Sisteme distribuite – Tehnologii
7. Standarde WS: XML, WSDL, SOAP, UDDI

XML

De ce XML?

- Paginile Web – au nevoie de un om de a înţelege ceea ce semnifica .
- Ce se întâmplă dacă informatia Web este disponibil într-o formă care ar putea fi uşor de utilizat de către alte medii?
- Zilele timpurii ale Webului: *screen scraping* – HTML este analizat şi înţelesul său se deduce bazează pe ipoteze despre aspectul paginii, de la poziţiile de masa, etc
  - cauză pierdută din cauza web designeri schimba aspectul paginii frecvent pentru a păstra propriile site-uri interesante
- XML – eXtensible Markup Language (W3C, 1998) - iniţial un HTML "mai bun"
  - În curând a devenit evident că XML este bun ca format de schimb de date.
- Informatia disponibila pentru alte programe pe Web: publicare in format XML
  - Necesitatea de a defini un vocabular XML care descrie datele aplicatiei sau de a folosi un vocabular standard, în cazul în care unul adecvat există.
  - Activitatea predominantă în urma publicării specificaţia XML a fost definirea de vocabulare standard, cum ar fi:
    - Mathematical Markup Language (MathML),
    - Chemical Markup Language (CML)
    - Meat and Poultry Markup Language (mpXML)
    - …
- XML : textual si neutru fata de arhitectura , deci nu a fost nici o confuzie cu privire la detaliile de nivel scăzut, cum ar fi ordinea de octeţi într-un nr. întreg.

Prezentare generală a XML

- Permite reprezentarea structurata de date arbitrare.
- Fisierele XML sunt fişiere text simple, care pot fi editate cu orice editor .
- XML e numit un limbaj de marcare, deoarece datele sunt "marcate" prin tag-uri
- Exemplu: o specificare XML

```
<Person>
    <FirstName>Mickey</FirstName>
    <LastName>Mouse</LastName>
    <Age>75</Age>
</Person>
```

  - Person, FirstName, LastName, si Age sunt taguri (etichete).
  - Un *tag de start* este marcat de “<” si “>”iar un *tag de sfarsit* de “</” so “>”.
  - Identificatorii Person, FirstName sunt specifice aplicatiei, nu sunt parte a standardului XML.
  - Între tag-ul de început şi tag-ul final este *conţinutul* tag-ul.
  - Combinaţie tag-uri de început şi sfârşit plus conţinut este menţionată ca *element*.
  - Tag-uri pot fi conţinutul pentru alte tag-uri; Tag-ul Age face parte din conţinutul de tag-ul Person .
  - Tag-uri trebuie să fie strict imbricate, ceea ce duce la o ierarhie sau structura arbore-in reprezentarea datelor .

Prezentare generală a XML

- Tag-urile pot avea unul sau mai multe atribute - de ex ("definiţie de tip"):

```
<struct name="Person">
  <member type="string" name="first_name"/>
  <member type="string" name="last_name"/>
  <member type="int" name="age"/>
</struct>
```

- Numele si tipul sunt numite atribute, de exemplu, name este un atribut al tag-ul struct .
- Valoarea unui atribut este scris între ghilimele duble, de exemplu, Person este valoarea de atributului name.
- Valorile atributelor fac parte din conţinutul unui tag,
  - nu există norme absolute dacă datele ar trebui să fie introduse ca si conţinutul unei etichete sau ca o valoare a unui atribut al acelui tag.
- Dacă nu există nici un conţinut pentru o etichetă, tag-ul poate fi inconjurata de "<" şi "/>" în loc de o etichetă de sfârşit explicită.
- XML poate fi folosit pentru a descrie atât tipurile cat şi instanţele de date (primul exemplu), pentru a fi tratate de către un middleware.

Interpretoare (parsere) XML

- Pentru a citi un document XML, o aplicatie utilizează un parser-ul pentru a obţine datele conţinute în document.
- Un parser-ul de obicei constă dintr-un API care-i permite programatorului sa aleaga la ce elemente să se uite în document.
- *Arhitectura Microsoft .NET* furnizează un dezvoltator cu mai multe clase, pentru a accesa datele într-un document
- *Grupul Apache* dezvoltă un parser-ul numit Xerces care funcţionează între platforme.
- Prin servicii Web, o cerere trece un document XML pe Internet prin protocoalele de transport diferite.
  - Prin urmare, fie un client sau un program din partea serverului trebuie să parse XMLul pentru a ajunge la datele din document.

Procesare de instructiuni & elementul Root

- Prima parte a oricărui document XML este Instrucţiunea Processing (PI).
  - Aceasta spune parser-ul că data este un document XML şi care-I versiunea de XML utilizată (în exemple va fi 1).
  - Începutul documentului acum arata astfel: <?xml version="1.0" ?>

- Pentru a începe descrierea de date, un element rădăcină trebuie să fie prezent.
  - Acesta este elementul cel mai periferic din document.
  - Un element este pur şi simplu o etichetă care arată de mult ca un tag-ul HTML, dar în cazul XML programator alege numele de tag-ul.
  - Exemplu: BOOK este elementul rădăcină. <?xml version="1.0" ?> <BOOK> </BOOK>
  - Un document XML trebuie să aibă doar un element rădăcină
  - De obicei, elementul rădăcină începe definirea unui document SOAP sau un fişier WSDL.

Atributele

- Informaţia suplimentara adăugata la un element este un *atribut*
- Exemplu:

  <?xml version="1.0" ?> <BOOK TITLE="Distributed Systems">
  </BOOK>
- Atributele apar întotdeauna, ca parte a elementului de deschidere şi pot fi in orice element din document
- Atributele definesc adesea spaţii de nume sau locaţii, cum ar fi nodul următor SOAP, pentru documentul XML.
- Atribut central pentru un document: toate informaţiile îşi au reşedinţa în termen de atribute

  <?xml version="1.0" ?>
  <BOOK TITLE="Distributed Systems"
  PAGECOUNT="400"
  AUTHOR="Ion Ionescu"
  PUBLISHER="New House Press"/>.

Spatii de nume (namespaces)

- Pentru a permite utilizarea aceluiaşi nume cu sensuri diferite în contexte diferite, schemele XML poate defini un spaţiu de nume.
- Un spaţiu de nume = un set de nume unice care sunt definite pentru un anumit context şi care sunt conforme cu normele specifice
  - Acelaşi nume poate fi folosit în spaţii de nume diferite, fără a provoca un conflict de nume duplicat.
  - Asigura că numele de elemente folosite în documentul XML sunt unice.
  - Definiţia spaţiu de nume apare în elementul rădăcină (elementul ultraperiferice) si utilizeaza un URL ca un identificator unic.
- Exemplu:

  <BOOK XMLNS:WEBSERVICES="www.newhouse.com/XML"></BOOK>
  - Atunci, toate elementele fii ale BOOK încep cu spaţiul de nume.

```
<?xml version="1.0" ?>    <BOOK XMLNS:WEBSERVICES="www.newhouse.com/XML">
    <WEBSERVICES:TITLE>Distributed Systems</WEBSERVICES:TITLE>
    <WEBSERVICES:PAGECOUNT>400</WEBSERVICES:PAGECOUNT>
     <WEBSERVICES:AUTHOR>Ion Ionescu</WEBSERVICES:AUTHOR>
     <WEBSERVICES:PUBLISHER>New House Press</ WEBSERVICES:PUBLISHER>
 </BOOK>
```

- In WS: utilizate de obicei ca o modalitate de a reprezenta diferite elemente care sunt dependente de furnizor sau de a sprijini tipuri primitive in scheme.

Spatii de nume in XML

- Un spaţiu de nume XML defineşte o colecţie de nume şi este identificat printr-o referinţă URI.
- Exemplu: xmlns="http://simple.example.com/CInfoXmlDoc".
- Numele în spaţiul de nume pot fi folosite ca tipuri de elemente sau atribute într-un document XML.
- Combinaţia de URI şi tipul de element sau un nume de atribut cuprinde un nume unic universal, care evită coliziuni.
- Exemplu: un spaţiu de nume care defineşte tipuri de elemente sle documentului ContactInformation, cum ar fi Name si Address.
  - aceste tipuri de elemente sunt unice în contextul informaţii de contact.
  - în cazul în care documentul a inclus si un alt context de spaţiu de nume, cum ar fi BankInformation care are definite tipuri de elemente Name şi Address proprii, aceste două spaţii de nume ar fi separate şi distincte
  - un nume şi o adresa utilizate în contextul BankInformation, nu ar intra în conflict cu un nume şi o adresa utilizate în contextul ContactInformation.

XML bine-formt si valid

- Înseamnă că documentul respectă toate regulile şi conţine toate informaţiile specificate în nici un Document Type Definition (DTD) sau într-o schemă de definire XML (XSD).
  - Ambele acţionează ca un bilet de ambalare pentru a documentelor XML, specificând datele trebuie să fie prezente în document.
  - Validarea unui document fata de o schema sau un DTD este un proces costisitor şi, prin urmare, probabil, apare numai în timpul dezvoltarii de software SOAP.
- Un document XML bine-format urmează regulile stabilite de W3C: :
  - trebuie să existe unul sau mai multe elemente,
  - nu poate exista decât un element rădăcină, care nu este supraimpus de orice alt element,
  - fiecare tag care începe trebuie să aibă o etichetă de sfârşit excepţia cazului în care este un element gol
- Exemplu 1:

  <?xml version="1.0" ?> <BOOK TITLE="Distributed Systems">

  Nu este bine format (lipsa / la sfarsit)
- Exemplu 2: elementul rădăcină este supraimpus de un alt tag

```
<?xml version="1.0" ?>
  <BOOK TITLE="Distributed Systems ">
     <AUTHOR> Ion Ionescu </AUTHOR>
     <BOOKDATA>  <PAGECOUNT>400</PAGECOUNT>
    <PUBLISHER> New House</PUBLISHER>   </BOOK>   </BOOKDATA>
```

DTD si XSD

- Un DTD sau XSD descrie structura unui document XML.
  - informaţiile cu privire la tag-urile documentului XML corespunzător pot avea ordinul acestor taguri
  - Validarea unui document XML garantează că documentul urmează structura definita în DTD sau XSD şi că aceasta nu are nici un tag XML incorect.
  - Sisteme ce schimba documente XML pentru anumit scop pot fi de acord asupra unui DTD sau XSD unic şi să valideze toate documentele XML primite în acest scop fata de DTD / XSD convenit înainte de prelucrarea documentelor
- DTDs provin din vechiul standard *Serialized General Markup Language* (SGML) utilizat in industria de publicare a cartilor.
  - Nu oferi lucrurile necesare pentru programarea comuna, cum ar fi tipurile sau ordinea
  - Permite unui utilizator să specifice referinţele entitate si de asubstitui valori in /out in documente
  - Multe instrumente, cum ar fi Sun Microsystems Forte, permite crearea lor cu uşurinţă
- Dezavantaje DTD:
  - DTD apar rar în afara industriei de publicare
  - Neclaritate şi sintaxa dificila a DTD este cauza pentru care majoritatea dezvoltatorilor prefera schemele ca o modalitate de a valida documentele XML.
  - DTD nu oferă nici o posibilitate reală de a exprima tipuri de date sau relaţiilor structurale complexe
- XSD standardideaza definiţiile formatul documentelor XML.

Exemplu de DTD

Un DTD simplu pentru exemplul precedent: :

- trebuie să recunoască faptul că BOOK este elementul rădăcină şi TITLE, PAGECOUNT, AUTHOR, şi EDITOR sunt fii.
- Deoerece BOOK este un element rădăcină, nu poate fi opţional, dar toate celelalte elemente pot fi.
- DTD-ul trebuie să recunoască faptul că titlul este un atribut de carte.

```
<?xml version="1.0" ?>
<!-- This is a comment -->
<!-- The following code is the DTD -->
<!-- The PI and the DTD are the prolog of the document -->
<!DOCTYPE BOOK [
<!ELEMENT BOOK (PAGECOUNT?,AUTHOR+,PUBLISHER+)>
<!ATTLIST BOOK TITLE CDATA #REQUIRED>
<!ELEMENT PAGECOUNT (#PCDATA)>
<!ELEMENT AUTHOR (#PCDATA)>
 <!ELEMENT PUBLISHER (#PCDATA)>    ]>
<BOOK TITLE="Distributed Systems">
<PAGECOUNT>400</PAGECOUNT>
<AUTHOR>Ion Ionescu</AUTHOR>
<PUBLISHER>New House Press</PUBLISHER>
</BOOK> The DTD at the beginning of the document is clearly not XML.
```

Specificarea cantităţilor se face cu simbolurile +, *, and ?.

+ inseamna 1 sau mai multe elemente pe cand * inseamna 0 sau mai multe.

? Inseamna ca elementul este optional.

Un alt exemplu: XML si DTD

```
<?xml version="1.0" encoding="ISO-8859-1"
    standalone="yes"?>
<ContactInformation>
  <Name>Ion Ionescu </Name>
   <Address>
      <Street>B-dul V.Parvan 4</Street>
      <City>Timisoara</City>
      <State>Timis</State>
      <Country>RO</Country>
   </Address>
   <Phone>0256666333</Phone>
   <EMail>ion_ionescu@yahoo.com</EMail>
 </ContactInformation>
```

```
<!ELEMENT ContactInformation
  (Name, Address, Phone,EMail)>
<!ELEMENT Name(#PCDATA)>
<!ELEMENT Address (Street, City, State,
    Country)>
<!ELEMENT Street (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT EMail (#PCDATA)>
```

XML si XSD

```
<?xml version="1.0" encoding="ISO-8859-1"
      standalone="yes"?>
<ContactInformation
      xmlns="http://simple.example.com/CInfoXmlDoc"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance"       xsi:schemaLocation=
      "http://simple.example.com/CInfoXmlDoc
      file:./CInfoXmlDoc.xsd">
<Name> Ion Ionescu </Name>
 <Address>
<Street>B-dul Vasile Parvan 4</Street>
<City>Timisoara</City>
<State>Timis</State>
<Country>RO</Country>
</Address>
<HomePhone>0256666333</Phone>
<EMail> ion_ionescu@yahoo.com </EMail>
</ContactInformation>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://simple.example.com/CInfoXmlDoc"
      xmlns=" http://simple.example.com/CInfoXmlDoc"
      elementFormDefault="qualified">
<xsd:element name="ContactInformation">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:string" />
      <xsd:element name="Address">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Street"  type="xsd:string" />
             <xsd:element name="City"    type="xsd:string" />
             <xsd:element name="State"   type="xsd:string" />
             <xsd:element name="Country" type="xsd:string" />
           </xsd:sequence>
         </xsd:complexType>
       </xsd:element>
       <xsd:element name="HomePhone" type="xsd:string" />
       <xsd:element name="EMail" type="xsd:string" />
    </xsd:sequence>
   </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Schema generta de Visual Studio .NET

```
<?xml version="1.0" ?>
 <xs:schema id="NewDataSet"
       targetNamespace="http://www.newhouse.com/~vs1C0.xsd"
       xmlns:mstns="http://www.newhouse.com/~vs1C0.xsd"
       xmlns="http://www.newhouse.com/~vs1C0.xsd"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
       attributeFormDefault="qualified"
       elementFormDefault="qualified">
<xs:element name="BOOK">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="PAGECOUNT" type="xs:string"
        minOccurs="0"          maxOccurs="3"  msdata:Ordinal="0" />
       <xs:element name="AUTHOR" type="xs:string" minOccurs="0"
        msdata:Ordinal="1" />
       <xs:element name="PUBLISHER" type="xs:string" minOccurs="0"
        msdata:Ordinal="2" />
    </xs:sequence>
   <xs:attribute name="TITLE" form="unqualified" type="xs:string"/>
   </xs:complexType>
  </xs:element>
 <xs:element name="NewDataSet" msdata:IsDataSet="true"
       msdata:EnforceConstraints="False">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
     <xs:element ref="BOOK" />
    </xs:choice>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

- *Visual Studio.NET* genereaza scheme automat bazadu-se pe un document XML dat
- namespaces trtez in mod specil necesitatile *Visual Studio*
- Primul xs:element defineste o cerinta pentru PAGECOUNT in documentul XML:
  - Este un string cf. tipului atributilui
  - minOccurs setat la 0 indica ca PAGECOUNT nu este cerut;
  - maxOccurs indica ca PAGECOUNT poate aparea cel multde trei ori.
- xs:sequence tag permite determinarea ordinii elementelor in schema

Avantajele XML

- conceput exact pentru scopul schimbului de date şi-a demonstrat puterea în timp.
- Un limbaj de marcare simplu, flexibil, bazat pe text
- standard acceptat de industrie, permite furnizorilor de servicii şi solicitaorilor acestora sa comunice unii cu alţii într-o limbă comună
- nu depinde de o platformă propriera sau de tehnologie,
- mesajele în XML pot fi comunicate prin Internet folosind protocoale standard de Internet, cum ar fi HTTP.
- produs al World Wide Web Consortium (W3C) => modificările la acestuia vor fi susţinută de toţi jucătorii de conducere
- Acest lucru garantează că odata cu evolutia XML, serviciile Web pot evolua, de asemenea, fără sa afectecze compatibilitatea cu versiunile anterioare
- Principale: Structurat, Portabil, Extensibil, Format text

WSDL

Contract al servicii

- Fiecare serviciu are o interfata bine definita, formala, numit contract de servicii:
  - defineşte în mod clar ceea ce face de serviciu
  - separa in mod clar interfata de servicii accesibil din exterior de implementarea serviciului
- Elemente ale contractului de servicii :
  - Nume de servicii: nume prietenoase (pseudonime), nume unice citibile de masina
  - Numar de versiune: sprijină ciclului de viaţă al serviciului.
  - Pre-condiţii: condiţii care trebuie îndeplinite înainte a fi utiliza serviciul, de exemplu, o operaţiune nu poate fi accesibila între miezul nopţii şi ora 2 am.
  - Serviciul de clasificare: note şi cuvinte cheie care identifica domeniu de afaceri (e), că acceptă serviciul ca intratre in “cartea galbena"
- Contractul de servicii poate fi
  - în mod explicit definit prin WSDL, XML Schema, cadru de politica WS, sau
  - In mod implicit definit pe baza mesajelor de intrare ce acceptă, mesajele de ieşire cu care răspunde, precum şi activităţile de afaceri care le implementează
- Pentru implementari SOA bazate pe servicii Web
  - WSDL este utilizat pentru a defini elemente cheie a contractelor de servicii
  - alte elemente care nu pot fi exprimate în WSDL sunt definite folosind cadre de politică WS sau documentate într-un document sau o foaie de calcul.

Web Services Description Language - evolutie

- defineşte un mod standard de a preciza detaliile unui serviciu Web.
- este o schema XML cu scop general care poate fi folosita pentru a specifica detaliile interfeţelor serviciului Web, legaturile, precum şi alte detalii de implementare
- descrie
  - Ce metode publice sunt disponibile şi
  - Unde este localizat serviciul
- WSDL 1.1 fost facut disponibil in 2001
  - In 2002, un draft de lucru WSDL 1.2 a fost lansat
  - WSDL este un protocol universal acceptat pentru a descrie serviciile Web
- WSDL 2.0 este o recomandare W3C emisa în 2007.
  - WSDL 1.1 si 2.0 sunt conceptual foarte asemănătoare. .
  - WSDL 2.0 a beneficiat de un proces de revizuire lung şi atent.
    - WSurile cate suporta plicipiile arhitecturale Web precum REST sunt primele care adopta WSDL 2.0.

Structura unui document WSDL

Şase elemente majore sunt incluse într-un document WSDL:

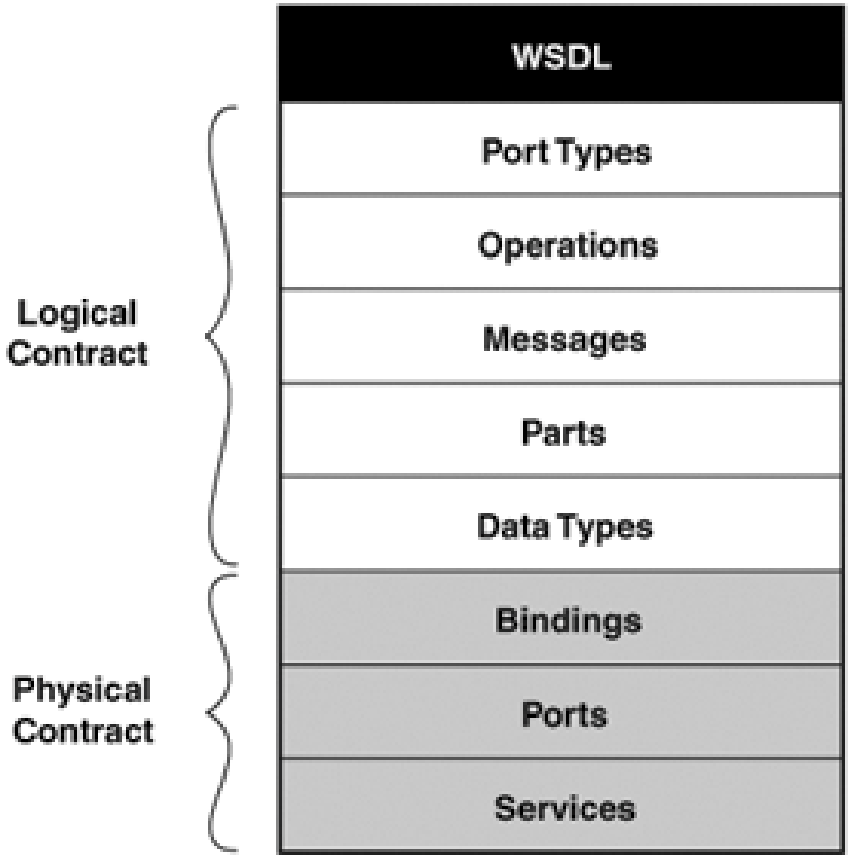
- Tipuri de date pe care le foloseşte WS
- Mesajele pe care le utilizează
- Operaţiuni pe care le efectuează
- Protocoale de comunicare care le utilizează
- Adrese individuale de legare
- Agregarea unei multimi de porturi relationate

- Portul descrie *CE* face un WS,
- Legatura descrie *CUM*,
- Serviciul descrie *UNDE*.

Structura WSDL

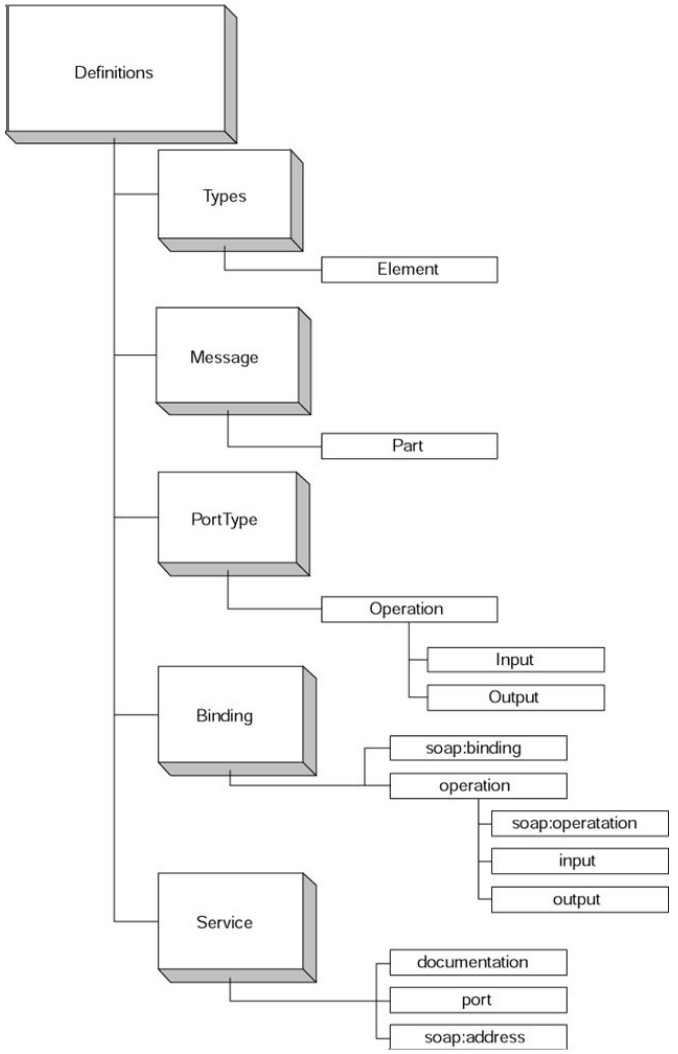
1. Contractul logic defineşte interfaţa public care este independenta de formatele de transport, de transmitere si de limbajele de programare.

2. Contractul fizic defineşte legaturile pentru formatele de transport si de transmitere, şi mai multe contracte fizice pot fi definite pentru fiecare contracte logice.



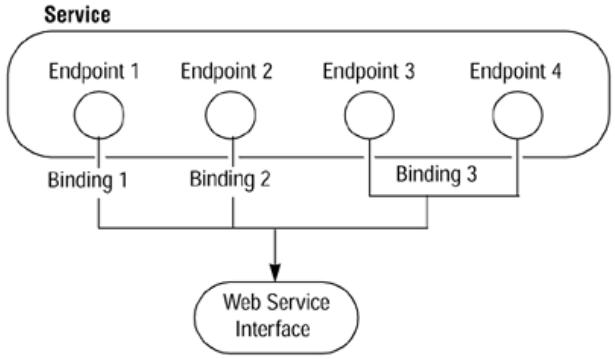
Elemente parinte ale unui document WSDL

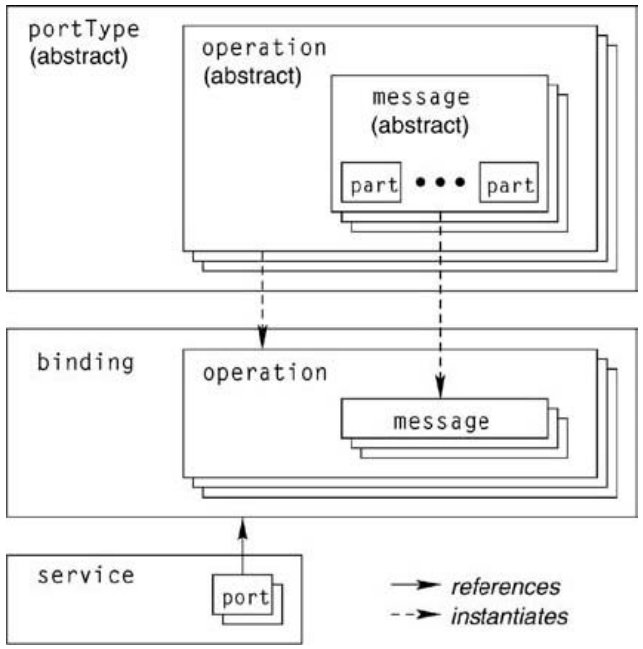
- Definitions:
  - Elementul rădăcină al documentului WSDL.
  - Defineşte multe dintre spaţiile de nume folosite pentru o descriere speciala.
- Types:
  - Defineşte elemente şi tipuri primitive găsite în XMLul cererii şi răspunsului SOAP.
- Message:
  - Numeste mesajele de cerere şi răspuns.
- PortType:
  - Laega o cerere speciala şi un mesaj de răspuns la un anumit serviciu.
- Binding:
  - Indică tipul de transport utilizate de serviciu.
  - Descrie conţinutul mesajului, cum ar fi literal, in sensul de a lua XML la valoarea nominală, sau image / gif. .
- Service:
  - Numeste serviciile
  - Oferă un element pentru a documenta ceea ce serviciul de fapt, .



Gramatica WSDL

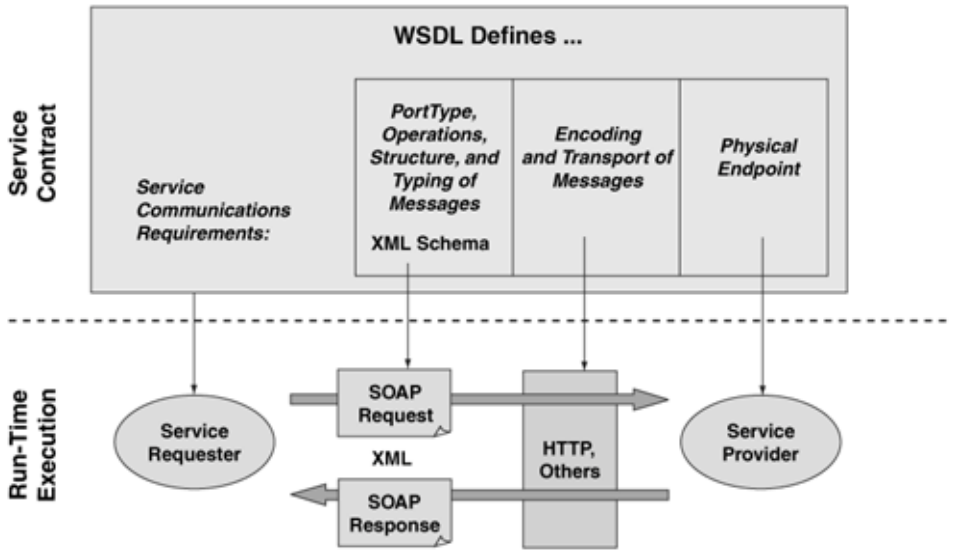
- Descrie o colecţie de obiective de comunicare (endpoints), numite porturi**.**
- Parti abstracte:
  - Data care se schimba este specificat in ***mesaje.***
  - Fiecre tip de ctiune care sunt permise intr-un endpointeste considerta o ***operatie.***
  - Colectii de operatii ale unui endpoint sunt grupate intr-un ***tip de port***.
- Protocolul şi specificaţiile formatului de date pentru un anumit tip de port sunt specificate ca ***legaturi.***
  - Port: definit prin asocierea de o adresă de reţea cu o legatura reutilizabila
  - Un mecanism comun de legare une impreuna toate protocolurile si formatele de date cu un mesaj, operatie sau endpoint





Utilizare WSDL

- Definiţii pot fi folosite pentru a genera dinamic mesajele SOAP, care sunt schimbate pentru a executa serviciul, ilustrat printr-un sablon cerere / răspuns.



Definitions- exemplu

- Este elementul radacina a documentului WSDL.
- Exemplu: <definitions name="GetStockQuote></definitions>
  defineste numele serviciului Web.
- Tag definiţiilor va sunt necesare pentru a defini mai multe spaţii de nume diferite pentru a sprijini diferitele tipuri primitive şi tipurile create de WS GetStockQuote
- Exemplu:

```
<definitions name="GetStockQuote
targetNamespace="http://advocatemedia.com/GetStockQuote.wsdl"
  xmlns:myns = "http://advocatemedia.com/GetStockQuote.wsdl"
  xmlns:myXSD = "http://advocatemedia.com/GetStockQuote.xsd"
  xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
 </definitions>
```

  - targetNamespace: Defineşte spaţiul de nume pentru acest document .
  - myns: O definiţie mai precisă pentru acest document .
  - myXSD*:* Spaţiu de nume pentru tipurile de schemă definite aici .
  - xmlns:soap Spaţiu de nume pentru elementele SOAP utilizate în document .
  - xmlns Setează spaţiul de nume implicit pentru elementele SOAP

Types - exemplu

- defineşte diferitele elemente utilizate în WS.
- Elementul tip defineşte o schemă, astfel încât un document WSDL este capabil de a utiliza tipurile definite în standardul schemă, mai degrabă decât să creeze propriile sale
- Exemplu:

```
<types>
 <schema  targetNamespace="http://advocatemedia.com/GetStockQuote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="StockQuoteRequest">
       <complexType>
         <all>      <element name="symbol" type="string"/>      </all>
       </complexType>
    </element>
   <element name="StockQuoteResponse">
        <complexType>
          <all>       <element name="price" type="float"/>        </all>
       </complexType>
    </element>
  </schema>
</types>
```

StockQuoteRequest / Response sunt elementele parinte ale documentelor cerere/raspuns.

Elementele care contin valorile (adica., symbol si price), sunt fii.

Tipurile pentru aceste elemente fii sunt definite ca: string pt. symbol,float pt. price.

Message - exemplu

- Descrie şi numeste atât mesajul de cerere cat şi mesajul de răspuns.
- Oferă o cale înapoi la tipurile din mesaj .
- Exemplu:

```
<message name="GetStockQuoteRequest">
    <part name="body" element="myXSD:StockQuoteRequest"/>
</message>
<message name="GetStockQuoteResponse">
    <part name="body" element="myXSD:StockQuoteResponse"/>
</message>
```

- Definitiile elementelor au numele elementelor parinte in documentul *SOAP* document.
- Spaţiu de nume MyXSD este folosit ca un prefix, astfel că aplicatia care foloseste documentul WSDL stie sa găsească defs pentru aceste în partea din doc a tipurilor de elemente
- Mesajul de cerere şi de răspuns au un nume a.i. aplicatille ce folosesc acest serviciu cunosc numele mesajelor pentru a expedia & receptiona atunci când se utilizează un serviciu
- Mesajul este independent de protocol deoarece nu exista nici o mentiune la HTTP sau SMTP.
- Valoarea numelui poate fi orice => se recomanda selectarea a ceva util

PortType - exemplu

- Pentru a utiliza cele două mesaje definite în secţiunea anterioară cu elementul de mesaj, sunt definite ca cererea şi răspuns pentru un anumit serviciu.
- Acest lucru se face cu comanda portType:

```
<portType name="GetStockQuotePort">
    <operation name="GetStockQuote">
        <input message="myns:GetStockQuoteRequest"/>
        <output message="myns:GetStockQuoteResponse"/>
    </operation>
</portType>
```

GetStockQuote este numele operatiei.

  - Operaţia este cererea preţului unui anumit stoc
  - Răspunsul este sub forma preţul pentru acest stoc. .
  - Mesajele de intrare şi ieşire doar combină cele două definiţiile folosite mai devreme, astfel încât clientul să ştie că un mesaj de şi unul de răspuns face parte dintr-o anumită metodă într-un serviciu Web. .
- Un portType este o definiţie abstractă a unei interfeţe .
  - Acesta este abstract, în sensul că descrie interfaţa de funcţionare a unui serviciu fără a intra în detalii cu privire la structura de date a diferiţilor parametri .
- Un portType, în esenţă, constă dintr-una sau mai multe operaţiuni, fiecare constând din mai multe mesaje.

Un alt examplu

```
<definitions name="MyAccountService">
  <types/>
      <message name="AccountIF_balance"/>
      <message name="AccountIF_balanceResponse">
        <part name="result" type="int"/>
      </message>
      <message name="AccountIF_deposit">
        <part name="amount" type="int"/>
      </message>
      <message name="AccountIF_depositResponse"/>
      <message name="AccountIF_withdraw">
        <part name="amount" type="int"/>
      </message>
      <message name="AccountIF_withdrawResponse"/>
  <portType name="AccountIF">
    <operation name="deposit" parameterOrder="amount">
      <input message="AccountIF_deposit"/>
      <output message="AccountIF_depositResponse"/>
    </operation>
    <!--similar definitions for withdraw and balance -->
  </portType>
```

Binding - exemplu

- Indică tipul de transport pe care un WS il utilizeaza.
  - Un binding descrie modul în care defs abstractă a unui portType sunt convertite într-o reprezentare concreta
  - De exemplu:
    - În cazul în care un doc XML transmite conţinutul său în corp, atunci doc WSDL trebuie să definească asta .
    - În cazul în care documentul transmite conţinutul său în codare base64, atunci acest lucru ar trebui să fie definit
- Există două stiluri de legatura şi de transport din care se poate alege :
  - RPC: indică faptul că un mesaj conţine parametri şi că nu există valori de returnare.
  - Document: atunci când atributul stil conţine documentul, cererea şi răspunsul trec documentelor XML în corpul mesajului SOAP .
- Spaţiu de nume în transportul indică care protocol este utilizat: HTTP / SMTP
- Alte informatii :
  - Elementele de intrare şi de ieşire: definesc conţinutul mesajelor de cerere şi de răspuns. .
  - Utilizeaza conţinutul documentului cu / fără nici o codare (poate fi base64, image / gif).
- Exemplu:

```
<binding name="GetStockQuoteBindingName"  type="GetStockQuotePort ">
   <soap:binding    style="rpc"    transport= "http://schemas.xmlsoap.org/soap/http"/>
       <operation name="GetStockQuote">
        <soap:operation     soapAction="http://advocatemedia.com/GetStockQuote"/>
        <input>     <soap:body use="literal"/>   </input>
         <output>  <soap:body use="literal"/>   </output>
      </operation>
</binding>
```

Definirea serviciului - exemplu

- defineşte numele de serviciu împreună cu documentaţia şi locaţia serviciului

- Exemplu 1:

```
<service name=
    "AdvocateMediaGetStockQuotes">
<documentation>Simple Web Service to
    Retrieve a stock quote</documentation>
<port name="GetStockQuotePort"
    binding=
    "myns:GetStockQuoteBindingName">
  <soap:address  location=
"http://advocatemedia.com/GetStockQuote"/>
 </port>
</service>
```

- Exemplu 2:

```
<service name="MyAccountService">
  <port name="AccountIFPort"
     binding="AccountIFBinding">
<address location="http://localhost:8080/account"/>
  </port>
 </service>
```

Elementul de documentare oferă unui dezvoltator posibilitatea de a furniza aditionale informaţii despre ceea ce face servciul Web.

Soap:address numeste WSul ca un intreg

Utilizare import

- un mod alternativ şi, probabil, mai uşor de scris fişierul WSDL .
- implică definirea tuturor tipurilor într-un fisier schemă XML redusă (XSD) punand celelalte definitii relevante pentru WS in fisierul WSDL.
- În acest fel, elementul schemă şi totii fiii sunt într-un fişier separat
- Dacă utilizaţi aceleaşi elemente în diferite Web Services, aveţi posibilitatea să mutaţi cu uşurinţă definiţiile schemă de la aplic al aplic
- De exemplu:pot exista mai multe WSuri legate de stocuri care utilizeaza aceleasi tipuri si acelaseasi variabile – un fisier XSD poate suporta toate serviciile diferite.

Ex. GetStockQuote.xsd care sa fie importat

```
<schema targetNamespace=
   "http://advocatemedia.com/GetStockQuote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
   <element name="StockQuoteRequest">
    <complexType>
      <all> <element name="symbol" type="string"/> </all>
    </complexType>
   </element>
   <element name="StockQuoteResponse">
     <complexType>
      <all> <element name="price" type="float"/> </all>
     </complexType>
   </element>
</schema>
```

WSDL simplificat

```
<definitions name="GetStockQuote
targetNamespace="http://advocatemedia.com/GetStockQuote.wsdl
      "
      xmlns:myns =
       "http://advocatemedia.com/GetStockQuote.wsdl"
      xmlns:myXsd =
       "http://advocatemedia.com/GetStockQuote.xsd"
      xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap"
      xmlns="http://schemas.xmlsoap.org">

  <import
       namespace="http://advocatemedia.com/GetStocks/schemas"
       location ="http://advocatemedia.com/GetStocks/quote.xsd">

  <message name="GetStockQuote">
      <part name="body" element="myXSD:StockQuoteRequest"/>
  </message>
  <message name="GetStockQuoteResponse">
      <part name="body“
      element="myXSD:StockQuoteResponse"/>
  </message>

  <portType name="GetStockQuotePort>
          <operation name="GetStockQuote">
               <input message="myns:GetStockQuoteRequest"/>
               <output message="myns:GetStockQuoteResponse"/>
          </operation>
   </portType>
```

```
<binding name="GetStockQuoteBindingName"
           type="StockQuoteBinding">
    <soap:binding
           style="rpc"
           transport=" http://schemas.xmlsoap.org/soap/http"/>
       <operation name="GetStockQuote">
        <soap:operation
         soapAction="http://advocatemedia.com/GetStockQuote"/>
          <input>
             <soap:body use="literal"/>
          </input>
          <output>
            <soap:body use="literal"/>
          </output>
       </operation>
</binding>

<service name="AdvocateMediaGetStockQuotes">
      <documentation>Simple Web Service to
      Retrieve a stock quote</documentation>
   <port name="GetStockQuotePort"
          binding="myns:GetStockQuoteBindingName">
     <soap:address
      location="http://advocatemedia.com/GetStockQuote"/>
   </port>
</service>

</definitions>
```

SOAP

Simple Object Access Protocol

- permite ca obiectele să nu fie cunoscute inainte de comunicare
- SOAP similar cu Internet Inter-ORB Protocol (IIOP) si cu Java Remote Method Protocol (JRMP)
  - în loc sa foloseasca o reprezentare binară de date, a adoptat o forma text de reprezentare a datelor care folosesc XML.
- este independent atât de limbajul de programare & platformă operaţională
- nu necesită nici o tehnologie specifică la endpoints, ceea ce face complet agnostic la furnizori, platforme, şi tehnologii.
- format text => prietenos cu protocolul de firewall
- este susţinut de principalii companii industriale şi este de aşteptat să aibă suport universal.
  - Dezvoltat ca recomandare W3C
  - SOAP Version 1.2 Recommendation din 2003.
  - Odata cu apariţia WS, SOAP a devenit de facto protocol standard de comunicare pentru crearea si invocarea aplicatiilor expuse într-o reţea.

Asigurarea interoperabilitatii si comunicarii intre aplicatii

- defineşte un protocol de transmitere & un format de codare pentru a reprezenta tipuri de date, limbaje de programare, si baze de date
- se poate folosi o varietate de protocoale standard de Internet (cum ar fi HTTP şi SMTP), pentru transport mesaje
- acesta oferă convenţiile de a reprezenta modelele de comunicare cum ar fi
  - apeluri de procedură la distanţă (RPCs)
  - Mesagerie condusa de documente.
- Numite adesea *Service-Oriented Access Protocol*
- Examplu tipic:
  - permite comunicarea între o aplicatie Web scrisă în ASP.NET pe un Windows Server care comunica cu un WS scris în Perl pe un server de Ubuntu

SOAP defineste …

1. O anvelopa XML pentru mesaje WS
   - Anvelopa SOAP consta dintr-un corp si un antet optional
     - Corpul SOAP contine sarcina aplicatiei,
     - Antetul SOAP cuprinde orice alte date de aplicaţie, cum ar fi securitatea, fiabilitatea, sau informaţii de tranzacţie.
2. un model de procesare pentru intermediarii de servicii Web,
   - specifica modul în care intermediarii de reţea ar putea procesa informaţii de antet SOAP înainte de a oferi corpul SOAP la WS sau client.
3. un algoritm de codificare pentru serializarea obiectelor ca XML .
   - Probleme ! Dificultate principala este aceea ca nu exista o definitoe general acceptata a obiectelor
   - Fiecare limbaj OOP implementeaza numeroase particularitati speciale ale obiectetor, dar adauga si diferente. De exemplu,
     - C++ suporta mostenire multipla dar Java suporta numai mostenire simpla
     - Nu exista o modalitate simpla de a schimba obiecte intre limbaje de programare diferite

Codare SOAP

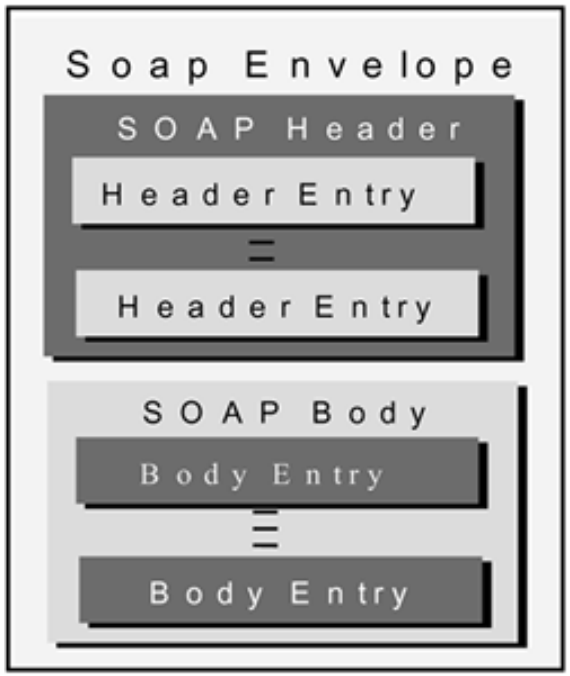
- Motivat de dorinţa de a crea tehnologie pentru obiecte distribuite pe Web
  - Tehnologiile anterioare ale obiectelor distribuite, cum ar fi CORBA, Java RMI, DCOM nu au reuşit să câştige teren semnificative de pe Internet.
  - Aparitia XML => Microsoft a propus ca acesta sa fie folosit ca o modalitate neutra la arhitectura pentru serializarea obiectelor => SOAP a fost propus ca protocol de transport pentru aceste obiecte serializate şi algoritmul serializare a fost supranumită codare *SOAP*.
  - Pentru a utiliza codificarea SOAP este nevoie de o potrivire intre client si implementarile de servicii ce înţeleg algoritmul de codare SOAP .
  - Client şi server se presupune a fi implementate in limbaje OOP !
    - Unul dintre principiile de proiectare din spatele XML este că ar trebui să fie uşor de procesat de o varietate de aplicaţii .
    - Codarea SOAP încalcă acest principiu .

Standardul SOAP

- Dictează
  - modul în care XML arată în documentul SOAP ,
  - modul în care conţinutul acestui mesaj este transmis
  - modul în care acest mesaj este manipulat atât la expeditor şi receptor
- Oferă un set standard de vocabular
- Include modul în care mesajele SOAP ar trebui să se comporte, transporturi diferite utilizate, modul în care erorile sunt manipulate, şi mai mult
- Terminologia legată de SOAP vine în două categorii diferite:
  1. **transmission**
     - modul în care mesajele SOAP se referă la protocol ,
     - modul în care diferitele SOAP nodurile converseaza .
  2. **message**.
     - termeni legati de XML in *SOAP*
- De asemenea, defineşte un set mic de elemente XML pentru a îngloba datelo transmise între noduri
- XMLP intenţionează să fie o rescriere completă a standardului, a.î WS devene şi mai compatibil intre platforme
  - Din cauza acestei schimbări, o mare parte din acest vocabular trebuie să evolueze.
  - Cu toate că numele se va schimba, ideea fiecărui termen vor rămâne aceleaşi .

Elemente

- Un document SOAP adecvat: **envelope + body.**
- Elemente optionale:
  header, fault, attachment
- Intreaga anvelopa—corp plus antet—este un document XML complet.
- Intrarile antetului
  - Pot contine information de utilizare a recipientilor,
  - Pot fi utilizati si de proceseoarele intermediare deoarece permit facilitati aditionale.
- Corpul care contine mesajul este consumat de recipient
- SOAP este agnostic relativ la continutul mesajului
  - singura restrictie este aceea ca mesajul trebuie sa fie in format XML



Structura si exemple

```
<?xml version="1.0" encoding="utf-8"?>

 <soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle=
    "http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  <!-- Header information -->
</soap:Header>

<soap:Body>
   <!-- Body Information -->
</soap:Body>

</soap:Envelope>
```

- Cerere

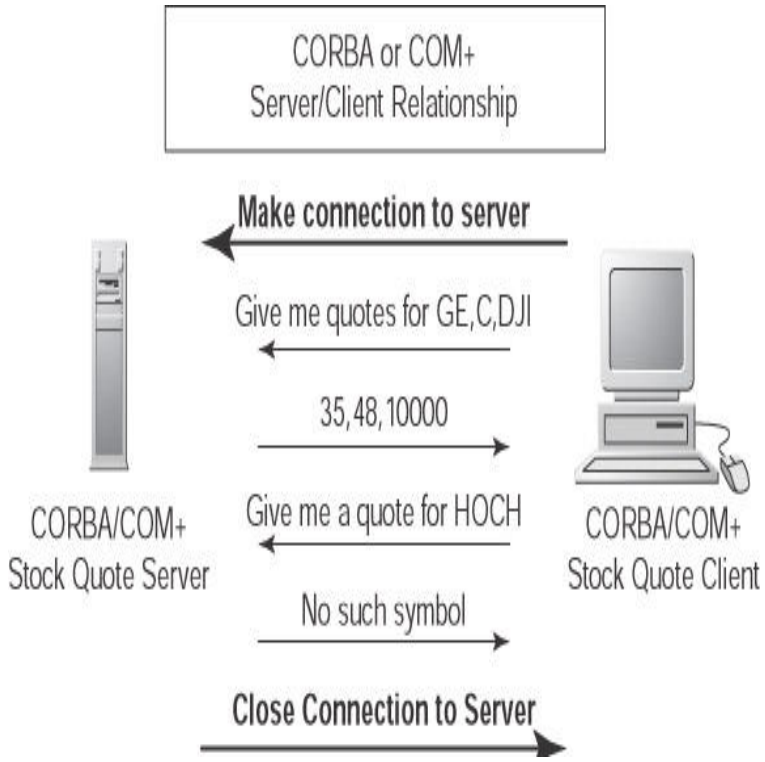
```
<Envelope>
      <Body>
        <deposit>
          <amount
    type="int">700</amount>
        </deposit>
      </Body>
</Envelope>
```

- Raspuns

```
<Envelope>
    <Body>
      <depositResponse/>
    </Body>
</Envelope>
```

SOAP este doar un sistem pentru cereri si raspunsuri

- Exemplu: o conversatie intre doua noduri care discuta despre stoc.
  - In prima cerere, clientul *SOAP* cere valorile stocurilor pentru simbolurile: “GE,” “C,” si “DJI.”
  - Raspunsul este “35,” “48,” si “10,000.”
  - Clientul *SOAP* care intreaba despre stoc citand simbolul “HOCH,” va primi de la receptor informatia ca nu exista un asemenea simbol
- Diferenta intre client si server in COM+, CORBA, si alte tehnologii este faptul ca intreaba conversatie pre intr-o singura conexiune
- Aceasta conexiune este asemanatoare cu telnet—o sesiune este tinuta constant intre client si server.



Anvelopa SOAP - exemple

- Containerul primar pentru o structura de mesaj SOAP si este elementul mandatoriu pentru un mesaj SOAP.
- Este reprezentat ca element radacina a mesajului: Envelope.
- Este declarad uzual ca un element ce utilizeaza spatiu de nume XML http://schemas .xmlsoap.org/soap/envelope/.

Codul urmator arata elementul anvelopa SOAP intr-un mesaj SOAP

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xsi=”http://www.w3c.org/2001/XMLSchema-instance”
  xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
  SOAP-ENV:
   encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”/>
     <!--SOAP Header elements - -/>
     <!--SOAP Body element - -/>
 </SOAP-ENV:Envelope>
```

Antet SOAP - exemplu

- Reprezentat ca prim element fiu imediat unei anvelope SOAP,
- Are nume de spatii calificate
- Poate contine zero sau mai multe elemente fii optionale, care sunt referite ca intrari de antet SOAP.
- Atributul encodingStyle va fi utilizat pentru a defini codarea tipurilor de date utilizate in intrarile elementelor antet.
- Atributul actor si atributul mustUnderstand pot fi utilizate pentru a indica nodul aplicatie SOAP (Sender/Receiver/Intermediary) si pentru a procesa intrarile antetului.

Codul arata un exemplu de reprezentare a unui element antet SOAP in mesajul SOAP.

```
<SOAP-ENV:Header>
  <wiley:Transaction
            xmlns:wiley="http://jws.wiley.com/2002/booktx"
            SOAP-ENV:mustUnderstand="1">
       <keyValue> 5 </keyValue>
  </wiley:Transaction>
 </SOAP-ENV:Header>
```

Antetul SOAP reprezinta o intrare semantica a tranzactiei utilizand astributul mustUnderstand

Atributul mustUnderstand este setat pe "1", ceea ce asigura ca receptorul (URI) acestui mesaj trebuie sa-l proceseze.

Corpul SOAP - exemplu

- Reprezinta informatia de procesare mandatorie si incarcarea intentionata pentru receptorul mesajului
- Un bloc corp a unui mesaj SOAP poate contine urmatoarele:
  - Metoda RPC si parametrii sai
  - Datele specifice aplicatiei tinta
  - Esec SOAP pentru raportarea erorilor si informatie de stare

Exemplu: corp SOAP reprezentand un apel RPC pentru a obtine pretul unei carti de la www.wiley.com

```
<SOAP-ENV:Body>
    <m:GetBookPrice xmlns:m="http://www.wiley.com/jws.book.priceList/">
    <bookname xsi:type='xsd:string'> Developing Java Web
    services</bookname>
    </m:GetBookPrice>
</SOAP-ENV:Body>
```

- Corpul SOAP poate contine informatii ce definesc un apel RPC, documentele in XML, si orice alte date XML cerute ca parte a unui mesaj in timpul comunicarii

Atasament SOAP

- atasamentele in orice format de date pot fi in ASCII sau binare (precum XML sau non-text).

Exemplu: Utilizare “WileyCoverPage.gif” ca atasament si ilustrare utilizare a referintei Content-ID (CID) in corpul mesajului SOAP 1.1ce folosete o entitate referentiata URI etichetata pentru utilizxarea antetelor Content-Location:

MIME-Version: 1.0 Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml; start=”<http://jws.wiley.com/coverpagedetails.xml>” Content-Description: SOAP message description.

```
--MIME_boundary-
    Content-Type: text/xml; charset=UTF-8
    Content-Transfer-Encoding: 8bit
    Content-ID: <http://jws.wiley.com/coverpagedetails.xml>
    Content-Location: http://jws.wiley.com/coverpagedetails.xml
 <?xml version=’1.0’ ?>
    <SOAP-ENV:Envelope  xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”>
       <SOAP-ENV:Body>
         <!-- SOAP BODY - ->
            <theCoverPage href=”http://jws.wiley.com/DevelopingWebServices.gif”/>
         <!-- SOAP BODY - ->
       </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
 --MIME_boundary-
    Content-Type: image/gif
    Content-Transfer-Encoding: binary
    Content-ID: <http://jws.wiley.com/DevelopingWebServices.gif>
    Content-Location: http://jws.wiley.com/DevelopingWebServices.gif
<!--...binary GIF image... - ->
--MIME_boundary-
```

Esec SOAP

- Utilizat pentru a trata erori si pentru a gasi informatia de stare

1. Faultcode:
   - defineste mecanismele algoritmice pentru aplicatia SOAP pentru a identifica esecul.
   - contine valorile standard pentru identificarea erorii sau starii unei aplicatii SOAP

   Valorile elementului fault-code este definit in specificatia SOAP 1.1 :
   - VersionMismatch
   - MustUnderstand
   - Client
   - Server
2. Faultstring : ofera a descriere citibila a erorii SOAP data de aplicatia SOAP
3. Faultactor: ofera info despre actorul SOAP (Sender/Receiver/Intermediary) in mesaj care este responsabil de erorea SOAP la o destinatie particulara a unui mesaj.
4. Detaliu: ofera eroarea sepecifica aplicatiei si informatia de statis legata de blocul Body

   Exemplu:

```
<SOAP-ENV:Fault>
    <faultcode>Client</faultcode>
    <faultstring>Invalid Request</faultstring>
    <faultactor>http://jws.wiley.com/GetCatalog
      </faultactor>
</SOAP-ENV:Fault>
```

Codare SOAP

- Specificatiile SOAP 1.1 afirma ca aplicatiile bazate pe SOAPpot reprezenta datele lor fie ca literal sau valori codate.
  - Literal se refera la continut de mesaj codat conform cu XML Schema.
  - Valoarea codata se refera la mesaje codate bazandu-se pe stilul de codare SOAP.
- Identificatorii din spatiul de nume pentru aceste stiluri de codare SOAP sunt definite in
  - http://schemas.xmlsoap.org/soap/encoding/ (SOAP 1.1) si
  - http://www.w3.org/2001/06/soap-encoding (SOAP 1.2).
- Codarea SOAP defineste multime de reguli pentru explimarea tipuriloe sale de date
- Este o multime de tipuri de date care sunt reprezentate in limbajele de programare, baza de date si date semi-structurate cerute pentru o aplicatie
- Codarea SOAP defineste de asemenea reguliele de serializare pentru modelul de date utilizand atributul encodingStyle sub saptiu de nume SOAP-ENV care specifica regulile de serializate pentru elemente specifice si pentru un grup de elemente
- Codarea SOAP suporta atat valori simple cat si complexe

Tipuri de date si structuri SOAP: Tipuri primitive

Tipul primitiv	**Descriere**	**Exemple**
xsd:int	Valoare intreaga cu semn	-9 sau 9
xsd:boolean	boolean acarui valoare este 1 sau 0	1 sau 0
xsd:string	sir de caractere	Rocky Mountains
xsd:float or xsd:double	Numar in virgula mobila cu semn (+,-)	-9.1 sau 9.1
xsd:timeInstant	data/timp	1969-05-07-08:15
SOAP-ENC:base64	Informatie codata in base64 utilizata pentru pasarea datelor binare in documente *SOAP*	SW89IjhhibdOI111QWgdGE

<int>98765</int> <decimal> 98675.43</decimal> <string> Java Rules </string>

Structuri - exemple

```
<CRM:AuthorInfo xmlns:CRM=
    "http://www.charlesriver.com/authorinfo">
<CRM:FirstName xsi:type="string">
   Brian</CRM:FirstName>
 <CRM:LastName xsi:type="string">
   Hochgurtel</CRM:LastName>
 <CRM:PhoneNumber xsi:type="int">
    3035551212</CRM:PhoneNumber>
 <CRM:BookTitle xsi:type="string">
   Cross Platform Web Services
    </CRM:BookTitle>
</CRM:AuthorInfo>
***
<stockquote:symbolist
     xmlns:stockquote=
     "http://advocatemedia.com/stocks">
     <stockquote:symbol>C
        </stockquote:symbol>
     <stockquote:symbol>GE
        </stockquote:symbol>
     <stockquote:symbol>DJI
        </stockquote:symbol>
 </stockquote:symbolist>
```

```
<xs:element name=”ShippingAddress”
     xmlns:xs=”http://www.w3.org/2001/XMLSc
     hema”>
<xs:complexType>
<xs:sequence>
    <xs:element ref=”Street”type=”xsd:string”/>
    <xs:element ref=”City” type=”xsd:string”/>
    <xs:element ref=”State” type=”xsd:string”/>
    <xs:element ref=”Zip” type=”xsd:string”/>
    <xs:element ref=”Country”
     type=”xsd:string”/>
  </xs:sequence>
  </xs:complexType>
</xs:element>
    ***
```

XML instance:

```
<e:ShippingAddress>
     <Street>1 Network Drive</Street>
     <City>Burlington</City>
     <State>MA</State>
     <Zip>01803</Zip>
     <Country>USA</Country>
</e:ShippingAddress>
```

Enumerare

- Defineste o multime de nume specifice pentru un tip de baza
- Exemplu explimat in W3C XML Schema:

```
<xs:schema xmlns:xs=”http://www.w3.org/2001/XMLSchema”
        elementFormDefault=”qualified”>
    <xs:element name=”ProductType”>
        <xs:simpleType base=”xsd:string”>
          <xs:enumeration value=”Hardware”>
          <xs:enumeration value=”Software”>
        </xs:simpleType>
    </xs:element>
</xs:schema>
```

Arrays - exemple

```
<SymbolList
  SOAP-ENC: arrayType= "xsd:string[3]">
     <symbol>C</symbol>       <symbol>GE</symbol>
     <symbol>DJI</symbol>
</SymbolList>
                          ****
 <?xml version='1.0' ?>
  <env:Envelope       xmlns:env=
     "http://www.w3.org/2001/12/SOAP-envelope"
     xmlns:xsd="http://www.w3.org/1999/XMLSchema"
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-
     instance"
     xmlns:SOAP-ENC=
     "http://schemas.xmlSOAP.org/SOAP/encoding/"
     xmlns:stockquote="http://advocatemedia.com/exa
     mples">
  <env:Body>
<SOAP-ENC:Array SOAP-ENC:
     arrayType="xsd:string[3]">
     <stockquote:symbol>
        C</stockquote:symbol>
     <stockquote:symbol>
        GE</stockquote:symbol>
     <stockquote:symbol>
          DJI</stockquote:symbol>
 </SOAP-ENC:Array>
 </env:Body>  </env:Envelope>
```

```
<AuthorInfo
  SOAP-ENC:arrayType="xsd:ur-type[4]" >
     <FirstName xsi:type="string">
       Brian</FirstName>
     <LastName        xsi:type="string">
       Hochgurtel</LastName>
     <PhoneNumber        xsi:type="int">
       3035551212</PhoneNumber>
     <BookTitle xsi:type="string">
     Cross Platform Web Services
     </BookTitle>
<AuthorInfo>
```

xsd:ur-type[4], indica c sunt pentru elemente intr-o matrice cu tipuri variate

Exemplu de matrice de date binare

- este reprezentata ca text utilizand algoritmii base64 si exprimata utilizand o XML Schema

```
<myfigure xmlns:xsi=
        ”http://www.w3.org/2001/XMLSchema-instance”
  xmlns:enc=
        ” http://schemas.xmlsoap.org/soap/encoding”>
  xsi:type=”enc:base64”>
  sD334G5vDy9898r32323
</myfigure>
```

Matrice transmise partial

- este definit utilizand SOAP-ENC:offset, care permite sa fie indicata pozitia offset de la primul element (contorizat ca origine zero), care este utilizat ca offset pentru toate elementele care sunt transmise

Urmatoarea lista este o matrice de dimensiune [6];

utilizand SOAP-ENC:offset=”4” transmite elementele 5 si 6 ale unei matrice date c numerele (0,1,2,3,4,5).

```
<SOAP-ENC:Array SOAP-ENC:arrayType=”xsd:string[6]”
   SOAP-ENC:offset=”[2]”>
    <item> No: 2</item>
    <item> No: 3</item>
    <item> No: 4</item>
    <item> No: 5</item>
</SOAP-ENC:Array>
```

Matrice rare

- Sunt definite folosind SOAP-ENC:position, care primite ca pozitia unui atribut sa fie indicata intr-o matrice si sa fie returnata valoarea sa in locul fisarii fiecari intrari din matrice

Un exemplu de utilizare a unei matrice rare:

```
<SOAP-ENC:Array
  SOAP-ENC:arrayType=”xsd:int[10]”>
    <SOAP-ENC:int SOAP-ENC:position=”[0]”>
      0</SOAP-ENC:int>
    <SOAP-ENC:int SOAP-ENC:position=”[10]”>
      9</SOAP-ENC:int>
</SOAP-ENC:Array>
```

Comunicare SOAP

- SOAP suporta doua tipuri de modele de comunicare:

1. SOAP RPC.
   - Defineste o comunicare sincrona bazata pe apel de procedura la distanta in care nodurile SOAP expediza si receptioneaza mesaje ustilizabd metode de cereree si raspuns si schimba parametrii si poi returneaza valorile
2. SOAP Messaging.
   - Defineste o comunicare condusa de document in care nodurile SOAP expediaza si receptioneaza documente bazate pe XML utilizand mesaje sicrone si asincrone

SOAP RPC

- Defineste un model de comunicare strans cuplat bazat pe cereri si raspunsuri
- Mesajul SOAP message este reprezentat de nume de metode cu zero sau mai multi parametrii si valor de returnare
- Fiecare mesaj de cerere SOAP reprezinta un aoel de metoda la obiectul la distanta dintr-un server SOAP
- Apelurile metodei sunt serializate in tipuri de date bazate pe XML prin reguli de codare SOAP

SOAP Messaging

- Reprezinta un model de comunicare slab cuplat bazat pe notificari si schimbare a documentelor XML
- Corpul mesajului SOAP este reprezentat de documente XML sau literali codati conform unei scheme XML specifice,si sunt produsi si consumati de nodurile SOAP expeditoare si receptoare.
- Nodul SOAP expeditor trimite un mesaj cu un document XML ca si mesaj corp si nodul SOAP receptor care il proceseaza.

Lagaturi de transport

- In plus la XML intr-o cerere *SOAP*, exista si un antet in afara XMLului care este specific pentru protocolul care este utilizat, precum HTTP.
  - Informatia din acest antet contine codul respuns, versiunea utilizata de protocol, tipul continutului mesajului, si posibil informtii specifice vendorilor
- Specificatiile SOAP nu specifica sau mandateaza orice protocol pentru comunicare deoarece toate alege sa se lege la o varietate de protocoale de transport intre nodurile SOAP.
- Conform specificatiilor SOAP pentru legarea cadrului de lucru, legaturile SOAP definesc cerintele pentru mesajele de cerere si raspuns utilizand un protocol ed transport intre nodurile SOAP.
  - Legaturile acestea definesc de asemenea regulile sintactice si semantice pentru procesarea mesajelor SOAP sosite/plecate si suporta o multime de sabloane de schimburi de mesaje
  - Aceasta permite ca SOAP sa fie utilizat intr-o varietate de aplicatii si platforme utilizand o varietate de protocoluri

Preferinte pentru legaturi de transport

- Desi SOAP poate fi utilizat potential peste o varietate de protocoale de transport, initial specificatia SOAP 1.0 mandateaza utilizarea HTTP ca protocolul sau de transport
- Specificatiile ulterioare au deschis suportul pentru alte protocoluri bazate pe Internet precum SMTP si FTP.
- Mai tarziu, majoritatea vendorilor SOAP u realizat implemnetari folosind si alet protocoluri de transport precum POP3, BEEP, JMS, Custom Message-Oriented-Middleware, si protocoluri proprietare utilizand socluri TCP/IP.
- SOAP utilizeaza aceste legaturi de protocol ca mecanism pentru a transporta URI al nodurilor SOAP.
  - Tipic intr-o cerere HTTP, URIul indica endpointul a unei resurse SOAP in care invocarea este realizata

SOAP peste HTTP

- Utilizarea HTTP ca protocol de transport pentru comunicare SOAP devine o potrivire naturala pentru SOAP/RPC.
- Utilizand SOAP peste HTTP nu necesita suprascrierea oricaror reguli sintactice si semantice a HTTPului, dar mapeza sintaxa si semnatica HTTPului
- Adoptand SOAP peste HTTP, mesajele SOAP pot fi expediate prin portul 80 HTTP fara a necesita deschiderea altor porturi
- Singura constrangere cnd este utilizat SOAP peste HTTP este cerinta de a utiliza un antet special pentru definirea tipului MIME ca si Content-Type: text/xml.

SOAP peste SMTP

- SOAP peste SMTP permite mesajelor SOAP sa fie transmise prin comunicare asincrona si sa suporte notificari intr-un singur sens precum si cerinte de transmitere conduse de document
- Ajut transmiterea SOAP cand nu exista o potrivire buna intre cerere si raspuns si cnd semantica HTTP nu se aplica in mod natural.
- Pentru a lucra cu SMTP,
  - Un document *SOAP* trebuie sa inlocuiasca antetul HTTP cu info necesara pt. e-mail.
  - In locul informatiei necesare pentru HTTP, precum Post si URLul serviciului, antetul SMTP *SOAP* contine o adresa e-mail, un subiect si o data
  - O cerere *SOAP* poate de asemenea sa contina Iduri unice de mesaj.
  - In locul mesajului text, aplicatia expediaza documentul *SOAP* care contine XMLul necesar pentru serviciu.
  - Mesajul SOAP va trimite cererea cu un ID de mesj in antet si mesajul de raspuns SOAP va contine un antet In-Reply-Tocontinand Message-Id initial.
- cu e-mail si *Standard Mail Transport Protocol* (SMTP) mesajul poate parcurge un timp pana a ajunge la receptor deoarece e-mail este impartit in mai multe bucati care sunt trimise prin Internet.
  - HTTP, pe de alta parte, urmareste un raspuns imediat intr-o perioda de timp imediata

Alte legaturi SOAP

- **SOAP peste HTTP/SSL.**
  - Avantaj de utilizare Secure Socket Layer (SSL) pentru securitate si alte protocoluri bazate pe HTTP
  - Posibila adaugarea adresei MAC (Media access control) in mesaje.
- **SOAP peste JMS.**
  - Pentru a permite mesaje SOAP pentru a comunica cu componente J2EE
  - Majoritatea vendorilor SOAP ofera transmitere SOAP peste JMS (Java Messaging Service): Sun One MQ, Sonic MQ, Websphere MQSeries, etc.
  - Permite transmitere asincrona bazata pe SOAP
  - Cozile destinatie JMS sunt reprezentate in mesaje SOAP ca destintii tinta.
  - Nodurile SOAP utlizeaza coada JMS pentru expedierea si receptionarea cererilor si raspunsurilor SOAP
- **SOAP peste BEEP.**
  - Blocks Extensible Exchange Protocol (BEEP) defineste un cadru de lucru si protocol de transport generic pentru transmitere asincrona bazata pe conexiune care permite transmitere peer-to-peer, client-server, sau server-to-server.
  - Permite dezvoltatorilor SOAP sa se concentreze la aspecte ale aplicatiilor SOAP in locul gasirii unei modalitato de stabilire a comunicatiei : BEEP se ocup de protocolul de comunicare.

Securitate SOAP

- Mesajele SOAP nu poarta sau definesc nici un mecanism specific de securitate
- Utilizarea antetelor SOAP este o modalitate de defini si aduaga facilitai care permit implementarea unei securitati specificie aplicatiei in forma de metadate bazate pe XML.
  - Informatia metadata poate fi informatie specific aplicatiei incorporad securitatea mesajelor cu algoritmi de securitate asociati precum criptarea si semnaturile digitale
- SOAP suporta protocoluri variate de transport pentru comunicare, astfel este de asemenea posibil sa incorporeze si mecanisme de securitate a transportului precum SSL/TLS pentru mesaje SOAP.
- Toate elementele legate de securitatesunt identificate utilizand prefixul SOAP-SEC si URIul asociat http://schemas.xmlsoap.org/soap/security/.
  - Defineste trei etichete element de securitate <SOAP-SEC: Encryption>, <SOAP-SEC:Signature>, si <SOAP-SEC:Authorization>.
    - Utilizarea acestor etichete permite incorporarea criptarii, semnaturilor digitale si autorizarea in mesaje SOAP.

UDDI

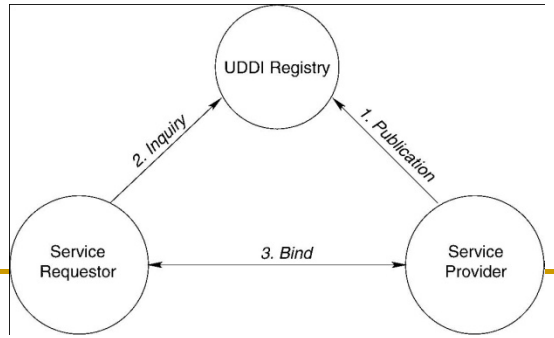
Universal Discovery, Description & Integration

- Defineste o modalitate standar pentru inregistrarea, de-inregistrarea si gasire unui WS.
- Anuntat in 2000 ca un punct comun Microsoft, IBM, si Ariba.
- Specificatia este disponibila la http://www.uddi.org
- Directoarele UDDI nu sunt limitate numai la WS, si pot contine servicii bazate pe un numar de protocoluri si tehnologii precum telefonie, FTP, email, CORBA, SOAP,Java RMI.
- Exista doau parti principale la UDDI:
  - Specificatie pentru mentinerea informatiei
  - Implementarea specificatiei

Service Lookup prin UDDI

- Service lookup: ofera un director care ofera publicitate serviciilor
  - UDDI este solutia WS la aceasta problema.
- In general, un ciclu de negociere consta in urmatorii pasi:

1. Un furnizor ofera un serviciu si doresye sa le faca publicitate catre clienti pentru a le utiliza
   - Furnizorul inregistreaza serviciul intr-un registru UDDI.
   - Cererea de publicare include informatie relativa la serviciul oferit, precum specificatia WSDL
2. Un solicitant de serviciu cauta o anumita functionalitate
   - Chestioneaza registrul UDDI, specificand ceea ce cauta.
   - Cand este gasita o protrivire, registrul UDDI raspunde cu informatia referitoare la un furnizor de servicii adecvat
3. Odata ce soliciatntul serviciului cunoaste detaliile furnizorului de servicii, se poate lega la furnizor. Din acest moment, solicitantul interactioneaza cu furnizorul



Ce ofera UDDI

- UDDI defineste un model de informare care descrie datele mentinute in registru
- Conceptual acest model contine:
  - Informatia de afaceri despre entitatea care ofera serviciul,
  - Tipul de serviciu care este oferit
  - Detalii cum poate fi invocat serviciu
- Intrarile stocate in registrul UDDI sunt clasificate conform tipurilor.
  - UDDI utilieaza mai multe scheme de categorisire, precum North American Industry Classification System (NAICS).
  - Categorisirea organizeaza serviciile intr-o ierarhie si faciliteaza descoperirea lor
- Un registru conform cu UDDI trebuie sa inteleaga numeroase mesaje SOAP prin care clientii pot interactiona cu registrul
  - Interfata SOAP este utilizata pentru crearea, actualizarea si interogarea intrarilor din registru
  - Serviciile Web utilizeaza propriile standarde precum WSDL pentru a descrie interfeta intr-un registru UDDI.

Operator UDDI

- Ofera un registru pentru utilizare publica generala
- Trebuie sa ofere o interfata la registrul sau
- Poate oferi service extra catre clientii sai
  - Specificatia UDDI indica faptul ca registrul nucleu descris de modelul de informare este o replica excata a altor registrii ai operatorului
  - E.g. ofera o interfata bazata pe Webpentru registrii sai care faciliteaza descoperirea serviciilor in momentul conceperii

Standarde pentru registrii

- UDDI este o specificatie bazata pe standarde pentru inregistrarea serviciilor Web, descriere, si descoperire.
- Similar de pagini galbene ale unui sistem de telefonie scopul unui registru UDDI este de a permite furnizorilor sa-si inregistreze serviciile si solicitantilor sa geseasca serviciile.
  - Odata ce un solicitant a gast un serviciu, registru nu mai joaca un rol intre solicitant si furnizor
- Specificatia UDDI defineste
  - O schema XML pentru mesajele SOAP
  - APIuri pentru aplicatiile care doresc sa utilizeze registrul.
    - Un furnizor ce inregistreaza un WS la un UDDI trebuie sa furnizeze afacerea, serviciul, legatura si informatia tehnica despre serviciu

Informatia stocata consista din trei parti

- Pagini albe
  - Descrie informatia de afaceri generala precum nume, descriere, numere de telefon, etc
- Pagini galbene
  - Descrie afacerea in termeni de taxonomii standard
  - Aceasta informatie trebuie sa urmeze categorisiri industriale standard asfel incat serviciile sa fie localizate dupa industrie, categorie sau locatie geografica.
- Pagini verzi
  - Listeaza serviciile, legaturile si informatiile tehnice specifice

Comentarii la cele 3 seturi de date

- Prim este de informatia general de contact care include de ex. Adresa strazii si un individ responsabil pentru completarea cerintelor
  - Cautarea pentru tipul de serviciu, precum valori de stocuri, este de asemenea disponinila cu informatii despre tip afacerii pe care se bazeaza o companie particulara
  - Gandind UDDI ca o combinatie de pagini albe si pagini galbene pentru servicii Web Services si afaceri se poate obtine o imagine clara cum se poate exploata UDDI.
- Legaturile la fisierele WSDL sunt adesea gasite pe site-ul UDDI, dar ele nu partajeaza nici o relatie formala
  - UDDI spune unde rezida serviciul Web Service si cine-l sponsorizeaza
  - Cand se incearca descoperirea de servicii Web, se gaseste adesea un site UDDI care conduce la fisierul WSDL

APIuri

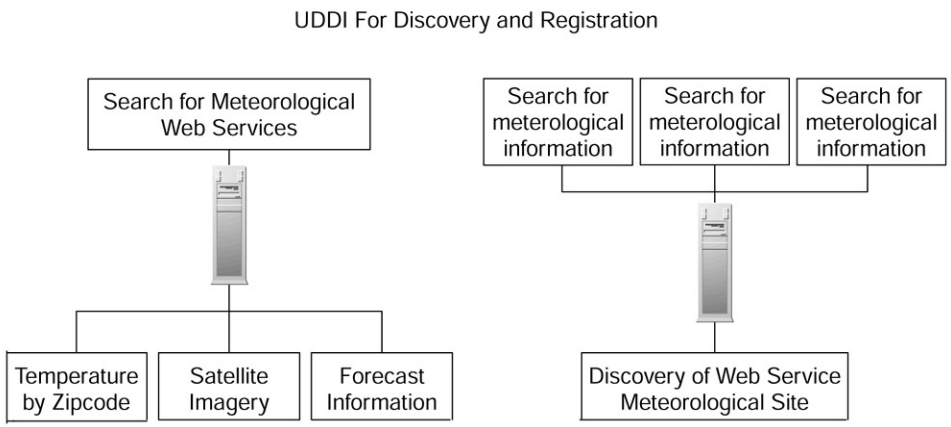
- Un registru poate fi contactat in doua modalitati:
  - Fie printr-un navigator Web sau
  - Prin cereri si raspunsuri *SOAP* utilizand un API particular.
- Specificatia UDDI include doua categorii de APIuri pentru accesaea serviciilor UDDI din aplicatii:

1. APIuri de interogare— permit cautarea si navigarea prin informatia din registru
2. APIurile de inregistrare— permit aplicatiilor sa inregistreze serviciile intr-un registru

- APIurile UDDI se comporta intr-o maniera sincrona.
- UDDI utilizeaza SOAP ca protocol de baza.
- UDDI este o specificatie pentru un registru, nu un repositoriu
  - Ca un registru functioneaza ca un catalog, permitand solicitantilor sa gaseasca serviciile disponibile
  - Un registru nu este un repositoriu deoarece nu contine serviciile in sine

Site-uri Web UDDI

- Site-urile Web UDDI sunt sustinute de un grup de leaderi industriali precum HP, IBM, Microsoft, si SAP, care formeaza un consortiu pentru a mentine *UDDI registry system*.
- Acest sistem este o baza de date pentru afaceri si URLurile serviciilor Web pe care pe ofera, orice fisiere WSDL disponibile, si informatia de afaceri.
- Fiecare dintre vendori replica intrarile pe care le primeste de la ceilalti
- Figura: replicarea intre diferite siteuri UDDI.



URLuri ale siteurilor Web

- Implementari UDDI versiunea 1:
  - Microsoft http://uddi.microsoft.com/
  - IBM http://www-3.ibm.com/services/uddi/find
- Implementari UDDI versiune 2
  - Microsoft https://uddi.rte.microsoft.com/search/frames.aspx
  - IBM https://www3.ibm.com/services/uddi/v2beta/protect/registry.html
  - Hewlett Packard https://uddi.hp.com/uddi/index.jsp
  - SAP https://websmp201.sap-ag.de/~form/uddi_discover
- Principala diferenta intre versiunea 1si 2 este GUI si securitate imbunatatite.

Studiu de caz UDDI

- Implementarea UDDI implica instalarea unui site UDDI internal la o organizatie mare, fie de telecom.
  - O asemenea organizatie poate avea sute de WSuri interne si externe pentru audienta larga
  - Utilizand UDDI, coproratiile departamentelor de IT au avantaje familiare:
    - Un nou dezvoltator din organizatie cunoaste aproape imediatunde se gaseste informatia despre serviciile disponibile in organizatie
    - Daca registrul este actualizat, poate salva financiar departamentul ITprin reducerea costurilor de training si documentare
- Eg. fie o sarcina de a aduce un sistem sau date proprietar pe un site public Web.
  - In loc de a vana administratorul, gasi documente, si crearea propriei documentatii, se foloseste doar un site intern UDDI.
  - Se cauta un tip de WS, se conecteaza pagina Web la acest WS, si apoi se lanseaza noua aplicatie Web.