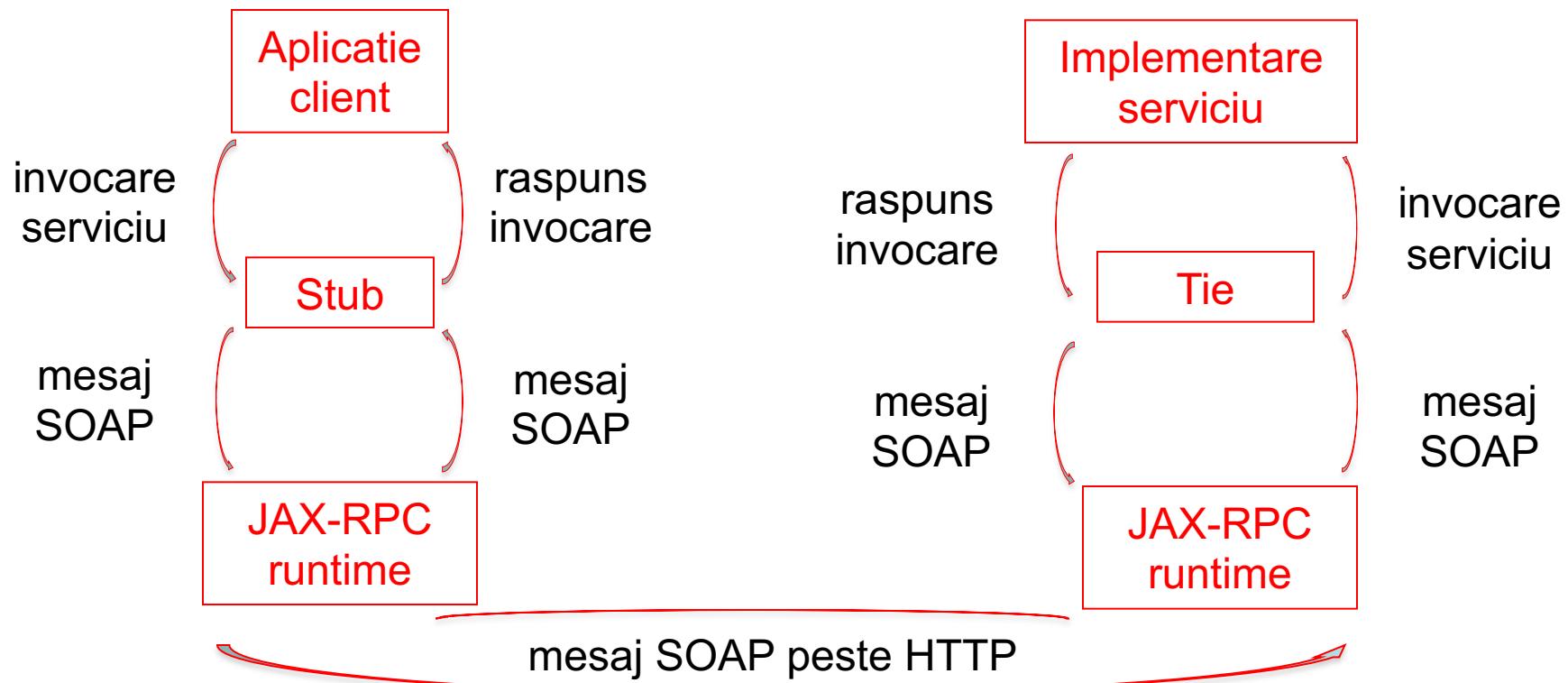
Stub

- O **invocare** este delegata de **stub** sistemului de runtime client care o transmite ca mesaj SOAP
- **Raspunsul** trimis de server ajunge la sistemul de **runtime client** si este pasat la **stub**; acesta il returneaza aplicatiei client ca rezultat al invocarii



Tie

- Similar, la **server**, mesajul SOAP primit ca urmare a invocarii unui client este convertit în **apel de metoda** de către **tie**
- acesta extrage din mesajul SOAP **numele metodei** și **parametrii**, și invoca serviciul
- **tie** convertește rezultatul executiei metodei în mesaj SOAP de raspuns, returnat sistemului de runtime al clientului



Constructia si instalarea serviciului

Generarea **stubs**, **ties**, rutine de **serializare** si fisiere **WSDL** se face cu instrumente dedicate

De **ex.** implementarea in Sun Java System Application Server ofera instrumentul **wscompile**

- genereaza stubs, ties, serializatoare
- genereaza un document WSDL din descrierea serviciului
- genereaza interfata serviciului din fisierul WSDL
- genereaza modelul structurilor de **date interne** din descrierea serviciului sau WSDL

Pentru generare, **wscompile** foloseste un fisier de **configurare** care contine informatia necesara

Caracteristici JAX-RPC

- Functionalitati similare cu Java RMI
 - codul **clientului** apare ca facand invocari de metode ale unor **obiecte locale** (proxy-uri)
 - folosesc **interfete** pentru descrierea serviciilor
 - folosesc **referinte / URI**-uri la obiecte remote / resurse pentru a invoca servicii;
 - **convertesc** (marshaling/unmarshaling) invocari si raspunsuri la forme potrivite pentru transport prin internet – la JAX-RPC asigurate de Service/Client Runtime Environment

Caracteristici JAX-RPC

Diferente fata de Java RMI

- in JAX-RPC, mesajele intre client si server folosesc **reprezentari XML** → gama mai larga protocoale de transport (inclusiv HTTPS)
- **interoperabilitate** - JAX-RPC mai usor de folosit decat RMI/IIOP
 - descrierile XML folosite sunt textuale (usor de inteles de dezvoltatori)
- JAX-RPC suporta un **set limitat de tipuri** Java – corespunzator tipurilor de date din SOAP XML

Bibliografie

- Ethan Cerami, Web Services Essentials. Distributed Applications with XML-RPC, SOAP, UDDI & WSDL. O'Reilly 2002
- George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. Distributed Systems Concepts and Design, ed. 5, Addison Wesley
- Eric Armstrong, The JavaTM Web Services Tutorial, Copyright © 2003 Sun Microsystems, Inc.
- Kim Topley, Java Web Services in Nutshell, O'Reilly 2003
- <http://www.inf.fu-berlin.de/lehre/SS03/19560-P/Docs/JWSDP/tutorial/doc/JAXRPC3.html>