# Service Oriented Architectures

## Introduzione ai Web Services

Fulvio Corno

Dipartimento di Automatica e Informatica

Politecnico di Torino

# Il web per le macchine

- Il concetto di navigazione web si basa su tre “servizi primitivi”, definiti nel protocollo http: GET, POST, PUT
  - Adatti a scambiare pagine HTML con un browser
  - Tecnologia semplicissima e diffusissima
- Perché non usare una tecnologia simile per risolvere problemi di integrazione tra sistemi informativi aziendali?

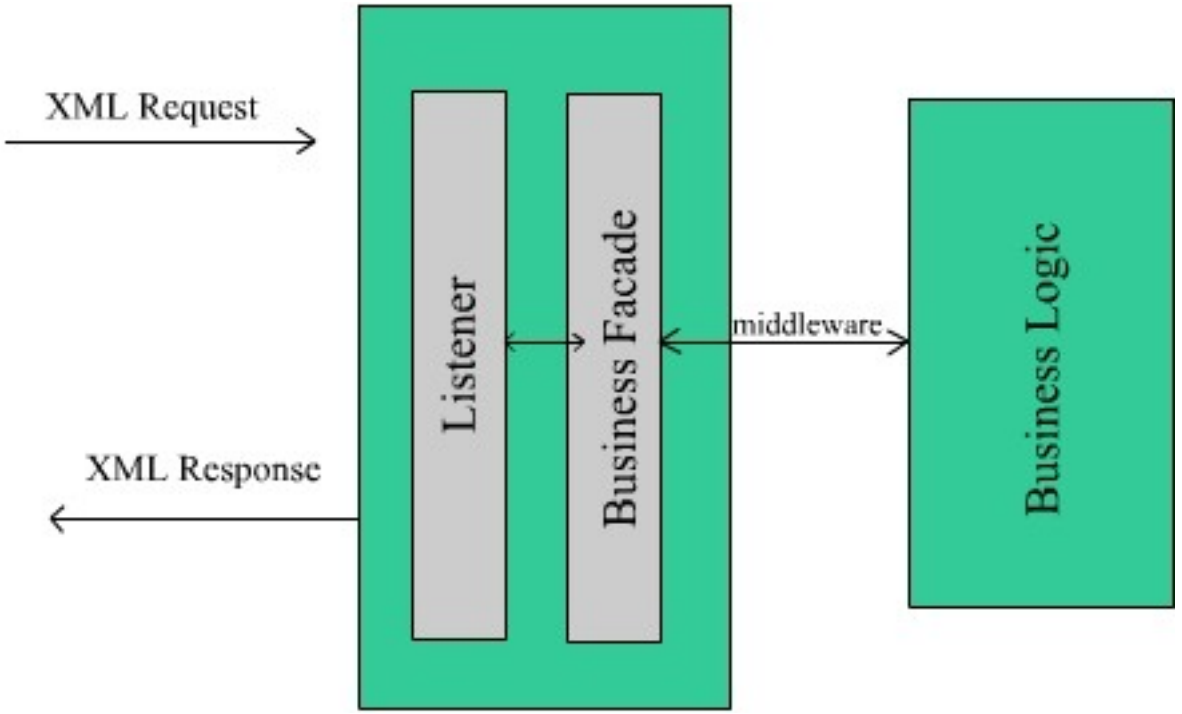
# Integrazione di sistemi informativi

- Parte dell'informazione necessaria ad un sistema informativo "A" è in realtà gestita da un sistema informativo "B"
  - A richiede a B: get_info_latest_news
- Parte delle operazioni necessarie ad un sistema informativo "A" devono in realtà essere eseguite da un sistema informativo "B"
  - A ordina a B: store_new_customer
  - A ordina a B: execute_creditcard_transfer
- Parte delle informazioni su cui il sistema informativo "A" si deve basare vengono aggiornate dal sistema informativo "B"
  - B informa A: update_product_list

# Quale tecnologia usare?

- Diverse tecnologie disponibili per integrare tra loro applicazioni diverse
  - Piattaforme di Middleware
    - RMI, Jini, CORBA, DCOM, ...
  - Spesso specifiche ad una tecnologia di application server oppure ad un linguaggio di programmazione
- Separariamo le funzioni:
  - Invocazione della funzione
    - Tecnologie e linguaggi web-based
      - XML, XMLSchema, http
  - Implementazione della funzione
    - A piacere...

# Invocazione + Implementazione



# Web Services

- Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes...Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service. [fonte: IBM]

# Sotto sotto...

- Un web service non è altro che un web server (un server http)
- La richiesta HTTP in realtà contiene una “richiesta XML” che specifica una funzione richiesta al sistema informativo sottostante
- La risposta HTTP conterrà una “risposta XML” contenente i dati o le informazioni richieste
- Il sistema informativo può adottare qualsiasi tecnologia, purché sia in grado di
  - analizzare e comprendere “richieste XML”
  - rispondere generando “risposte XML”

# ...ma non solo!

- Oltre al meccanismo di base per la chiamata (XML +HTTP) servono altri standard per fornire una "piattaforma" completa:
  - discovery, transactions, security, authentication, ...
- I Web Service usano:
  - SOAP (remote invocation)
  - WSDL (expression of service characteristics)
  - UDDI (trader, directory service)
  - ed altri, meno diffusi o non standard
    - XLANG/XAML (transactional support for complex web transactions involving multiple web services)
    - XKMS (XML Key Management Specification)

# SOAP

- Era acronimo di Simple Object Access Protocol
  - ora SOAP è diventato nome proprio (non più acronimo)
- Protocollo per lo scambio di **messaggi** su Internet
- Standard:
  - SOAP Version 1.2 Part 1: Messaging Framework
    - Struttura dei messaggi scambiati
    - Responsabilità dei partner nello scambio messaggi
    - Relazione con i protocolli di trasporto
  - SOAP Version 1.2 Part 2: Adjuncts
    - Modello dati
    - Realizzazione di RPC (Remote Procedure Call)
    - Relazione con il protocollo http
  - http://www.w3.org/2000/xp/Group/

# Caratteristiche di SOAP

- SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information.
  - stateless
  - one-way
  - complex interaction patterns
    - request/response

# Caratteristiche di SOAP

- SOAP is **silent** on the semantics of any application-specific data it conveys, as it is on issues such as the routing of SOAP messages, reliable data transfer, firewall traversal, etc.
  - No understanding of transported data
  - No understanding of transport mechanisms
- However, SOAP provides the framework by which **application-specific information may be conveyed in an extensible manner**.
- Also, SOAP provides a full description of the required **actions** taken by a **SOAP node** on receiving a **SOAP message**.

# Definitions

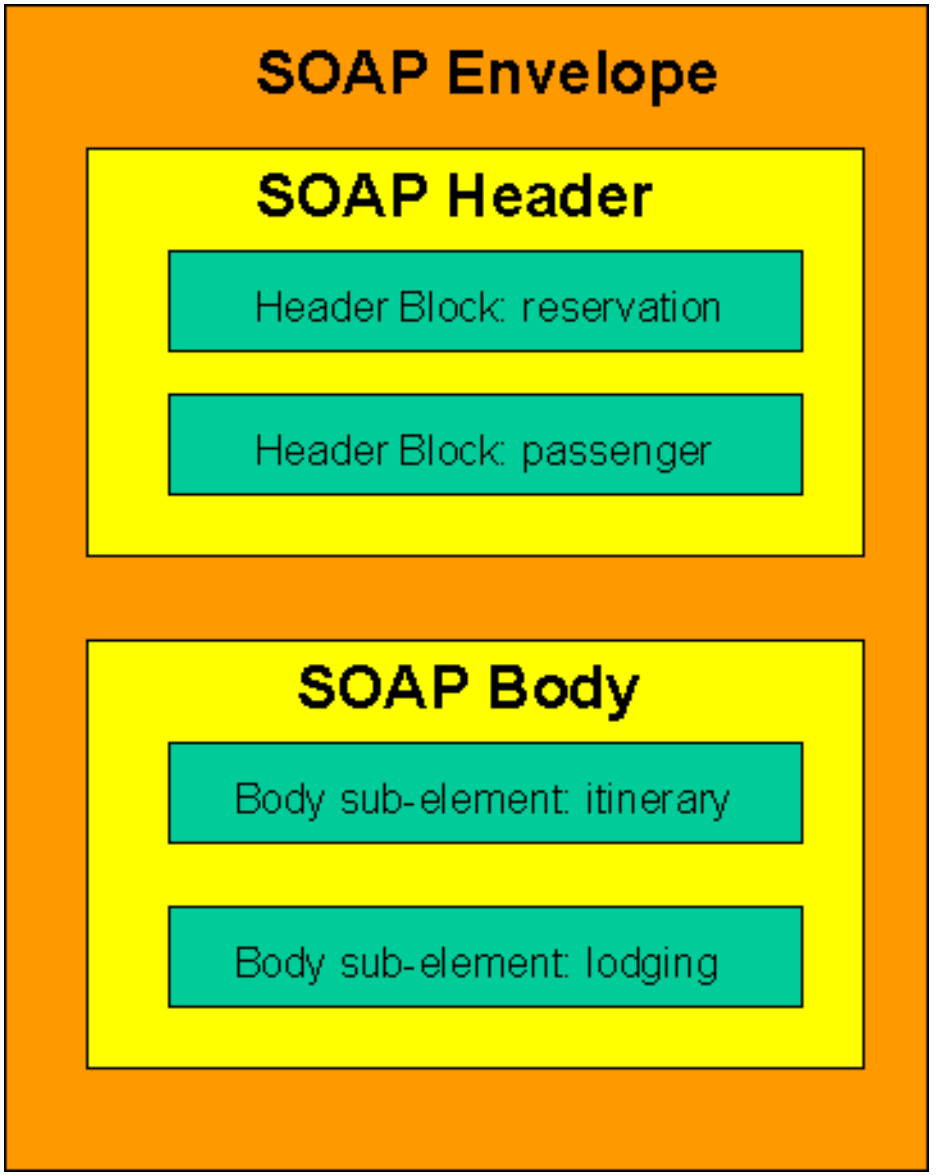
- SOAP node
  - The embodiment of the processing logic necessary to transmit, receive, process and/or relay a SOAP message, according to the set of conventions defined by this recommendation. A SOAP node is responsible for enforcing the rules that govern the exchange of SOAP messages (see 2. SOAP Processing Model). It accesses the services provided by the underlying protocols through one or more SOAP bindings.
- SOAP sender
  - A SOAP node that transmits a SOAP message.
- SOAP receiver
  - A SOAP node that accepts a SOAP message.
- SOAP message
  - The basic unit of communication between SOAP nodes.

# SOAP Paths

- Un messaggio SOAP viene
  - Inviato da un “Initial sender”
  - Ricevuto da un “Ultimate receiver”
  - Può essere intercettato, elaborato e ritrasmesso da zero o più “intermediari”
  - Ciascun intermediario riceve il messaggio, lo elabora, e lo ri-trasmette al successivo intermediario (o all’ultimate receiver)
- Complessivamente si parla di “percorso” (path) compiuto da un messaggio SOAP

# SOAP Message

- Messaggio = Envelope, che contiene Header + Body
  - Header = informazioni sul messaggio (data, autenticazione, ...)
  - Body = effettivi dati scambiati



# Richiesta e Risposta

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>827635</productID>
    </getProductDetails>
  </soap:Body>
</soap:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productName>Toptimate 3-Piece Set</productName>
        <productID>827635</productID>
        <description>3-Piece luggage set. Black Polyester.</description>
        <price>96.50</price>
        <inStock>true</inStock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```

# Example

- Travel reservation application
- The travel reservation application for an employee of a company negotiates a travel reservation with a travel booking service for a planned trip.
- The information exchanged between the travel reservation application and the travel service application is in the form of SOAP messages.

- Source: http://www.w3.org/TR/soap12-part0/

# Namespace

- env = http://www.w3.org/2003/05/soap-envelope
- enc = http://www.w3.org/2003/05/soap-encoding
- rpc = http://www.w3.org/2003/05/soap-rpc
- rep = http://www.w3.org/2004/08/representation
- xop = http://www.w3.org/2004/08/xop/include
- xmime = http://www.w3.org/2004/11/xmlmime
- xs = http://www.w3.org/2001/XMLSchema
- xsi = http://www.w3.org/2001/XMLSchema-instance

# Travel reservation: Request

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
   xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

# Travel reservation: Request Header

- SOAP Header (optional)
- Provides a way to pass information that is not application payload.
- E.g., directives or contextual information related to the processing of the message.
- Children of env:Header are called header blocks, and can be targeted at SOAP nodes encountered in the path of a message

```
 <env:Header>
  <m:reservation
xmlns:m="http://travelcompany.example.org
/reservation"

env:role="http://www.w3.org/2003/05/soap-
envelope/role/next"
            env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-
ba5d-pqff98fe8j7d</m:reference>

<m:dateAndTime>2001-11-29T13:20:00.000-05
:00</m:dateAndTime>
  </m:reservation>
  <n:passenger
xmlns:n="http://mycompany.example.com/emp
loyees"

env:role="http://www.w3.org/2003/05/soap-
envelope/role/next"
            env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
```

# Request header

- Header blocks
  - reservation : provides a reference and time stamp for this instance of a reservation
  - passenger : traveller's identity
- Must be processed by the next SOAP intermediary encountered in the message path or, if there is no intermediary, by the ultimate recipient of the message
  - env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
  - env:mustUnderstand="true"
- Such header block specifications are application defined and not a part of SOAP.

# Travel reservation: Request Body

- SOAP Body (mandatory)
-

```
<env:Body>
 <p:itinerary
xmlns:p="http://travelcompany.example.org/rese
rvation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
 <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late
afternoon</p:departureTime>
   <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
 <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-
morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
xmlns:q="http://travelcompany.example.org/rese
rvation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
```

# Request body

- env:Body and its contents are implicitly targeted and are expected to be understood by the ultimate receiver
- Child elements, are intended for exchange of information between the initial SOAP sender and the SOAP node which assumes the role of the ultimate SOAP receiver
- Ultimate receiver: travel service application
  - itinerary
  - lodging

# Design decisions

- What data is placed in a header block and what goes in the SOAP body are decisions made at the time of application design.
- The main point: header blocks may be targeted at various nodes encountered along a message's path from a sender to the ultimate recipient.
  - Such intermediate SOAP nodes may provide value-added services based on data in such headers.
- The passenger data is placed in a header block to illustrate the use of this data at a SOAP intermediary to do some additional processing. The outbound message is altered by a SOAP intermediary by having the travel policies pertaining to this passenger appended to the message as another header block.

# SOAP Message Exchange

- Uno scenario completo richiede più messaggi scambiati tra il [initial] sender ed il [ultimate] receiver
- Solitamente si hanno pattern Request-Response
  - Conversational message exchange
    - SOAP messages to form a back-and-forth "conversation", where the semantics are at the level of the sending and receiving applications
  - Remote Procedure Call (RPC)
    - when there is a need to model a certain programmatic behavior, with the exchanged messages conforming to a pre-defined description of the remote call and its return

# Esempio (conversational)

- In questo caso il travel service risponde richiedendo maggiori informazioni (itineraryClarification)
- Non è “la risposta” alla richiesta iniziale, ma un “frammento di conversazione” che porterà (sperabilmente) alla risposta

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelop
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/rese
      env:role="http://www.w3.org/2003/05/soap-envelope/role/ne
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:re
   <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/ne
           env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itineraryClarification
    xmlns:p="http://travelcompany.example.org/reservation/trave
   <p:departure>
     <p:departing>
       <p:airportChoices>
          JFK LGA EWR
       </p:airportChoices>
     </p:departing>
   </p:departure>
   <p:return>
     <p:arriving>
       <p:airportChoices>
         JFK LGA EWR
       </p:airportChoices>
     </p:arriving>
   </p:return>
  </p:itineraryClarification>
 </env:Body>
</env:Envelope>
```

# Esempio (conversational)

- Il sender iniziale a questo punto può integrare le informazioni richieste
- Fornisce un itinerary parziale che contiene gli aeroporti
- L'header contiene tutti i riferimenti (m:reference) per permettere al receiver di correlare questi dati alla richiesta iniziale

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
 <env:Header>
  <m:reservation
     xmlns:m="http://travelcompany.example.org/reservati
      env:role="http://www.w3.org/2003/05/soap-envelope/
           env:mustUnderstand="true">
    <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j
    <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:date
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/emp
      env:role="http://www.w3.org/2003/05/soap-envelope/
           env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary
   xmlns:p="http://travelcompany.example.org/reservation
   <p:departure>
     <p:departing>LGA</p:departing>
   </p:departure>
   <p:return>
     <p:arriving>EWR</p:arriving>
   </p:return>
  </p:itinerary>
 </env:Body>
</env:Envelope>
```

# SOAP RPC

- Information needed for a procedure call:
  - The address of the target SOAP node.
  - The procedure or method name.
  - The identities and values of any arguments to be passed to the procedure or method together with any output parameters and return value.
  - A clear separation of the arguments used to identify the Web resource (target for the RPC), as contrasted with those that convey data or control information used for processing the call by the target resource.
  - The message exchange pattern which will be employed to convey the RPC, together with an identification of the so-called "Web Method" to be used.
  - Optionally, data which may be carried as a part of SOAP header blocks.

# Esempio (RPC)

- Esempio: payment for the trip using a credit card
  - The travel reservation application provides credit card information and the successful completion of the different activities results in the card being charged and a reservation code returned.
- chargeReservation( reservation, creditCard )
  - reservation: code
  - creditCard: name, number, expiration
- Calling one procedure with multiple parameters
- Each parameter may have a complex type

# RPC Request (Call)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
   <t:transaction
           xmlns:t="http://thirdparty.example.org/transaction"
           env:encodingStyle="http://example.com/encoding"
           env:mustUnderstand="true" >5</t:transaction>
 </env:Header>
 <env:Body>
  <m:chargeReservation
      env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
         xmlns:m="http://travelcompany.example.org/">
   <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
    <m:code>FT35ZBQ</m:code>
   </m:reservation>
   <o:creditCard xmlns:o="http://mycompany.example.com/financial">
    <n:name xmlns:n="http://mycompany.example.com/employees">
           Åke Jógvan Øyvind
    </n:name>
    <o:number>123456789099999</o:number>
    <o:expiration>2005-02</o:expiration>
   </o:creditCard>
  </m:chargeReservation>
 </env:Body>
</env:Envelope>
```

# RPC response

- Any RPC call must be responded
- The response is a SOAP body contaning one or more data blocks
- Example:
  - code: reference code for the reservation
  - viewAt: URL where the details of the reservation may be viewed
- By convention: the RPC response body child element name is the same as the RPC call, plus “Response”
  - chargeReservationResponse
- The Remote Procedure may also return a [simple] return value <rpc:result>

# RPC Response

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
     <t:transaction
        xmlns:t="http://thirdparty.example.org/transaction"
          env:encodingStyle="http://example.com/encoding"
           env:mustUnderstand="true">5</t:transaction>
 </env:Header>
 <env:Body>
     <m:chargeReservationResponse
         env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
             xmlns:m="http://travelcompany.example.org/">
       <m:code>FT35ZBQ</m:code>
       <m:viewAt>
         http://travelcompany.example.org/reservations?code=FT35ZBQ
       </m:viewAt>
     </m:chargeReservationResponse>
 </env:Body>
</env:Envelope>
```

# Return value

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
    <t:transaction
       xmlns:t="http://thirdparty.example.org/transaction"
         env:encodingStyle="http://example.com/encoding"
          env:mustUnderstand="true">5</t:transaction>
 </env:Header>
 <env:Body>
    <m:chargeReservationResponse
       env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
           xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
             xmlns:m="http://travelcompany.example.org/">
       <rpc:result>m:status</rpc:result>
       <m:status>confirmed</m:status>
       <m:code>FT35ZBQ</m:code>
       <m:viewAt>
        http://travelcompany.example.org/reservations?code=FT35ZBQ
       </m:viewAt>
    </m:chargeReservationResponse>
 </env:Body>
</env:Envelope>
```

# Gestione degli errori

- Per qualche motivo, una richiesta SOAP potrebbe non essere soddisfatta
- SOAP definisce un “formato standard” per restituire delle condizioni di errore:
  - env:Fault (deve essere l’unico blocco del Body)
    - env:Code
      - env:Value
      - env:SubCode (optional)
    - env:Reason
    - env:Detail (opzionale)

# Messaggio di errore

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:rpc='http://www.w3.org/2003/05/soap-rpc'>
  <env:Body>
   <env:Fault>
     <env:Code>
       <env:Value>env:Sender</env:Value>
       <env:Subcode>
        <env:Value>rpc:BadArguments</env:Value>
       </env:Subcode>
     </env:Code>
     <env:Reason>
      <env:Text xml:lang="en-US">Processing error</env:Text>
      <env:Text xml:lang="cs">Chyba zpracování</env:Text>
     </env:Reason>
     <env:Detail>
      <e:myFaultDetails
        xmlns:e="http://travelcompany.example.org/faults">
        <e:message>Name does not match card number</e:message>
        <e:errorcode>999</e:errorcode>
      </e:myFaultDetails>
     </env:Detail>
   </env:Fault>
 </env:Body>
</env:Envelope>
```

# SOAP protocol bindings

- SOAP definisce la struttura dei messaggi XML
- Deve esistere un meccanismo per “trasportare” il messaggio dal sender al receiver, eventualmente usando uno o più nodi intermediari
- SOAP non specifica un metodo di trasporto, ma ne può supportare diversi
- Il più utilizzato è il SOAP HTTP Binding
  - Il paradigma request-response è già realizzato da http
    - http request = SOAP request message
    - http response = SOAP response message
  - La URI della richiesta identifica in modo univoco il server e la funzione all’interno del server, ossia il destinatario SOAP

# SOAP su HTTP (application/soap+xml)

- SOAP request-response message exchange pattern
  - SOAP request
    - richiesta http POST
    - il messaggio SOAP è nel body della http request
  - SOAP response
    - risposta alla richiesta precedente (http response)
    - il messaggio SOAP è nel body della http response
- SOAP response message exchange pattern
  - richiesta non-SOAP (non serve un messaggio di richiesta)
    - normale richiesta GET (con eventuali parametri)
  - SOAP message
    - nel body della http response
  - usata per ottenere la rappresentazione di una risorsa senza alterarla in alcun modo

# Esempio (GET)

```
GET /travelcompany.example.org/reservations?code=FT35ZBQ  HTTP/1.1
Host: travelcompany.example.org
Accept: text/html;q=0.5, application/soap+xml
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
       env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-30T16:25:00.000-05:00</m:dateAndTime>
  </m:reservation>
 </env:Header>
 <env:Body>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:x="http://travelcompany.example.org/vocab#"
     env:encodingStyle="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <x:ReservationRequest
 rdf:about="http://travelcompany.example.org/reservations?code=FT35ZBQ">
      <x:passenger>Åke Jógvan Øyvind</x:passenger>
      <x:outbound>
       <x:TravelRequest>
```

# Esempio (POST)

```
POST /Reservations HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
   <t:transaction
           xmlns:t="http://thirdparty.example.org/transaction"
           env:encodingStyle="http://example.com/encoding"
           env:mustUnderstand="true" >5</t:transaction>
 </env:Header>
 <env:Body>
  <m:chargeReservati
     env:encodingSty
          xmlns:m="h
   <m:reservation xm
    <m:code>FT35ZBQ</
   </m:reservation>
    <o:creditCard xm
     <n:name xmlns:n=
           Åke Jógva
     </n:name>
     <o:number>12345
     <o:expiration>2
    </o:creditCard>
   </m:chargeReserva
 </env:Body>
</env:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
       ...
       ...
 </env:Header>
 <env:Body>
       ...
       ...
 </env:Body>
</env:Envelope>
```

# Problema

- Per poter invocare un web service devo conoscere
  - L'indirizzo URI (host + path) a cui inviare il messaggio SOAP
  - L'esatta struttura degli header block e body block che l'applicazione si aspetta
  - L'esatta struttura degli header block e body block che l'applicazione mi potrà restituire
- Come è possibile codificare tutte queste informazioni in modo flessibile, dichiarativo ed esterno al codice applicativo?
  - Risposta: WSDL

# WSDL

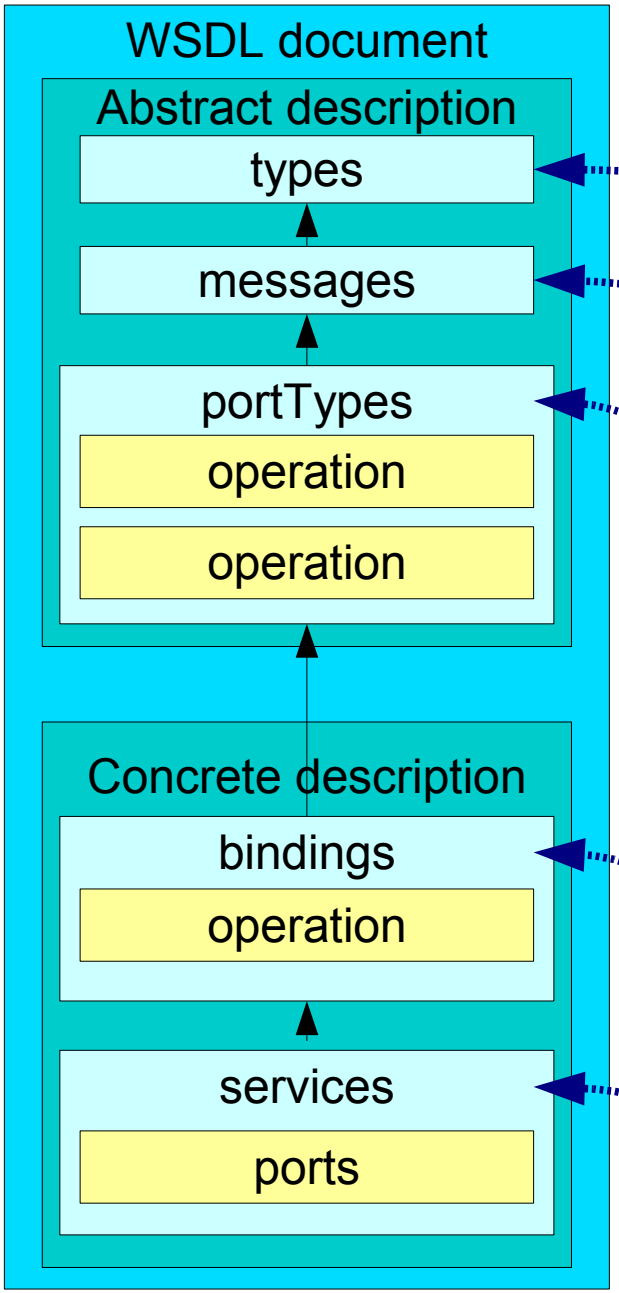
- Web Services Description Language
- Definisce le informazioni necessarie a comunicare con un determinato web service:
  - cosa il servizio può fare
  - dove si trova
  - come invocarlo
- La descrizione WSDL può essere memorizzata nella registry UDDI
- Definisce
  - Gli XMLSchema dei messaggi accettati dal web service
  - L'elenco dei comandi accettati nel Body
  - I tipi di dato dei parametri dei comandi
- http://www.w3.org/TR/wsdl

# Parti principali di un WSDL

- types
  - describes the kinds of messages that the service will send and receive.
- interface
  - describes what abstract functionality the Web service provides.
- binding
  - describes how to access the service.
- service
  - describes where to access the service.

```
<description>
  <documentation />?
  [ <import /> | <include /> ]*
  <types />?
  [ <interface /> | <binding /> | <service /> ]*
</description>
```

# Rappresentazione concettuale




Quali tipi di dato si usano?

Quali sono i messaggi che vengono scambiati nelle due direzioni?

Quali sono le funzioni supportate dal servizio?

Come si può richiamare il servizio sulla rete? Con quali protocolli? (es. dettagli sugli schemi SOAP)

Dove si trova il servizio?

# Esempio – Hotel reservation system (1)

- **CheckAvailability**.
  - The client must specify check-in date, check-out date, room type.
  - The Web service will return a room rate (a floating point number in USD) if a room is available, or zero room rate if not.
  - If input data is invalid, the service should return an error.
  - The service will accept a checkAvailability message and return a checkAvailabilityResponse or invalidDataFault.

# Esempio – Hotel reservation system (2)

- **MakeReservation**.
  - A client must provide a name, address, and credit card information, and the service will return a confirmation number if the reservation is successful.
  - The service will return an error message if any data field is invalid.
  - Thus, the service will accept a makeReservation message and return a makeReservationResponse or invalidCreditCardFault message.

# Target namespace

- Defines the namespace for the defined symbols
- Optionally, identifies and URL where the “official” version of the WSDL may be downloaded
- Root element: <description>
- Note: targetNamespace is a **wsdl**:targetNamespace attribute, not an **xsd**:targetNamespace (as in XML Schema)

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    . . . >
  . . .
</description>
```

# Message types (1)

- Use XML Schema to define the message contents (SOAP env:body)
- <types> is a container for XSD definitions
- We define checkAvailability, checkAvailabilityResponse, invalidDataError message types
- All normal and fault message types **must** be defined as single elements at the topmost level
  - Such elements may, in turn, be complex types

# Message types (2)

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    . . . >
  ...
  <types>
    <xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://greath.example.com/2004/schemas/resSvc"
        xmlns="http://greath.example.com/2004/schemas/resSvc">

      <xs:element name="checkAvailability" type="tCheckAvailability"/>
      <xs:complexType name="tCheckAvailability">
        <xs:sequence>
          <xs:element  name="checkInDate" type="xs:date"/>
          <xs:element  name="checkOutDate" type="xs:date"/>
          <xs:element  name="roomType" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>

      <xs:element name="checkAvailabilityResponse" type="xs:double"/>

      <xs:element name="invalidDataError" type="xs:string"/>
    </xs:schema>
  </types>
  . . .
</description>
```

WSDL namespace

XMLSchema namespace

# Interface (1)

- Abstract description of the web service, as a set of “abstract operations”
  - Each operation is a simple client-server interaction
- Message types are specified
- Interaction pattern is specified
  - Example: in-out pattern
    - if the client sends a message in to the service, the service will either send a reply message back out to the client (in the normal case) or it will send a fault message back to the client (in the case of an error)
  - in-only, out-only are also possible
- Operation: opCheckAvailability
  - Uses checkAvailability and checkAvailabilityResponse message types
  - Fault message is also specified

# Interface (2)

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    . . .
    xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions">
  . . .
  <types>
        ...
  </types>

  <interface  name = "reservationInterface" >
    <fault name = "invalidDataFault"
           element = "ghns:invalidDataError"/>

    <operation name="opCheckAvailability"
           pattern="http://www.w3.org/ns/wsdl/in-out"
           style="http://www.w3.org/ns/wsdl/style/iri"
           wsdlx:safe = "true">
        <input messageLabel="In"
              element="ghns:checkAvailability" />
        <output messageLabel="Out"
              element="ghns:checkAvailabilityResponse" />
        <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>
  </interface>
  . . .
</description>
```

# Binding (1)

- <interface> defined *what* message could be exchanged.
- <binding> defines *how*:
  - concrete message format
  - transmission protocol details
- We usually define that the message format will be SOAP 1.2 and the protocol will be HTTP
  - Not the only possibilities supported by WSDL, but the standard choice

# Binding (2)

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  . . .
  <types>
    . . .
  </types>

  <interface  name = "reservationInterface" >
        ...
  </interface>

  <binding name="reservationSOAPBinding"
          interface="tns:reservationInterface"
          type="http://www.w3.org/ns/wsdl/soap"
          wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>
  </binding>
  . . .
</description>
```

namespace for the SOAP 1.2 binding extension that is defined in WSDL 2.0 Part 3

what kind of concrete message format to use: SOAP 1.2

transmission protocol that should be used: HTTP

SOAP message exchange pattern (MEP) that will be used to implement the abstract In-Out pattern

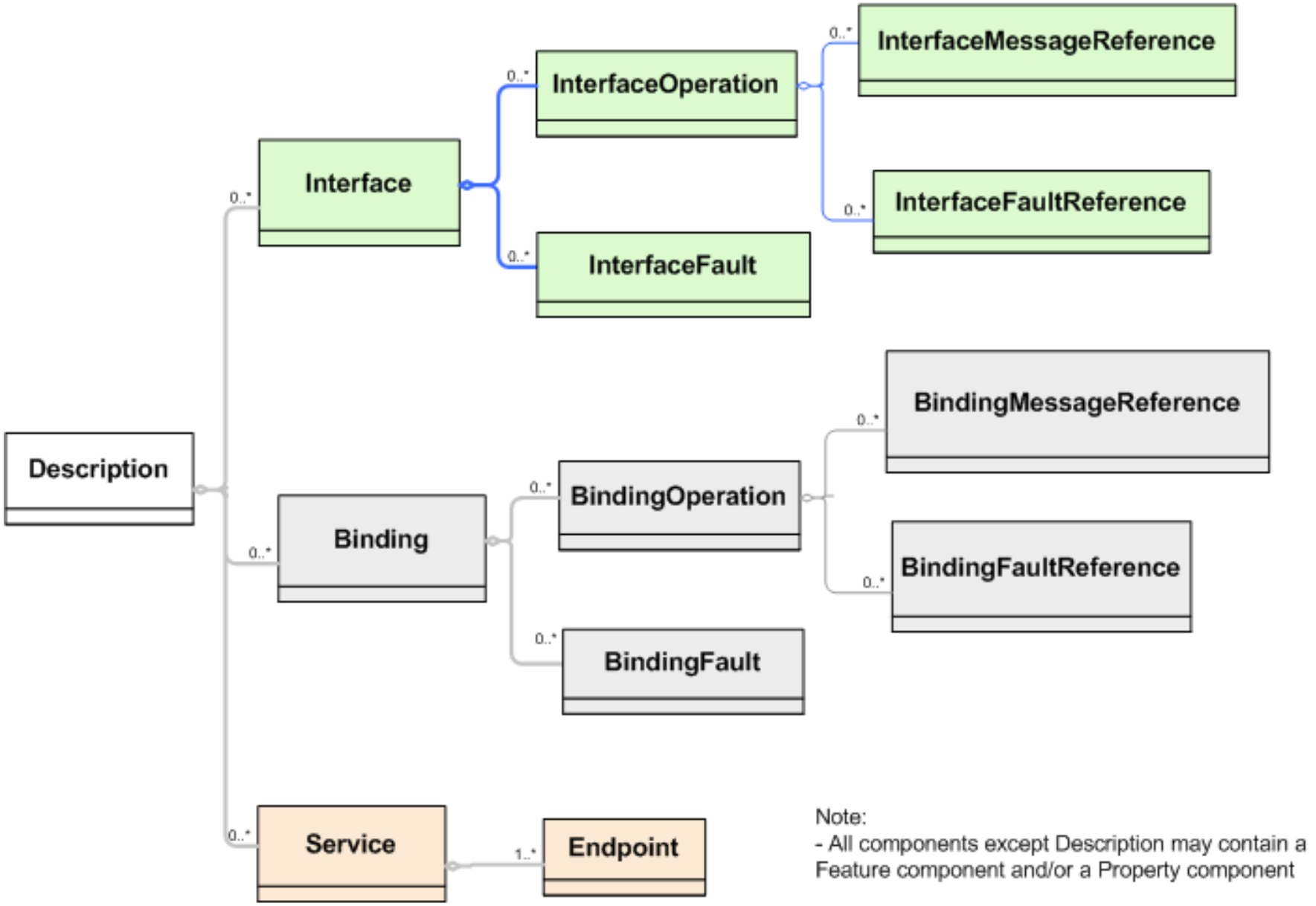
# Service (1)

- <service> specifies *where* the service can be accessed
- For each interface, we list a set of *endpoints* where that service may be accessed
  - Endpoint reference the bindings

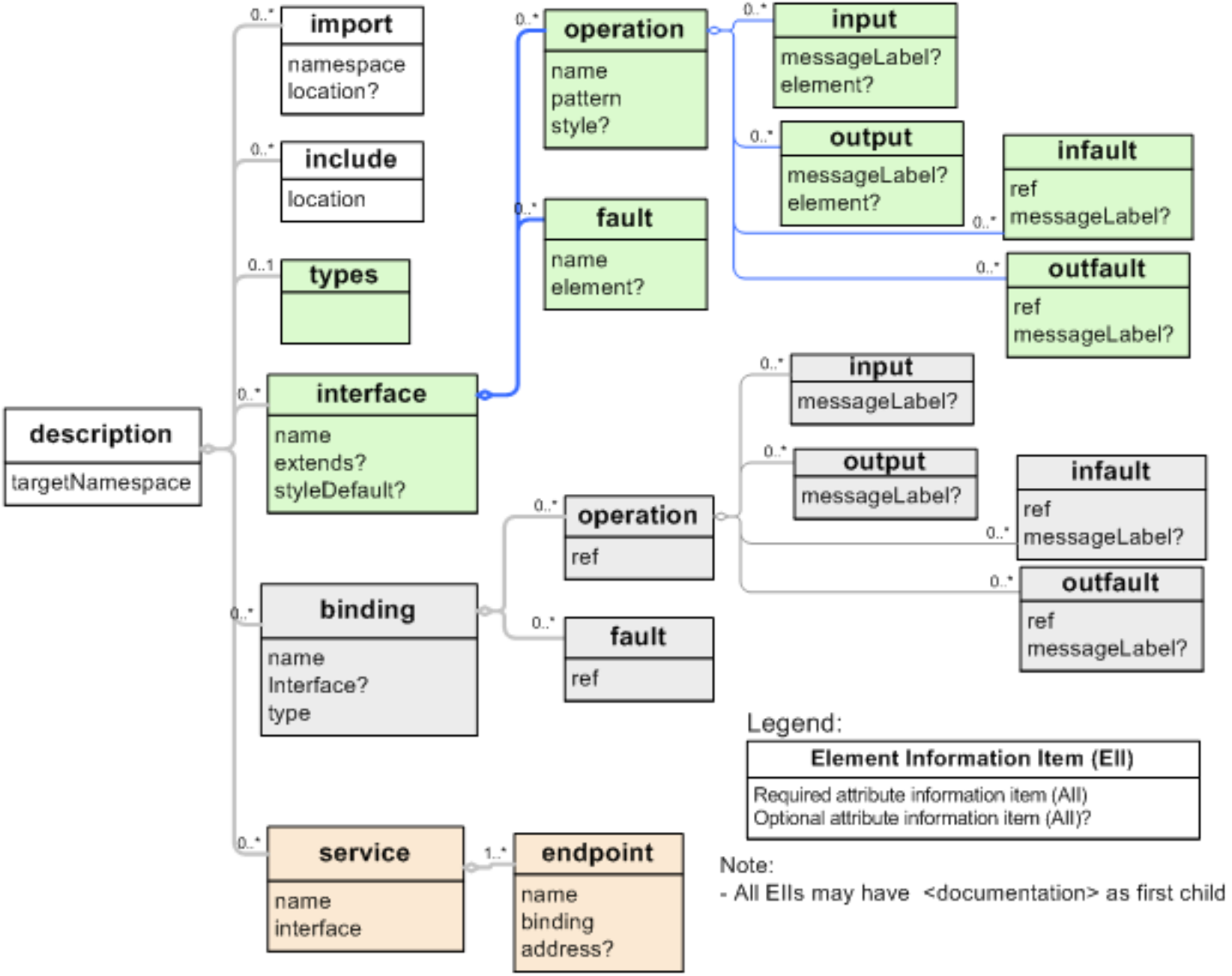
# Service (2)

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  . . .
  <types>
    . . .
  </types>

  <interface  name = "reservationInterface" >
    . . .
  </interface>

  <binding name="reservationSOAPBinding"
          interface="tns:reservationInterface" . . . >
    . . .
  </binding>

  <service name="reservationService"
       interface="tns:reservationInterface">
     <endpoint name="reservationEndpoint"
               binding="tns:reservationSOAPBinding"
               address ="http://greath.example.com/2004/reservation"/>

  </service>

</description>
```

# Gerarchia di un documento WSDL



# Struttura completa



# Esempio WSDL

```
<?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/1999/XMLSchema">
    <element name="TradePriceRequest">
     <complexType>
      <all>
       <element name="tickerSymbol" type="string"/>
      </all>
     </complexType>
    </element>
    <element name="TradePrice">
     <complexType>
      <all>
       <element name="price" type="float"/>
      </all>
     </complexType>
    </element>
   </schema>
  </types>
```

# Esempio WSDL (2)

```
<message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
</message>
<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
 <operation name="GetLastTradePrice">
  <input message="tns:GetLastTradePriceInput"/>
   <output message="tns:GetLastTradePriceOutput"/>
 </operation>
</portType>
```

# Esempio WSDL (3)

```
<binding name="StockQuoteSoapBinding"
        type="tns:StockQuotePortType">
<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"
        namespace="http://example.com/stockquote.xsd"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
     </input>
    <output>
     <soap:body use="literal"
       namespace="http://example.com/stockquote.xsd"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
   </operation>
</binding>
```

# Esempio WSDL (4)

```
<service name="StockQuoteService">
 <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
   <soap:address location="http://example.com/stockquote"/>
  </port>
</service>

</definitions>
```

# Esempio WSDL (5)

```
<binding name="StockQuoteServiceBinding"
 type="StockQuoteServiceType">
  <soap:binding style="rpc"
   transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="getQuote">
             <soap:operation
    soapAction="http://www.getquote.com/GetQuote"/>
     <input>
      <soap:body type="InMessageRequest"
       namespace="urn:live-stock-quotes"
       encoding="http://schemas.xmlsoap.org/soap/encoding/"/>
     </input>
     <output>
      <soap:body type="OutMessageResponse"
       encoding="http://schemas.xmlsoap.org/soap/encoding/"/>
     </output>
   </operation>
 </binding>
```

# Esempio WSDL (6)

```
<service name="StockQuoteService">
 <documentation>My first service
        </documentation>
  <port name="StockQuotePort"
   binding="tns:StockQuoteBinding">
   <soap:address location="http://example.com/stockquote"/>
  </port>
 </service>
</definitions>
```

# Esempio (richiesta)

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
    <m:GetLastTradePrice
     xmlns:m="Some-URI">
      <symbol>MOT</symbol>
    </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# Esempio (risposta)

```
HTTP/1.1 200 OK Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <SOAP-ENV:Body>
  <m:GetLastTradePriceResponse
   xmlns:m="Some-URI">
   <Price>14.5</Price>
  </m:GetLastTradePriceResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# UDDI

- Universal Description, Discovery, and Integration
- Memorizza gli indirizzi (ed i metodi offerti da ciascun servizio) su dei server (pubblici o privati)
- La directory ha due “clienti”
  - Aziende che vogliono pubblicare i propri servizi, rendendoli noti ed accessibili
  - Aziende che vogliono usufruire di determinati servizi
- Non esiste un solo server UDDI, occorre quindi “scegliere” a quale directory collegarsi
- Il server UDDI non contiene il web service, ma solo le informazioni su come raggiungerlo

# Esempio

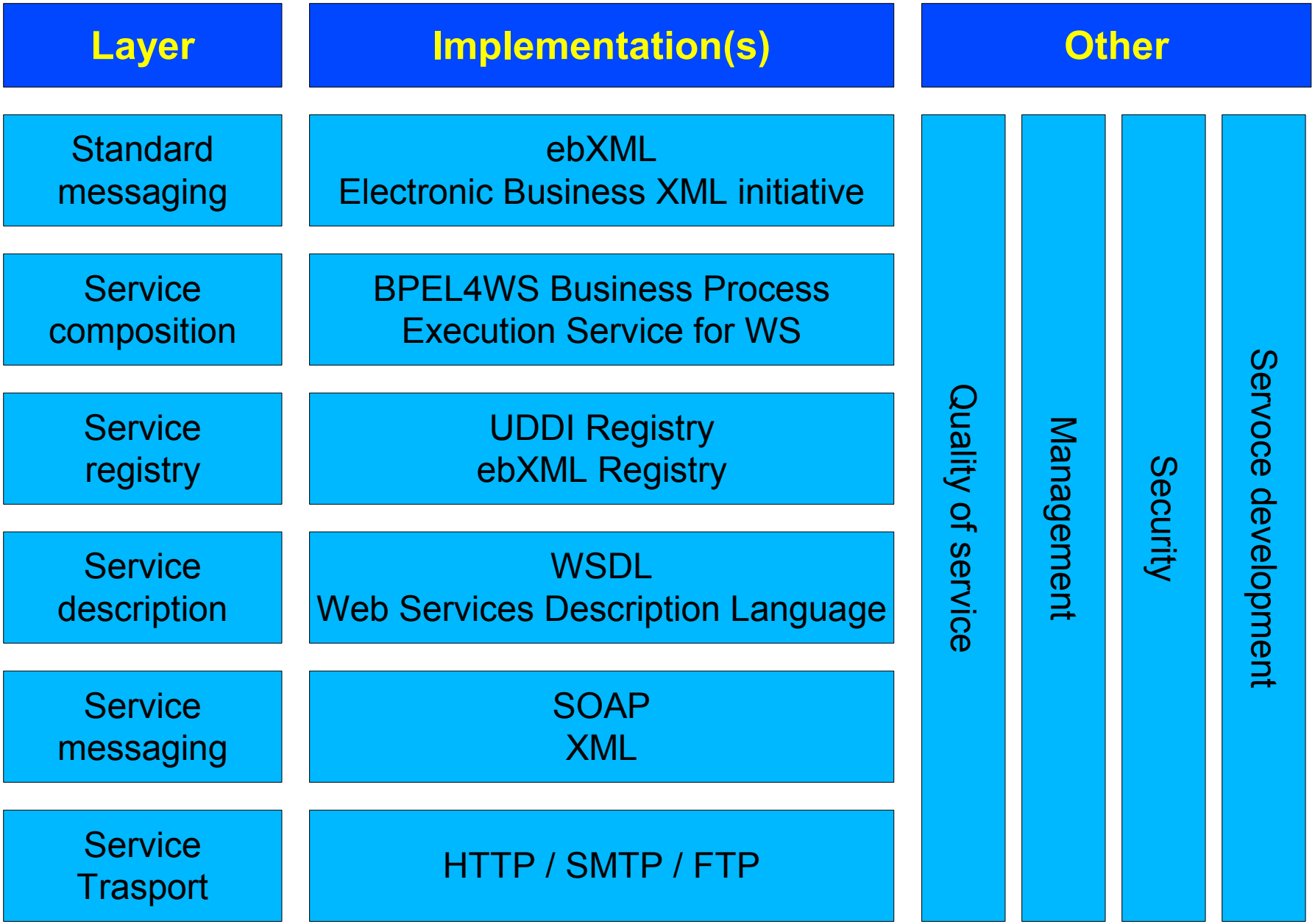
```
<find_business generic="1.0" xmlns="urn:uddi-org:api">
<name>Microsoft</name>
</find_business>
```

# Esemp

```
<find_busi
<name>Mi
</find_busi
```

```
<businessList generic="1.0" operator="Microsoft Corporation"
 truncated="false" xmlns="urn:uddi-org:api">
 <businessInfos>
  <businessInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3">
   <name>Microsoft Corporation</name>
    <description xml:lang="en">
          Empowering people through great software
    </description>
     <serviceInfos>
      <serviceInfo
         businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
         serviceKey="1FFE1F71-2AF3-45FB-B788-09AF7FF151A4">
       <name>Web services for smart searching</name>
      </serviceInfo>
     <serviceInfo
         businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
         serviceKey="8BF2F51F-8ED4-43FE-B665-38D8205D1333">
       <name>Electronic Business Integration Services</name>
     </serviceInfo>
     <serviceInfo
         businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
         serviceKey="611C5867-384E-4FFD-B49C-28F93A7B4F9B">
              <name>Volume Licensing Select Program</name>
     </serviceInfo>
```

# Technology stack



# Bibliografia


- http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html
- http://www.w3.org/TR/soap12-part0/
- http://www.w3.org/TR/wsdl20-primer/
- http://en.wikipedia.org/wiki/SOAP
- J. McGovern, S. Tyagi, M.E. Stevens, S. Mathew, Java Web Services Architecture, Morgan Kaufmann