Computação Distribuída
Cap V – Web Services

Licenciatura em Engenharia Informática

Universidade Lusófona

Prof. Paulo Guedes (paulo.guedes@ulusofona.pt)

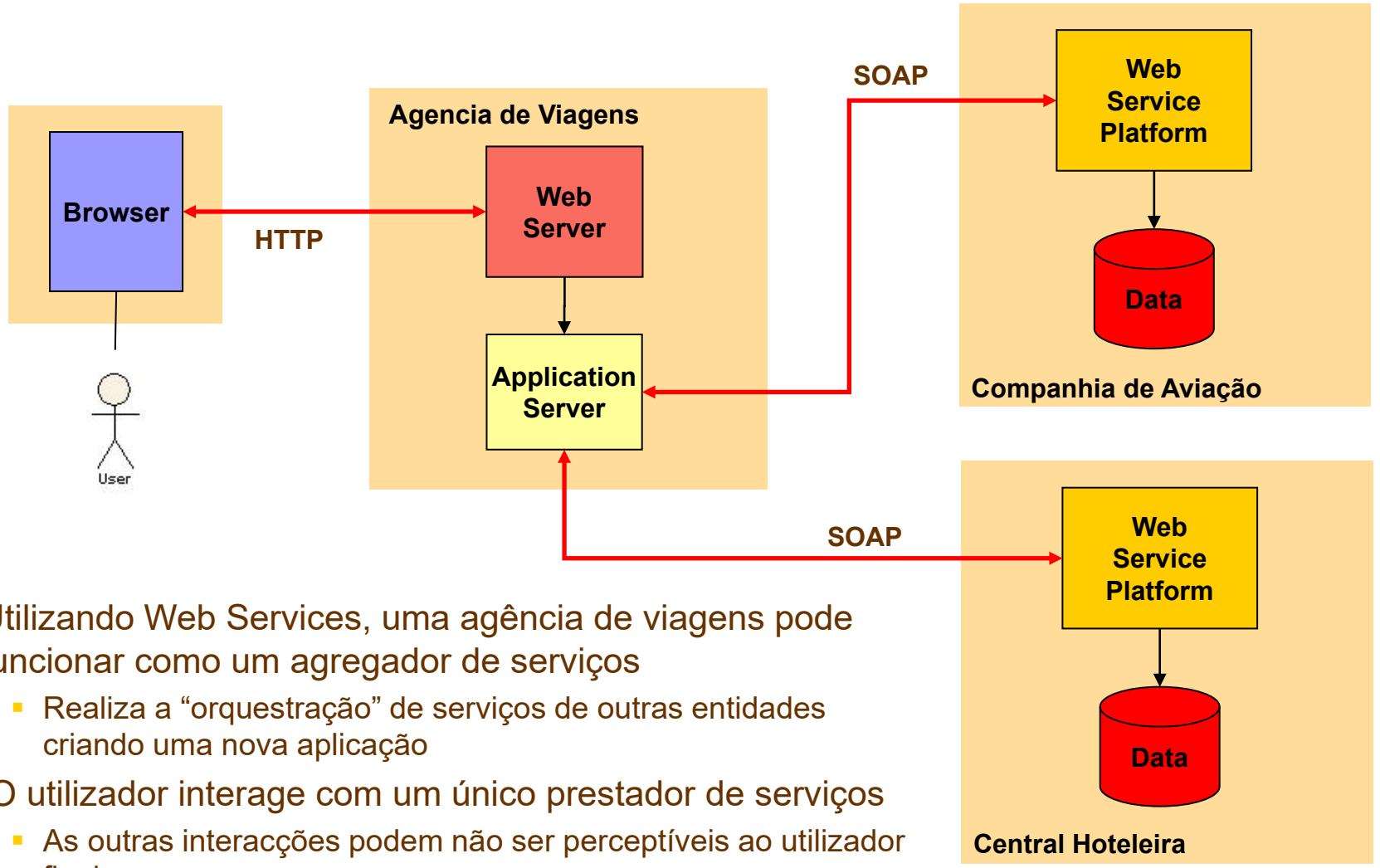
Web Services

- Objetivos e aplicações de Web Services (WS), exemplos
- Linguagem de Definição de Serviços (WSDL)
- Serviço de Directório de WS UDDI, Protocolo SOAP
- Aplicações de Web Services

Evolução da Arquitectura de Serviços

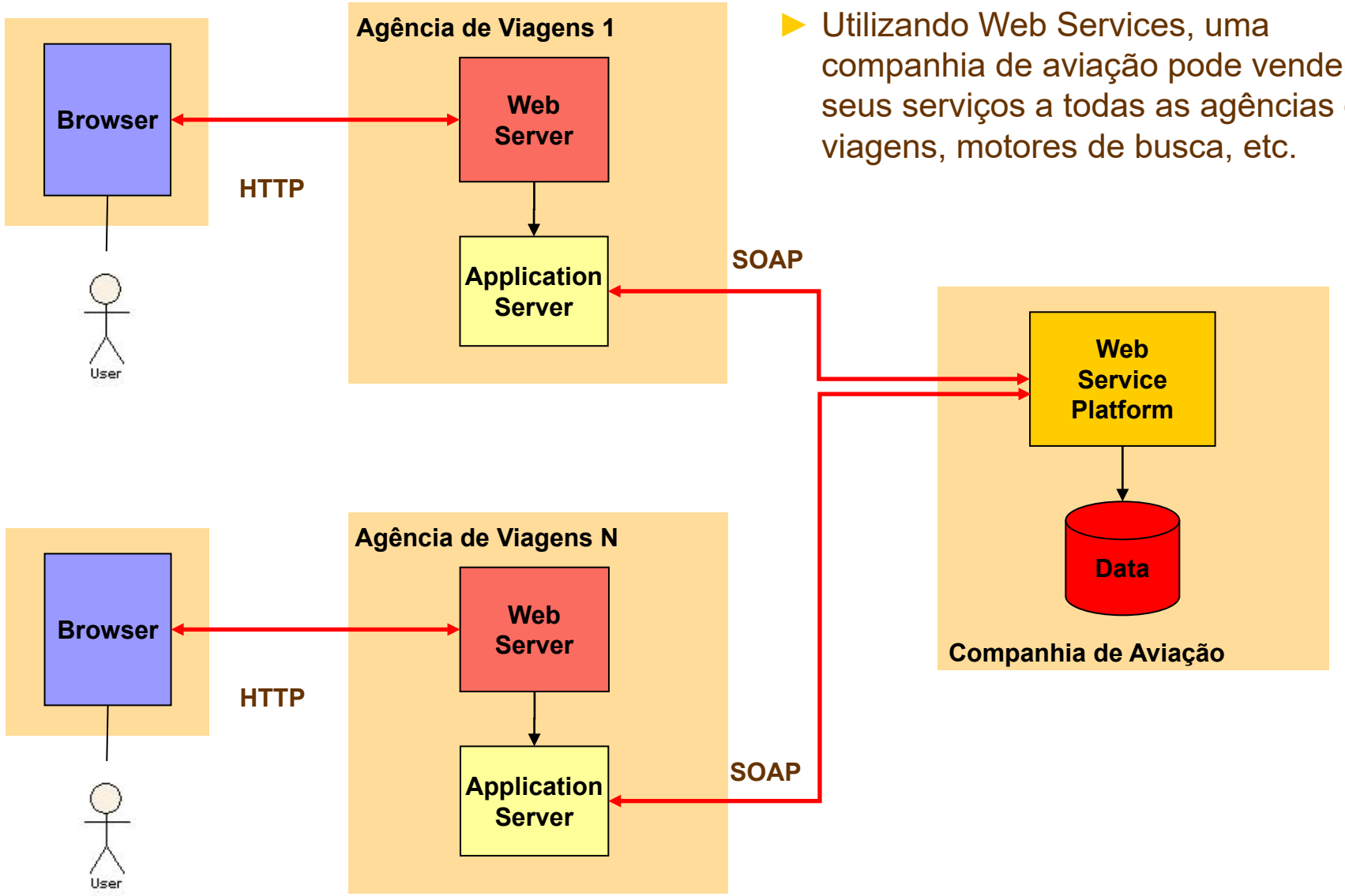
- A generalização da Internet como espaço de negócio implica modelos de interacção cada vez mais complexos
- Os serviços são fornecidos por aglomerados de empresas, com modelos de negócio dinâmicos e complexos
- A arquitectura da Web adapta-se a esta realidade
  - *Web Services* - maior flexibilidade e interoperabilidade
    - Definição das interfaces de forma independente das linguagens de programação (WSDL)
    - Clientes e servidores podem estar escritos em linguagens diferentes
    - Formato das mensagens independente da implementação (SOAP)

Aplicações de Web Services: Integração entre empresas distintas



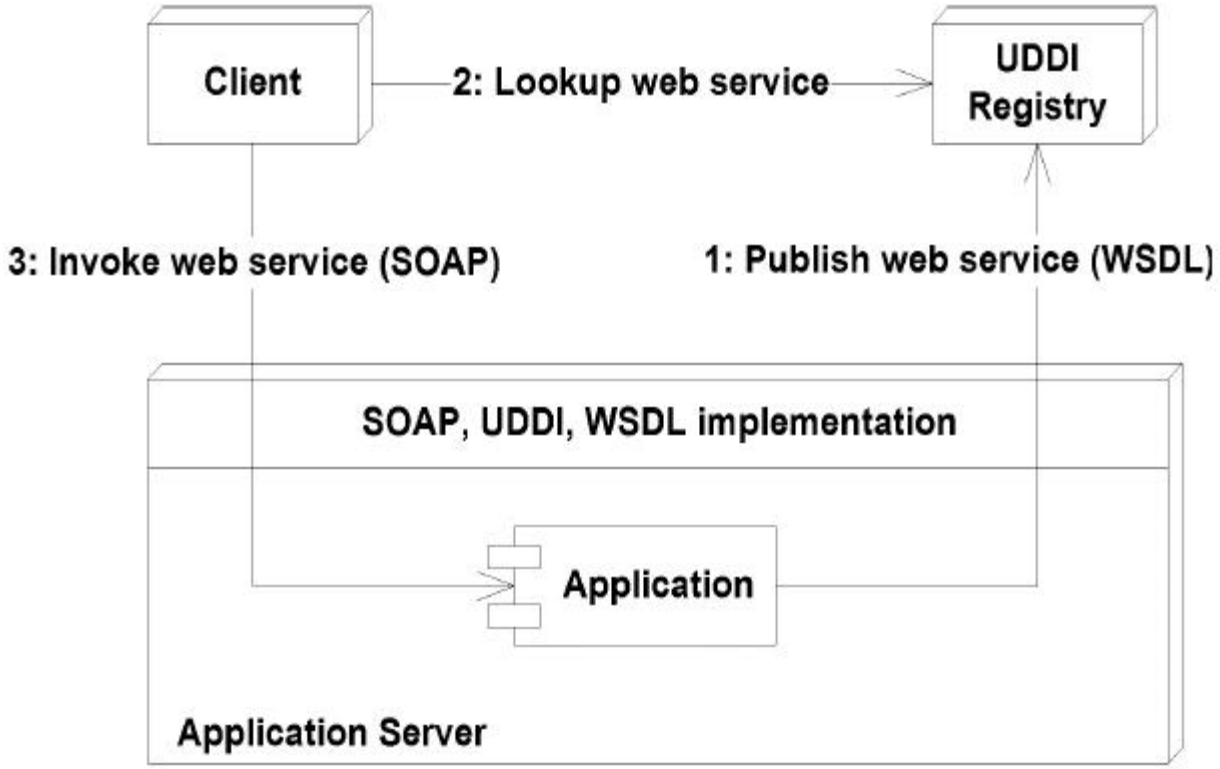
- Utilizando Web Services, uma agência de viagens pode funcionar como um agregador de serviços
  - Realiza a "orquestração" de serviços de outras entidades criando uma nova aplicação
- O utilizador interage com um único prestador de serviços
  - As outras interacções podem não ser perceptíveis ao utilizador final

Aplicações de Web Services: Integração entre empresas distintas

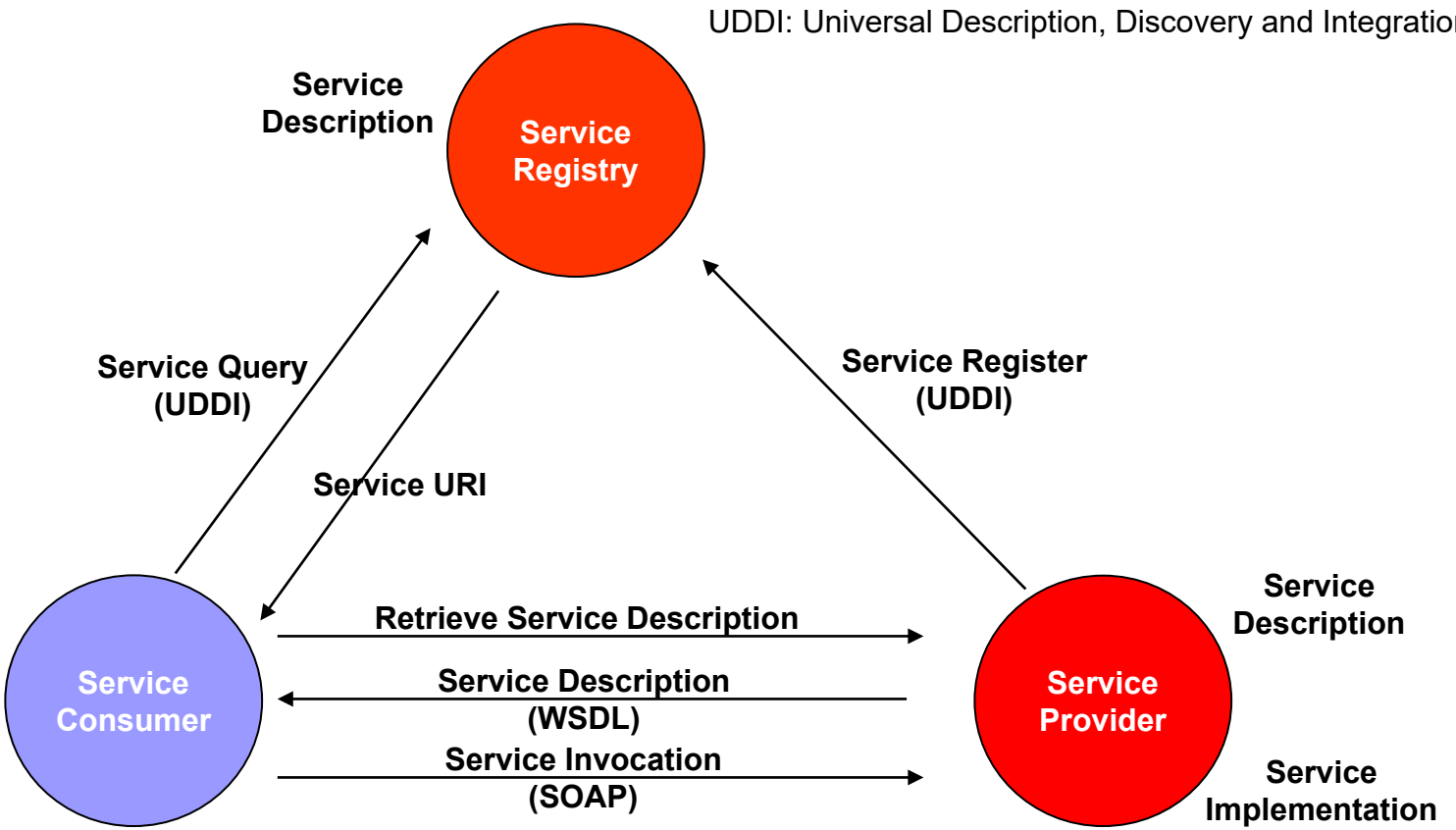


- Utilizando Web Services, uma companhia de aviação pode vender os seus serviços a todas as agências de viagens, motores de busca, etc.

Web Services – Publicação, Lookup e Invocação



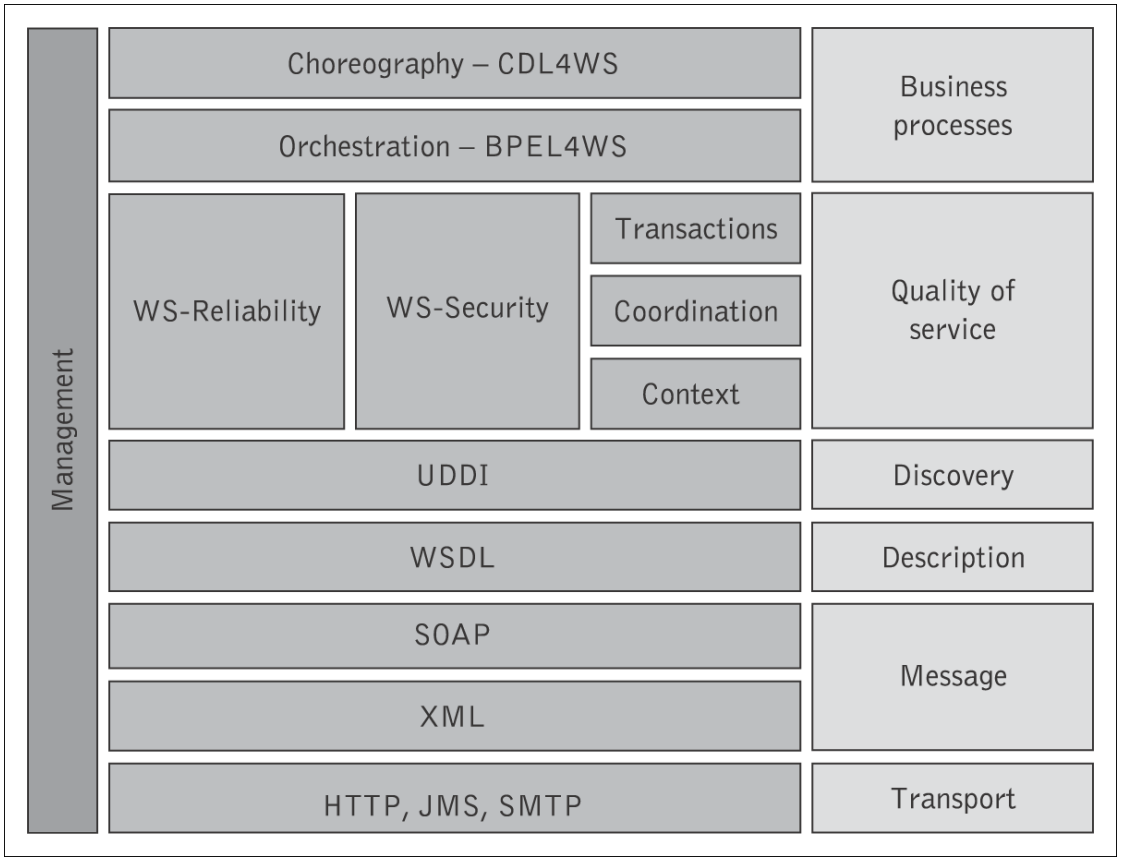
Web Services: obtenção dinâmica dos detalhes da invocação dos serviços



A obtenção da informação da localização do Service Provider e da descrição dos detalhes da invocação dos serviços podem ser realizadas dinamicamente

Web Services: Arquitectura

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008



- **Os Web Services podem são implementados por um conjunto de várias tecnologias e standards**
- **Alguns Web Services específicos providenciam funcionalidades utilizadas por outros Web Services**
  - **Directory Service, Security, etc..**

Interface e Implementação de Serviços

- A interface de serviço define a funcionalidade visível do exterior e fornece os meios para a aceder
  - A interface define as características do serviço, ou seja as operações disponíveis, os seus parâmetros, os tipos de dados e protocolos de acesso
  - Permite às aplicações que a invocam de determinar a sua funcionalidade, como pode ser invocada e quais os resultados previstos
- A implementação de um serviço instancia uma interface cujos detalhes estão escondidos dos seus utilizadores
  - A mesma interface pode ser chamada por qualquer cliente, mesmo clientes totalmente desconhecidos de quem fornece o servidor
  - A mesma interface pode ser implementada de forma diversa e em linguagens diferentes por vários fornecedores de serviços

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008

Web Services: Características

- A definição de um Web Service contém o conjunto de operações que podem ser invocadas através da sua interface
- A invocação de Web Service pode ser efectuada de duas formas distintas
  - Invocação de funções em modo request-reply
  - Troca assíncrona de documentos em formato XML
  - O protocolo de comunicação SOAP define várias formas de mensagens possíveis
- Os Web Services não impõem nenhum modelo de programação específico
  - Permite a interacção de qualquer tipo de aplicação que implemente a interface especificada
- Formato das Mensagens
  - Tanto as mensagens como o protocolo que as transporta (SOAP) são representados em XML
- Referência
  - Um Web Service é representado por um URI, que reveste geralmente a forma de um URL

Exemplo: WS do Google Maps

- The Maps Static API returns an image (either GIF, PNG or JPEG) in response to an HTTP request via a URL. For each request, you can specify the location of the map, the size of the image, the zoom level, the type of map, and the placement of optional markers at locations on the map. You can additionally label your markers using alphanumeric characters.
- The following example generates a Maps Static API for Berkeley, CA:

  https://maps.googleapis.com/maps/api/staticmap?center=Berkeley,CA&zoom=14&size=400x400&key=YOUR_API_KEY
- Mais informação em:

  https://developers.google.com/maps/documentation/maps-static/dev-guide

Google Maps API (exemplo)

https://maps.googleapis.com/maps/api/staticmap?center=40.714728,-73.998672&**zoom=12**&size=400x400&key=YOUR_API_KEY

https://maps.googleapis.com/maps/api/staticmap?center=40.714728,-73.998672&**zoom=14**&size=400x400&key=YOUR_API_KEY

Google Maps API (exemplo)

- https://maps.googleapis.com/maps/api/directions/outputFormat?parameters
- **Request Parameters**
- Certain parameters are required while others are optional. As is standard in URLs, all parameters are separated using the ampersand (&) character. The list of parameters and their possible values are enumerated below.
- **Required parameters**
- **origin** — The address, textual latitude/longitude value, or place ID from which you wish to calculate directions.
  - If you pass an address, the Directions service geocodes the string and converts it to a latitude/longitude coordinate to calculate directions. This coordinate may be different from that returned by the Geocoding API, for example a building entrance rather than its center.
    - origin=24+Sussex+Drive+Ottawa+ON
  - If you pass coordinates, they are used unchanged to calculate directions. Ensure that no space exists between the latitude and longitude values.
    - origin=41.43206,-81.38992
- **destination** — The address, textual latitude/longitude value, or place ID to which you wish to calculate directions. The options for the destination parameter are the same as for the origin parameter, described above.
- **key** — Your application's API key. This key identifies your application for purposes of quota management. Learn how to get a key.

Google Maps API (exemplo)

A sample HTTP request is shown below, calculating the route from Chicago, IL to Los Angeles, CA via two waypoints in Joplin, MO and Oklahoma City, OK.

```
<DirectionsResponse>
 <status>OK</status>
 <geocoded_waypoint>
  <geocoder_status>OK</geocoder_status>
  <type>locality</type>
  <type>political</type>
  <place_id>ChIJ7cv00DwsDogRAMDACa2m4K8</place_id>
 </geocoded_waypoint>
 <geocoded_waypoint>
  <geocoder_status>OK</geocoder_status>
  <type>locality</type>
  <type>political</type>
  <place_id>ChIJ69Pk6jdlyIcRDqM1KDY3Fpg</place_id>
 </geocoded_waypoint>
 <geocoded_waypoint>
  <geocoder_status>OK</geocoder_status>
  <type>locality</type>
  <type>political</type>
  <place_id>ChIJgdL4flSKrYcRnTpP0XQSojM</place_id>
 </geocoded_waypoint>
 <geocoded_waypoint>
  <geocoder_status>OK</geocoder_status>
  <type>locality</type>
  <type>political</type>
  <place_id>ChIJE9on3F3HwoAR9AhGJW_fL-I</place_id>
 </geocoded_waypoint>
…
```

Amazon Web Services: Product Advertising API

Operations

The following operations are available in the Product Advertising API.

Search

- ItemSearch (p. 170)

Lookup

- BrowseNodeLookup (p. 183)
- ItemLookup (p. 186)
- SimilarityLookup (p. 192)

Cart

- CartAdd (p. 195)
- CartClear (p. 200)
- CartCreate (p. 203)
- CartGet (p. 208)
- CartModify (p. 212)

AWS Product Advertising API: ItemSearch

- Description
- The ItemSearch operation searches for items on Amazon. The Product Advertising API returns up to ten items per search results page.

- An ItemSearch request requires a search index and the value for at least one parameter. For example, you might use the BrowseNode parameter for Harry Potter books and specify the Books search index.

Parameter	Description	Required
Actor	Actor name associated with the item. You can enter all or part of the name. Type: String Default: None	No
Artist	Artist name associated with the item. You can enter all or part of the name. Type: String Default: None	No
AudienceRating	Movie ratings based on MPAA ratings or age, depending on locale. You can specify one or more values in a comma-separated list in a REST request or with multiple elements in a SOAP request. Type: String Default: None Valid values: See Movie Ratings by Locale.	No

Perguntas a que devo ser capaz de responder

- Quais são os problemas que se pretendem resolver com os Web Services ?
- Porque razão o Sun RPC e o Java RMI não são adequados para resolver esses problemas ?

Descrição de Interfaces e Serviços

- Para permitir a invocação dos serviços, é necessário descrevê-los de uma forma independente das plataformas
  - Utilização de XML
- A descrição de serviços contém todos os dados necessários para a sua invocação
  - Espaço de nomeação utilizados
  - Tipo de mensagem
  - Semântica de invocação
  - Protocolo utilizado
  - Localização do serviço
  - Métodos e argumentos
- A descrição dos Web Services é feita em WSDL - Web Services Description Language

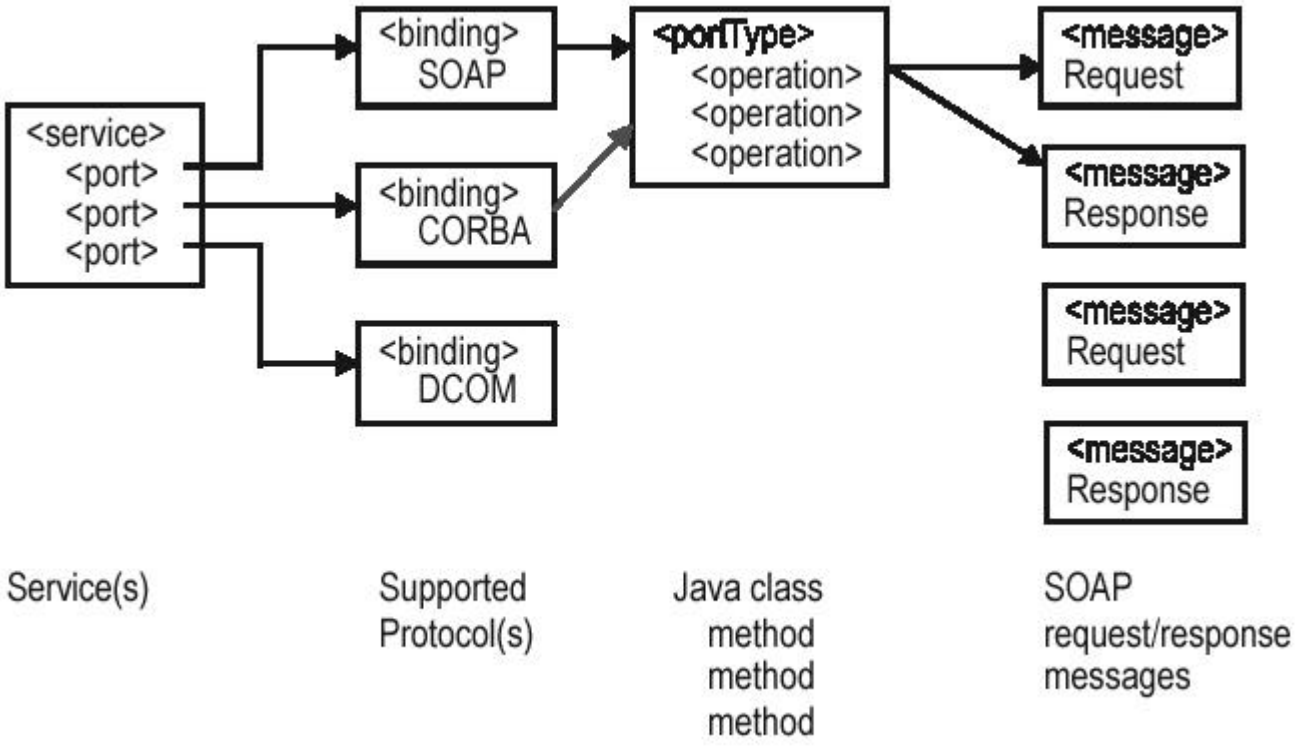
Componentes do WSDL

WSDL: Linguagem de definição de interface (IDL) baseada em XML para definir:

- Interface do serviço
- Protocolo de acesso
- Endpoints de acesso
- Aspectos de implementação

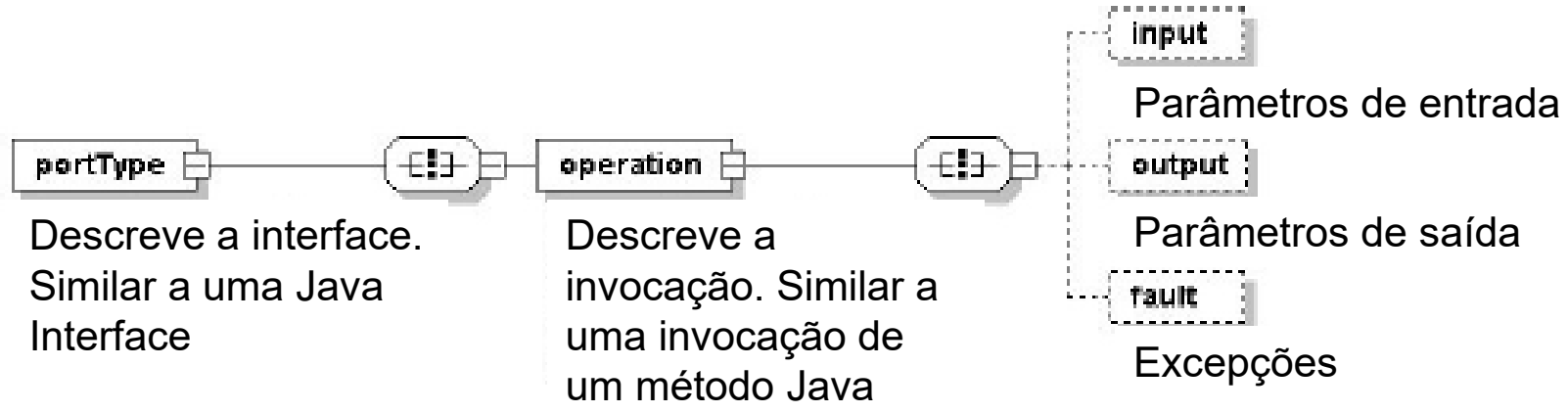
Elementos do WSDL

- Service: definição da **localização** (URL) do serviço
- Port: definição do **endereço físico** (URL) do serviço
- Binding: definição do **formato da mensagem**
- PortType: definição dos **métodos** a invocar
- Message: definição das **mensagens** trocadas
- Types: definição dos **tipos de dados** utilizados



PortType

- portType: descreve a interface do Web Service
  - Contém um único atributo name: nome_do_WSPortType
  - Contém um ou mais elementos operation, pode definir elementos input, output e fault

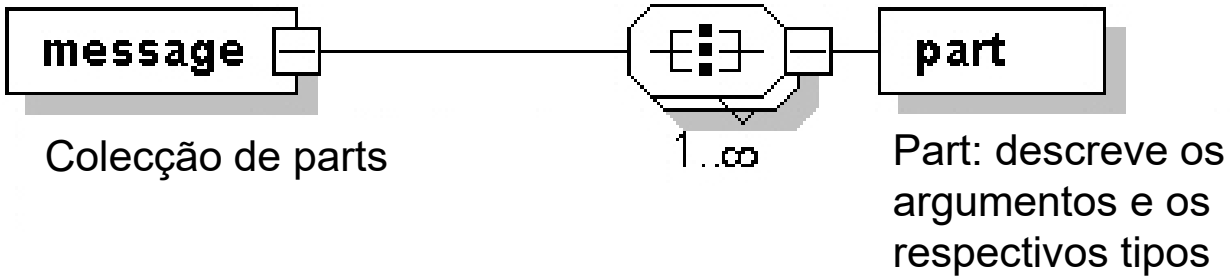


PortType: Exemplo

```
<!-- Port Type Definition Example -->
    <portType name="weatherCheckPortType">
        <operation name="checkTemperature">
            <input message="checkTemperatureRequest"/>
            <output message="checkTemperatureResponse"/>
        </operation>
        <operation name="checkHumidity">
            <input message="checkHumidityRequest"/>
            <output message="checkHumidityResponse"/>
        </operation>
    </portType>
```

Message

- Message: colecção de parts; part: named argument e respectivo tipo
  - Cada elemento message pode ser usado como mensagem input, output ou fault numa operation
  - O atributo type da part pode ser qualquer tipo standard do schema XSD ou um tipo definido pelo utilizador

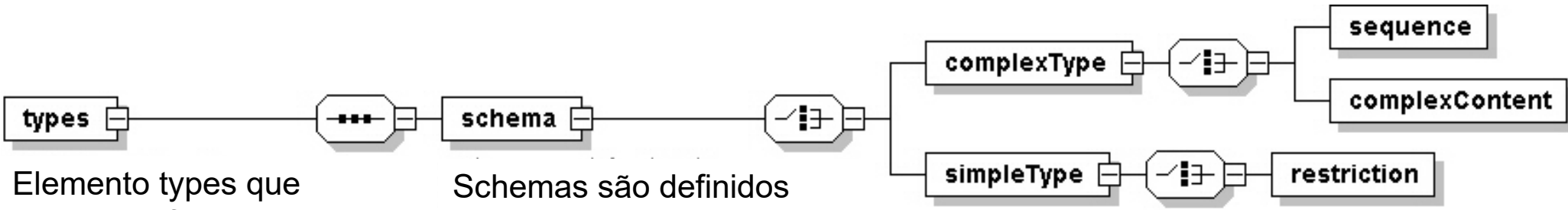


Message: exemplo

```
<!-- Message Definitions -->
<message name="checkTemperatureRequest">
      <part name="location" type="xsd:string">
</message>
<message name="checkTemperatureResponse">
      <part name="result" type="xsd:double">
</message>
<message name="checkHumidityRequest">
      <part name="location" type="xsd:string">
</message>
<message name="checkHumidityResponse">
      <part name="result" type="ns:HummidityType"
</message>
```

Types

- Types: definição abstracta dos tipos definidos pelo utilizador
  - Um documento WSDL pode conter no máximo um elemento types
  - O elemento types pode conter simpleType ou complexType.
  - Elementos são definidos com um name e um atributo type



Elemento types que permite definir os tipos de dados definidos pelo utilizador

Schemas são definidos neste nível, para ficarem acessíveis daqui para a frente

Types: exemplo

```
<!-- Type Definitions -->
<types>
    <xsd:schema targetNamespace="http://weather.com/ns"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:complexType name="HumidityType">
            <xsd:sequence>
                <xsd:element name="loc" type="xsd:string">
                <xsd:element name="humd" type="xsd:double">
                <xsd:element name="temp" type="xsd:double">
            </xsd:sequence>
        </xsd:complexType>
    </xsd:schema>
</types>
```

Binding

- Binding: define o formato da mensagem, de uma forma dependente do protocolo
  - Cada portType pode ter um ou mais elementos binding associados
  - Para um portType, o elemento binding especifica um par messaging e transport (SOAP/HTTP, SOAP/SMTP, etc).

Binding: exemplo

```
<binding name="WeatherBinding" type="weatherCheckPortType">
    <soap:binding style="rpc“ transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="checkTemperature">
        <soap:operation soapAction="" />
            <input>
                <soap:body use="encoded" namespace="checkTemperature"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="checkTemperature"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
            </output>
    </operation>
</binding>
```

Port

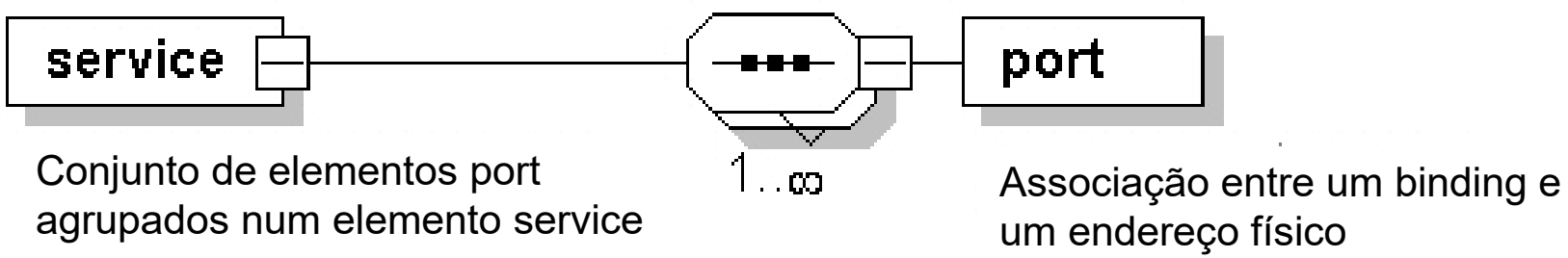
- Port: especifica o endereço do WebService, normalmente um URL
  - Associa um único endereço, protocol-specific, ao elemento binding
  - Ports têm que ser únicos num documento WSDL

- Exemplo:

```
<port name="WeatherCheck“ binding="wc:WeatherCheckSOAPBinding">
    <soap:address location="http://host/WeatherCheck"/>
</port>
```

Service

- Service: colecção de ports relacionados, identificados por um único nome
  - Um documento WSDL pode conter vários service, mas por convenção contém apenas um
  - Cada serviço tem que ter um nome único
  - A convenção de nome é *Nome*Service

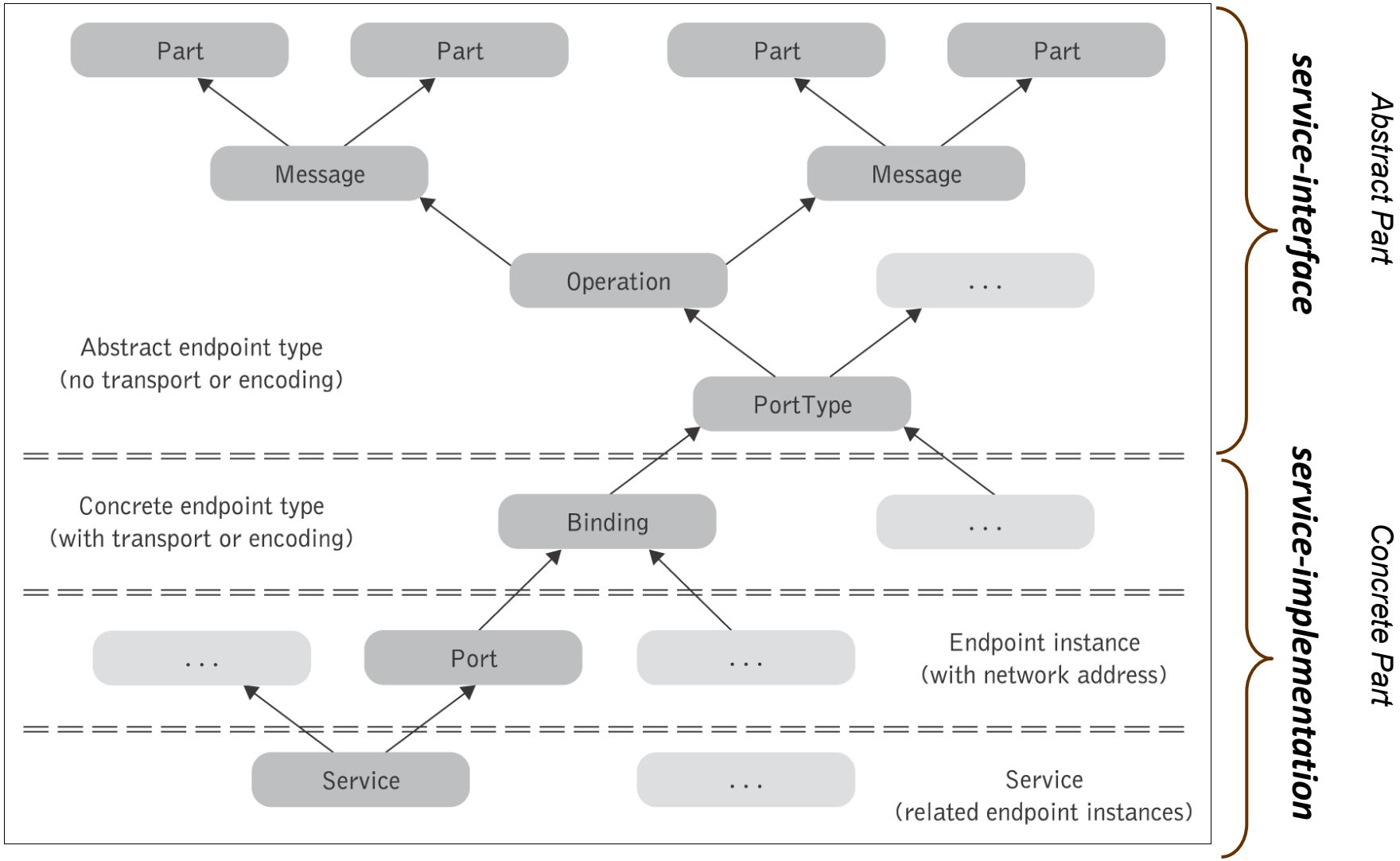


Service: exemplo

```
<!-- Service definition -->
<service name="WeatherCheckService">
      <port name="WeatherCheckSOAP“
            binding="wc:WeatherCheckSOAPBinding">
      <soap:address location="http://host/WeatherCheck"/>
      </port>
      <port name="WeatherCheckSMTP"
            binding="wc:WeatherCheckSMTPBinding">
      <soap:address location="http://host/WeatherCheck"/>
      </port>
</service>
```

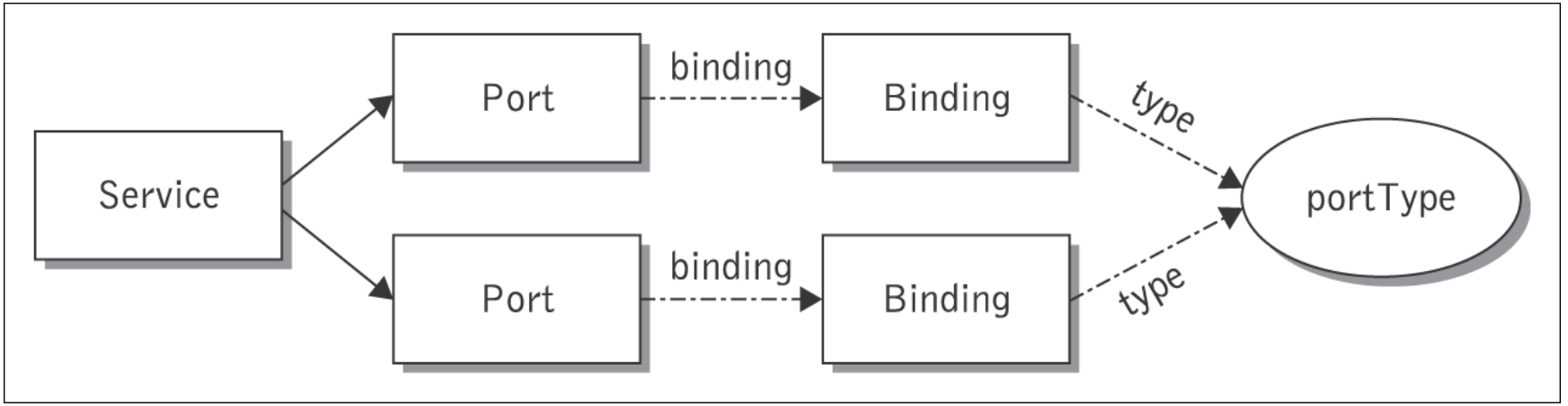
Hierarquia de Elementos WSDL

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008



Ligação entre Elementos WSDL

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008



Ligação entre a interface e a implementação do WebService

Exemplo: MapImage.wsdl (1)

```
<!-- generated by GLUE on Sat Dec 07 18:30:26 PST 2013 -->
<definitions
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:http=http://schemas.xmlsoap.org/wsdl/http/
        xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:ns11=http://www.themindelectric.com/package/com.esri.is.services.common.v2.geom/
        xmlns:ns12="http://www.themindelectric.com/package/com.esri.is.services.glue.v2.mapimage/"
        name="MapImage“
        targetNamespace="http://arcweb.esri.com/v2">
```

Exemplo: MapImage.wsdl (types)

```
<types>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.themindelectric.com/package/com.esri.is.services.glue.v2.mapimage/">
<xsd:import namespace="http://www.themindelectric.com/package/com.esri.is.services.common.v2/"/>
<xsd:import namespace="http://www.themindelectric.com/package/java.lang/"/>
<xsd:complexType name="MapImageInfo">
        <xsd:sequence>
                <xsd:element name="mapUrl" nillable="true" type="xsd:string"/>
                <xsd:element name="legendUrl" nillable="true" type="xsd:string"/>
                <xsd:element name="mapExtent" nillable="true" type="ns11:Envelope"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="MapImageOptions">
        <xsd:sequence>
                <xsd:element name="dataSource" nillable="true" type="xsd:string"/>
                <xsd:element name="mapImageSize" nillable="true" type="ns12:MapImageSize"/>
                <xsd:element name="mapImageFormat" nillable="true" type="xsd:string"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="MapImageSize">...</xsd:complexType>
<xsd:complexType name="PixelCoord">...</xsd:complexType>
```

Exemplo: MapImage.wsdl (messages)

```
<message name="getMap0In">
        <part name="mapExtent" type="ns11:Envelope">
        <documentation>Envelope of the map.</documentation> </part>
        <part name="mapImageOptions" type="ns12:MapImageOptions"> <documentation>
        MapImageOptions contains parameters for using map image.
        </documentation> </part>
        <part name="token" type="xsd:string">
        <documentation>
        Authentication token for accessing Map Image Web Service.
        </documentation> </part>
</message>
<message name="getMap0Out">
        <part name="Result" type="ns12:MapImageInfo">
        <documentation>
        MapImageInfo contains information about the URL, the legend URL, and the Envelope of the map
        image. </documentation>  </part>
</message>
<message name="getLayerInfo1In">...</message>
<message name="getLayerInfo1Out">...</message>
<message name="getBestMap2In">...</message>
```

Exemplo: MapImage.wsdl (portTypes)

```
<portType name="IMapImage">
        <operation name="getMap" parameterOrder="mapExtent mapImageOptions token">
                <documentation>Gets the map for the particular envelope.</documentation>
                <input name="getMap0In" message="tns:getMap0In"/>
                <output name="getMap0Out" message="tns:getMap0Out"/>
        </operation>
        <operation name="getLayerInfo" parameterOrder="dataSource">...</operation>
        <operation name="getBestMap" parameterOrder="mapImageOptions token">...</operation>
</portType>
```

Exemplo: MapImage.wsdl (bindings)

```
<binding name="IMapImage" type="tns:IMapImage">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="getMap">
<soap:operation soapAction="getMap" style="rpc"/>
<input name="getMap0In">
<soap:body use="encoded" namespace="http://arcweb.esri.com/v2" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</input>
<output name="getMap0Out">
<soap:body use="encoded" namespace="http://arcweb.esri.com/v2" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</output>
</operation>
<operation name="getLayerInfo">...</operation>
</binding>
```

Exemplo: MapImage.wsdl (Service)

<service name="MapImage">

<documentation>

The Map Image Web Service enables users to input a geographic extent and several optional items (e.g., themes, type of map, size of image, map annotation) and receive the location of an output image file. The ArcWeb Service is intended to support application developers who would like to provide dynamic map content within their Internet applications. The Map Image Web Service provides access to data sources from several leading data publishers.

</documentation>

<port name="IMapImage" binding="tns:IMapImage">

<soap:address location="http://arcweb.esri.com/services/v2/MapImage"/>

</port>

</service>

</definitions>

Perguntas a que devo ser capaz de responder

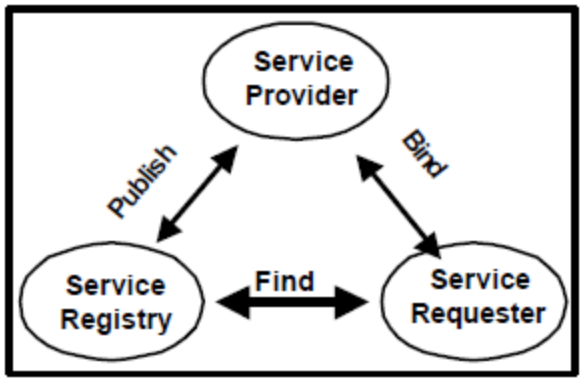
- Qual é a IDL dos Web Services ? Como é que essa IDL se compara com a IDL do Sun RPC e a IDL do Java RMI ?
- Porque razão os Web Services definem o formato das mensagens trocadas entre cliente e servidor, ao contrário do Sun RPC e do Java RMI ?
- Os que se define na parte abstrata e na parte concreta do WSDL ?
- Quais são e para que servem os 6 elementos do WSDL ?

UDDI - Universal, Description, Discovery and Integration

- A project to speed interoperability and adoption for Web services
  - Standards-based specifications for service description and discovery
  - Shared operation of a web based business registry
  - Partnership among industry and business leaders (more than 300 companies)
- Business registry has three components:
  - White pages: Information about the business (address, contacts, etc.)
  - Yellow pages: Categorization of the business and its services
  - Green pages: Technical information about services provided by a business
- Free, public, interconnected UDDI servers are deployed today
- Private UDDI Registry is available today for enterprise integration

UDDI - Componentes

- Service Registry
  - Provides support for publishing and locating services
  - Like telephone yellow pages
- Service Provider
  - Provides e-business services
  - Publishes availability of these services through a registry
- Service Requester
  - Finds required services via the Service Registry
  - Binds to services via Service Provider

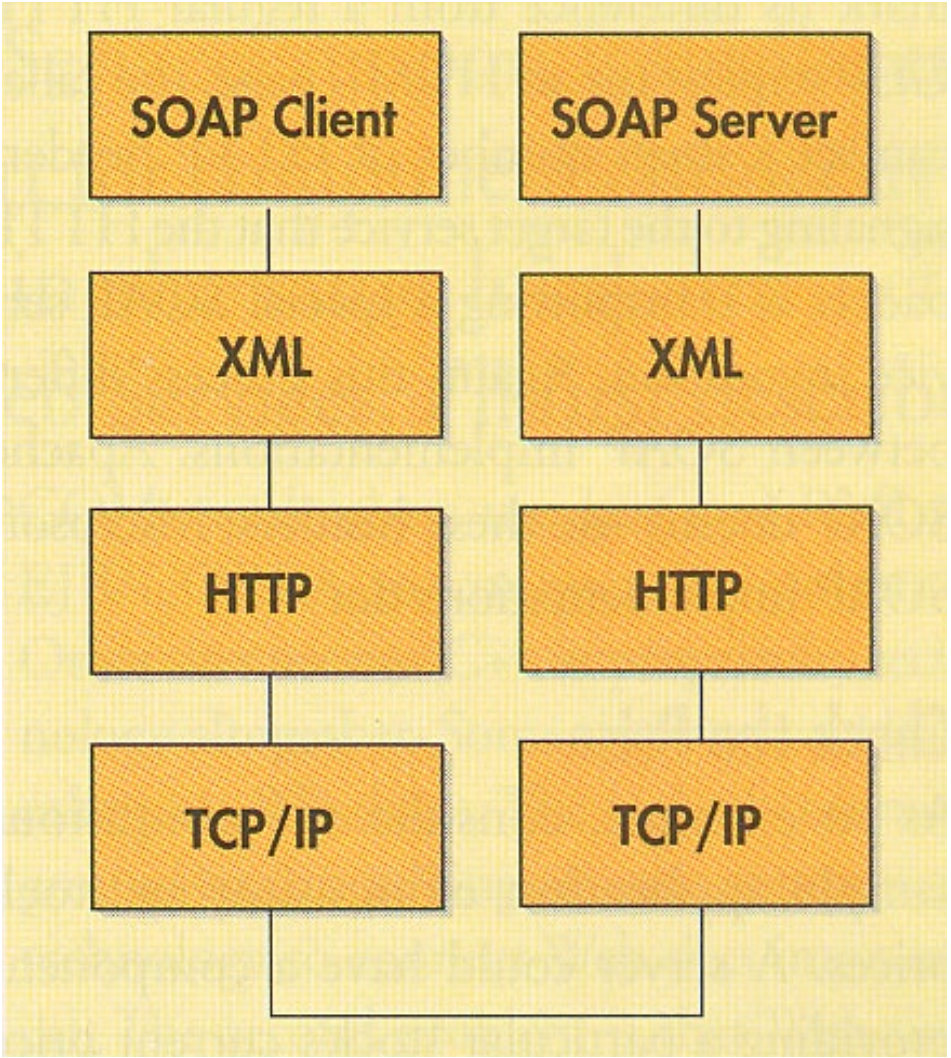


SOAP: Simple Object Access Protocol

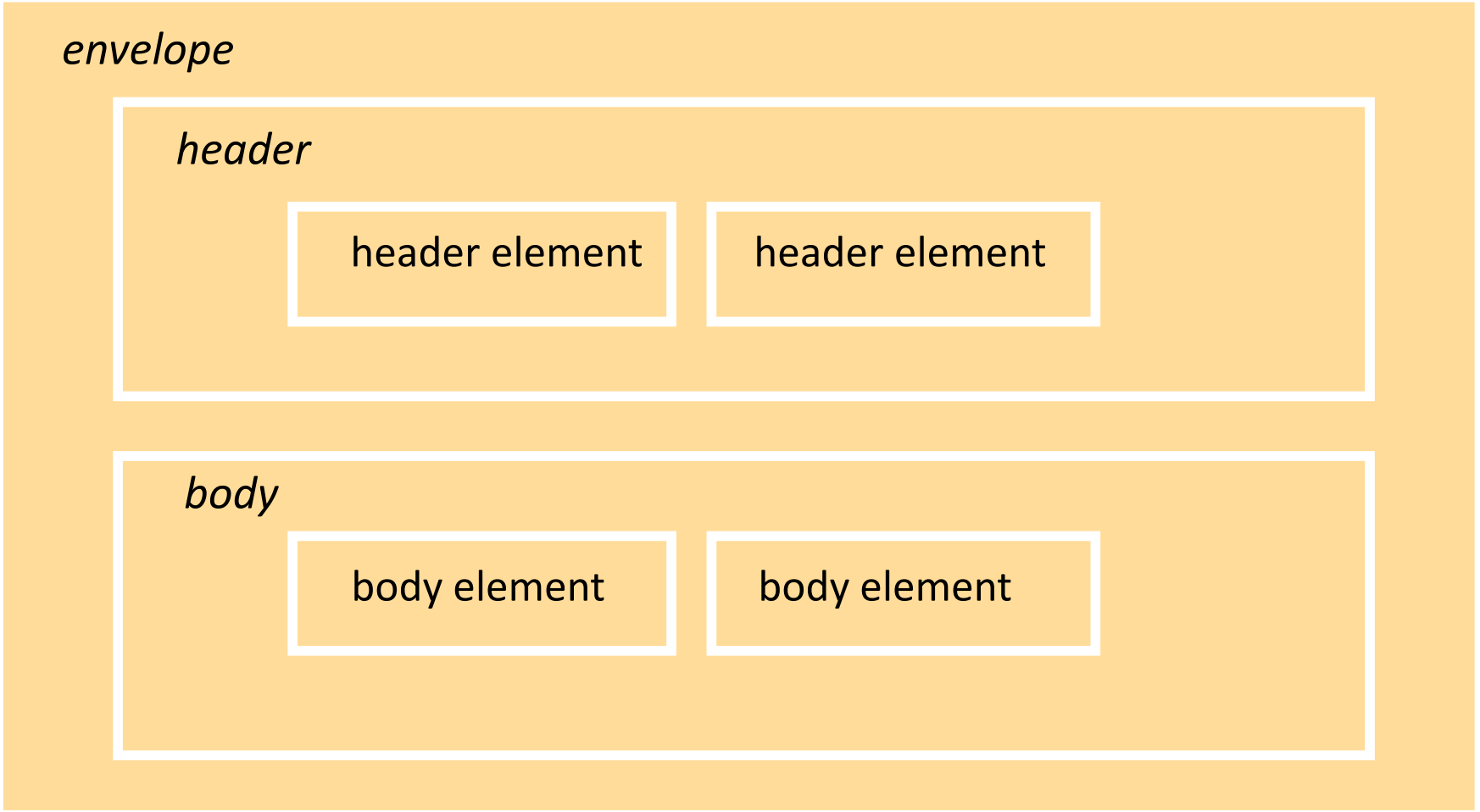
- W3C specification: a lightweight protocol for exchange of information in a decentralized, distributed environment.
- It is an XML based protocol that consists of three parts:
  - An envelope that defines a framework for describing what is in a message and how to process it
  - A set of encoding rules for expressing instances of application-defined datatypes
  - And a convention for representing remote procedure calls and responses
- A especificação do SOAP indica
  - Como utilizar XML para representar o conteúdo das mensagens
  - Como combinar mensagens simples para implementar *request-reply*
  - As regras de processamento dos elementos XML contidos nas mensagens
  - Como utilizar o protocolo subjacente para enviar as mensagens

SOAP: Arquitectura

- SOAP é um protocolo aberto que permite uma forma uniforme de realizar RPCs utilizando HTTP como protocolo de comunicações e XML para a serialização dos dados
- Na versão mais simples, uma mensagem SOAP é enviada através de um comando POST HTTP



Formato de uma Mensagem SOAP



Mensagem SOAP

- Uma mensagem SOAP é enviada num envelope, que contém um *header* e um *body*
- O envelope pode conter vários *headers*
  - O envelope é utilizado para estabelecer o contexto do serviço
    - Ex. tipo de encoding e name spaces utilizados na mensagem
  - Os elementos do *header* podem ser modificados pelos intermediários
- O *body* contém os dados associados ao tipo de mensagem
  - Documento, argumento ou resultados
  - Formatado de acordo com a definição do Web Service
  - No caso de comunicação RR, é indicado o tipo da mensagem e a operação invocada

Exemplo de Mensagem SOAP

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soap:Envelope xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mrns0="http://webservices.tekever.eu/ctt"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <soap:Body
      soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    - <mrns0:CustoEMSInternacional>
        <zona xsi:type="xs:int">2</zona>
        <peso xsi:type="xs:int">1000</peso>
      </mrns0:CustoEMSInternacional>
    </soap:Body>
  </soap:Envelope>
```

- Ver descrição do Web Service em
  - http://webservices.tekever.eu/ctt

Mensagens SOAP Pergunta/Resposta

Request

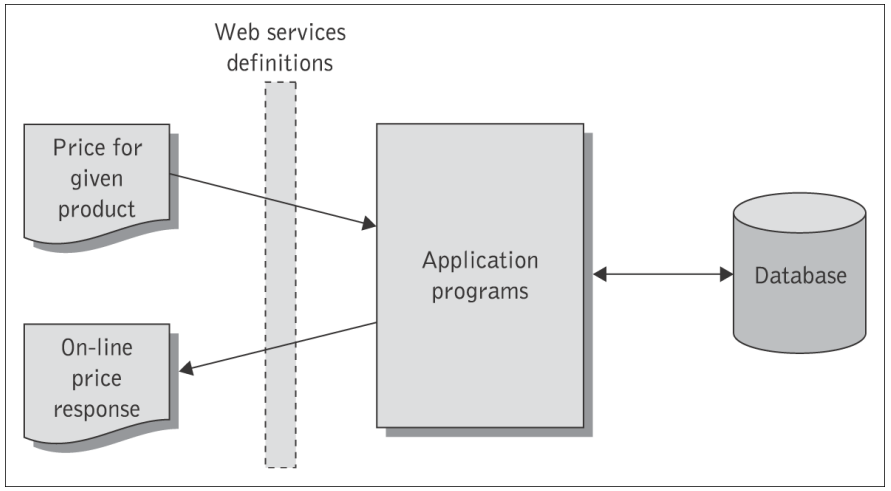
```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:ns1="http://calculator.me.org/">
  - <soapenv:Body>
    - <ns1:sub>
        <i>3</i>
        <j>7</j>
      </ns1:sub>
    </soapenv:Body>
  </soapenv:Envelope>
```

Reply

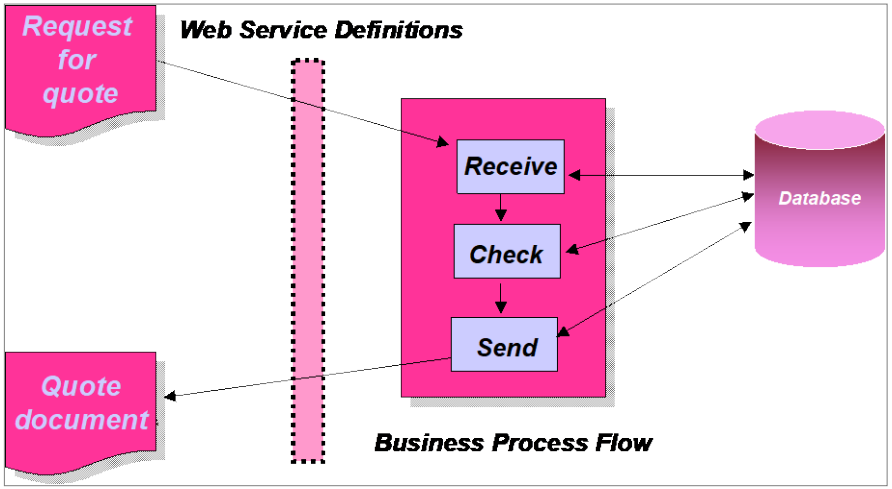
```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:ns1="http://calculator.me.org/">
  - <soapenv:Body>
    - <ns1:subResponse>
        <return>-4</return>
      </ns1:subResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

Modelos de Comunicação SOAP

- O SOAP suporta dois estilos de comunicação:
  - (Remote Procedure Call) RPC-Style
  - Document-Style



RPC-style interaction

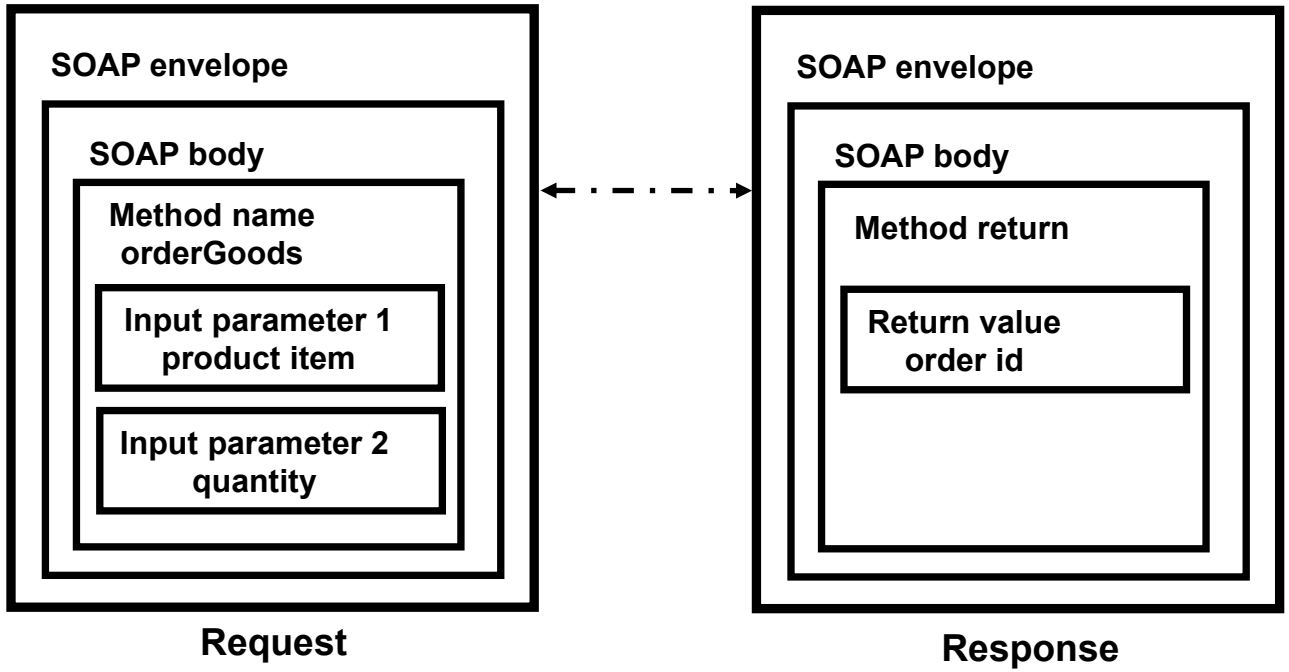


Document-style interaction

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008

Serviço SOAP RPC-Style

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008



- Um Web Service de RPC-Style comporta-se como um objecto remoto para a aplicação cliente
- A interacção entre o cliente e o serviço é centrada numa interface de específicação de serviço
- O cliente exprime os seus pedidos através de uma invocação de método com um conjunto de argumentos, que retornam uma resposta contendo o valor associado

Exemplo mensagem SOAP RPC-Style

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008

```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
           <tx:Transaction-id
           xmlns:t="http://www.transaction.com/transactions"
               env:mustUnderstand='1'>
              512
           </tx:Transaction-id>
    </env:Header>
    <env:Body>
           <m:GetProductPrice>
              <product-id>  450R6OP  </product-id >
           </m:GetProductPrice >
    </env:Body>
</env:Envelope>
```

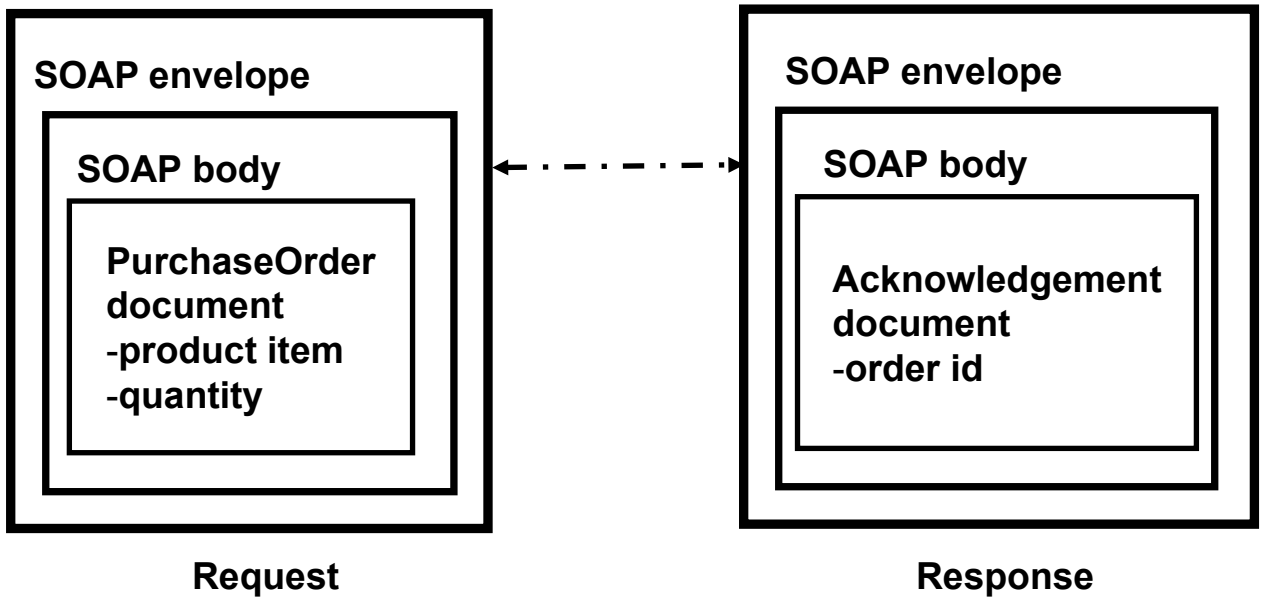
Example of RPC-style SOAP body

```
                    <env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
 xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
           <--! - Optional context information -->
    </env:Header>
    <env:Body>
           <m:GetProductPriceResponse>
              <product-price>   134.32  </product-price>
           </m:GetProductPriceResponse>
    </env:Body>
</env:Envelope>
```

Example of RPC-style SOAP response message

Serviço SOAP Document-style

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008



- Num Web Service Document-Style, o corpo da mensagem contém um fragmento de documento XML, sem estrutura explícita
- O runtime SOAP aceita o elemento do corpo da mensagem tal como é enviado e passa-o à aplicação para o qual é destinado sem modificação
- Pode ou não haver resposta associada a esta mensagem

Exemplo mensagem SOAP Document-Style

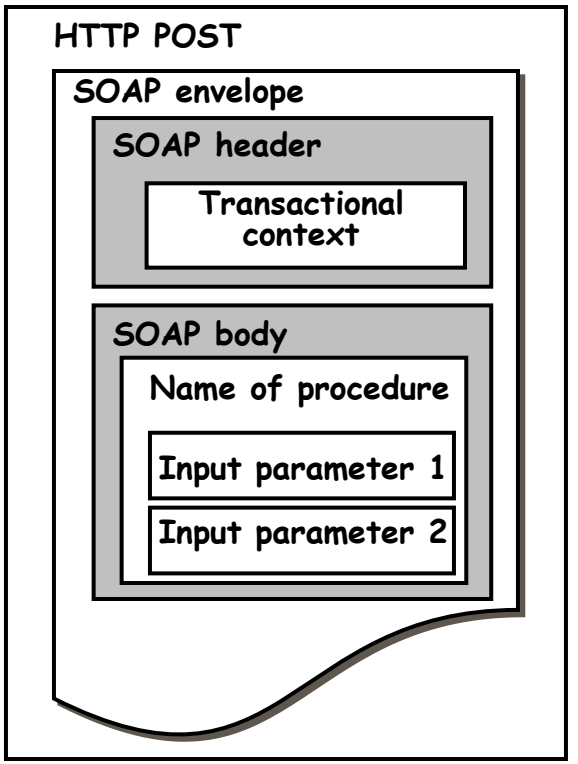

```
<env:Envelope
 xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">

    <env:Header>
            <tx:Transaction-id
           xmlns:t="http://www.transaction.com/transactions"
                env:mustUnderstand='1'>
               512
    </env:Header>
    <env:Body>
            <po:PurchaseOrder oderDate="2004-12-02"
           xmlns:m="http://www.plastics_supply.com/POs">
           <po:from>
             <po:accountName>   RightPlastics   </po:accountName>
                  <po:accountNumber>   PSC-0343-02  </po:accountNumber>
           </po:from>
           <po:to>
             <po:supplierName>  Plastic Supplies Inc.  </po:supplierName>
             <po:supplierAddress>  Yara Valley Melbourne  </po:supplierAddress>
           </po:to>
           <po:product>
               <po:product-name> injection molder </po:product-name>
               <po:product-model> G-100T </po:product-model>
               <po:quantity> 2 </po:quantity>
           </po:product>
            </ po:PurchaseOrder >
    </env:Body>
</env:Envelope>
```

Example of document-style SOAP body

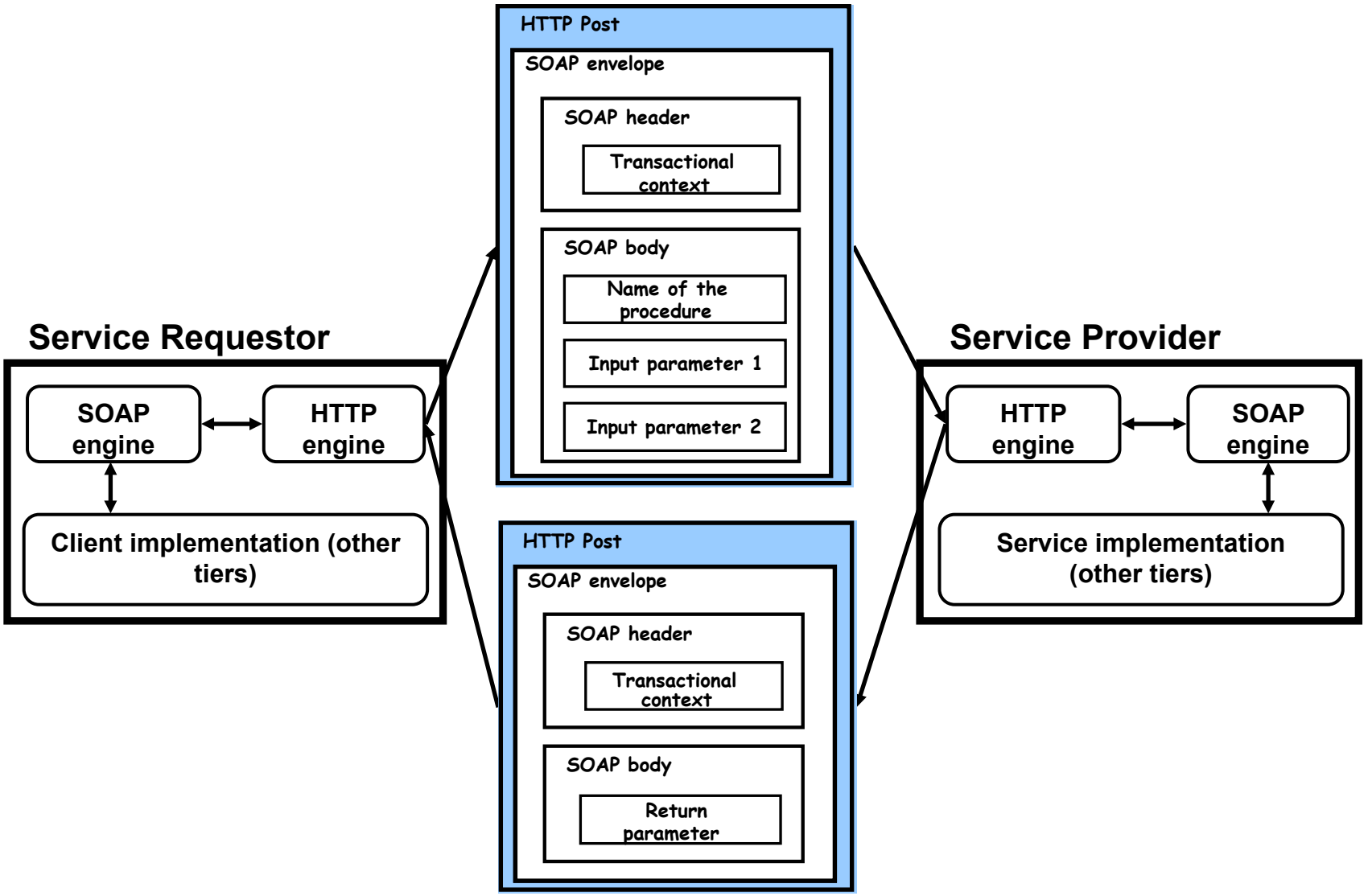
SOAP e HTTP

- A ligação (***binding***) SOAP a um protocolo é a descrição de como a mensagem é enviada usando esse protocolo de transporte
- A ligação típica do SOAP é HTTP
- O SOAP pode usar os métodos GET ou POST
  - GET: o pedido não é uma mensagem SOAP só a resposta
  - POST: pedido e resposta são ambos mensagens SOAP
- O SOAP utiliza os mesmos códigos de erro e estado do HTTP de forma a que as respostas possam ser directamente interpretadas pelo módulo SOAP



Invocação SOAP em HTTP

From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008



Aspectos do SOAP

- As mensagens SOAP são independentes do protocolo, mas subsiste um problema a nível do endereçamento
  - Não contém referências ao endereço de destino
  - O protocolo utilizado especifica o endereço
- No caso da utilização de HTTP em modo *request-reply*
  - O POST transporta a mensagem de *request*
  - A resposta ao POST retorna o *reply*
  - O header HTTP
    - Indica um *content-type: application/soap+xml*
    - Contém uma referência para o método: *Action: <Web Service URL>#method*
    - Existe aqui alguma dependência entre o protocolo HTTP e a mensagem
- WS-Adressing
  - Extensão do protocolo SOAP para incluir directivas de endereçamento nos headers
    - Identificação de mensagens
    - Definição de end-points (origem e destino)

Exemplo de Header HTTP para SOAP

```
POST /CalculatorWSApplication/CalculatorWSService?Tester HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.9) Gecko/20061206 Firefox/1.5.0.9
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://localhost:8080/CalculatorWSApplication/CalculatorWSService?Tester
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
action=add&PARAMadd0=3&PARAMadd1=4

HTTP/1.x 200 OK
X-Powered-By: Servlet/2.5
Pragma: no-cache
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 1276
Date: Sun, 07 Jan 2007 14:40:18 GMT
Server: Sun Java System Application Server Platform Edition 9.0_01
```

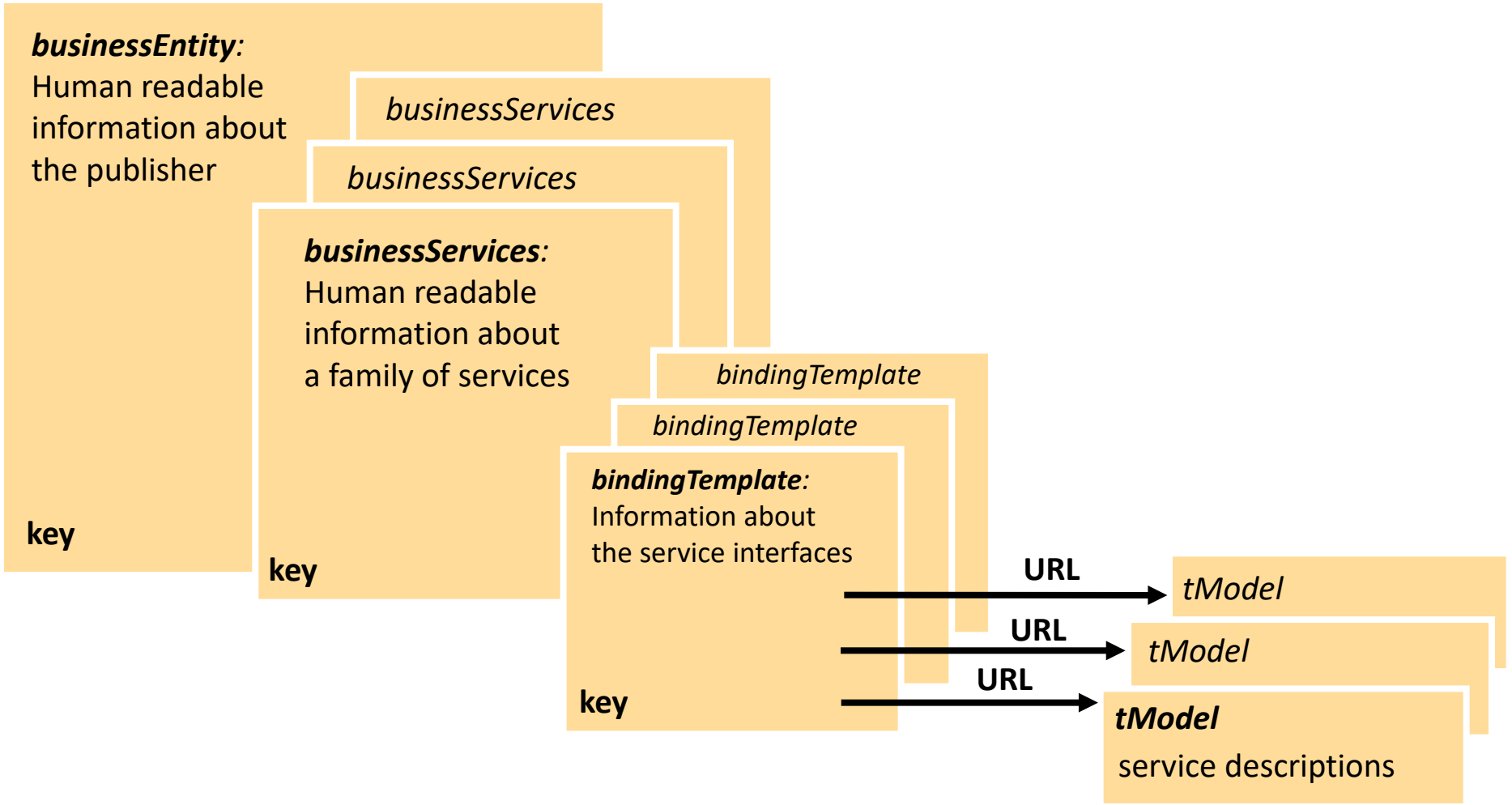
Serviço de Directório Web Services: UDDI

- Universal Description, Discovery, and Integration (UDDI)
  - Serviço de nomes e de directório para acesso a informação genérica sobre Web Services
- Permite dois tipos de acessos
  - White Pages - nomes
  - Yellow Pages - atributos
- O UDDI tem uma organização hierárquica em 4 níveis:
  - **businessEntity**: características da organização que fornece os serviços, tais como actividade, nome, endereço, etc..
  - **businessServices**: serviços agrupados por tipo designados por um nome e um ramo de actividade
  - **bindingTemplate**: endereço dos Web Services e referências para as suas descrições (por URL)
  - **tModel**: descrição detalhada do serviço, geralmente fornecida por um ficheiro wsdl
- Cada elemento (excepto o tModel) é identificado por uma chave e pode ser acedido directamente

Operações UDDI

- O UDDI fornece interfaces de acesso para procura e obtenção de informação sobre os WS
  - Operações de Procura (find_xxx) recebem um nome e retornam uma chave
    - find_business, find_service, find_binding, find_tModel
  - Operações para obtenção de informação (get_xxx) recebem uma chave e retornam dados
    - get_BusinessDetail, get_ServiceDetail, get_BindingDetail, get_tModelDetail
  - A partir da obtenção do tModel, o serviço pode ser invocado se o cliente tiver direitos de acesso
- Interface de Publicação
  - Permite registar um serviço e actualizar informação associada
- Registos
  - O conjunto da informação de uma dada instância de UDDI é armazenada em registos contidos em vários servidores que podem ser replicados
- Exemplos de serviços UDDI
  - XMethods
  - Microsoft

Estruturas do UDDI



Exemplo de um UDDI -XMethods

Business Details	
Business Name:	xmethods.net
Description:	
Business Key:	D1387DB1-CA06-24F8-46C4-86B5D895CA26
Related Businesses:	D1387DB1-CA06-24F8-46C4-86B5D895CA26 [1]
UDDI Operator:	XMethods

Contact	Description	Person Name	Address	Phone

Discovery URL	Usage Note
http://66.28.98.121:9004//?businessKey=D1387DB1-CA06-24F8-46C4-86B5D895CA26	businessEntity

Business Classifications		
Key Name	**Key Value**	tModel Key

Business Identifiers		
Key Name	**Key Value**	tModel Key

http://soapclient.com/uddisearch.html

Lista de Serviços da *businessEntity*

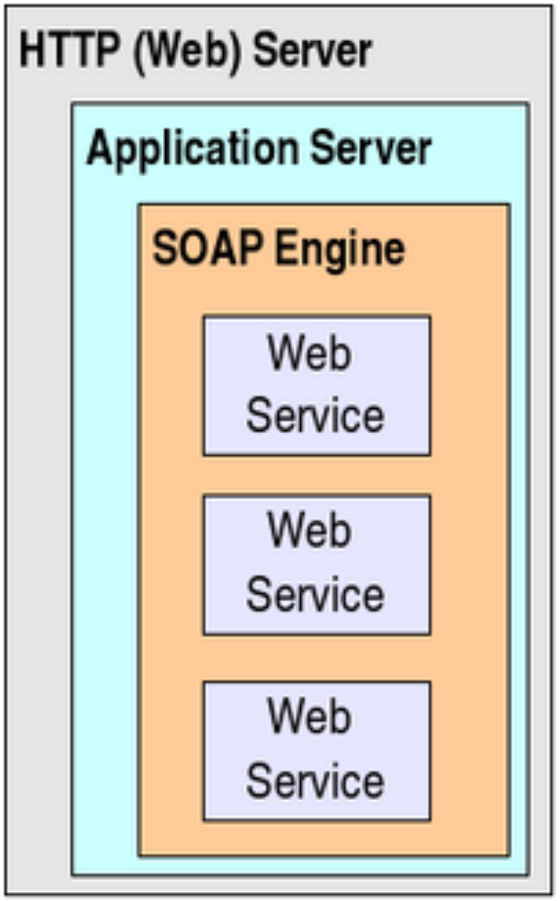
Service List		
Business Name:	xmethods.net	
Description:		
Service Name	**Business Key**	**Service Key**
Currency Exchange Rate	D1387DB1-CA06-24F8-46C4-86B5D895CA26	6FD77EF6-E7D6-6FF6-1E41-EBC80107D7B5
Domain Name Checker	D1387DB1-CA06-24F8-46C4-86B5D895CA26	6CD4E4D4-00AB-16D5-133E-9D713F954DEB
Barnes and Noble Price Quote	D1387DB1-CA06-24F8-46C4-86B5D895CA26	6EC30F31-A032-2435-64EE-95CCD8DC8D63
XMethods Filesystem	D1387DB1-CA06-24F8-46C4-86B5D895CA26	203594B5-5BA0-3706-B10F-F7F25AE891B8
FedEx Tracker	D1387DB1-CA06-24F8-46C4-86B5D895CA26	C7AEC037-E685-2374-233F-5F6A99091562
XMethods Query Service	D1387DB1-CA06-24F8-46C4-86B5D895CA26	06FAF333-31EF-0C44-4ABF-42E121276785
BabelFish	D1387DB1-CA06-24F8-46C4-86B5D895CA26	F1354B94-EA88-F13E-5F8C-61E3EC44EBF2
eBay Price Watcher	D1387DB1-CA06-24F8-46C4-86B5D895CA26	28F3BED2-E798-F55D-47F8-D8E29B274A5D
Delayed Stock Quote	D1387DB1-CA06-24F8-46C4-86B5D895CA26	A24342C3-1F7B-7014-C5BD-8B496182349E
California Traffic Conditions	D1387DB1-CA06-24F8-46C4-86B5D895CA26	E036BB34-E3AA-C7AF-81FC-B719D765C979
Dummy replacement for Weather - Temperature	D1387DB1-CA06-24F8-46C4-86B5D895CA26	4DDE5597-616A-B361-7DB6-75128C25A1F7
XSpace	D1387DB1-CA06-24F8-46C4-86B5D895CA26	54C31396-5823-88C0-CA3A-7F931C2AA6EE
test22	D1387DB1-CA06-24F8-46C4-86B5D895CA26	EA15B022-B653-7691-0562-ABFE23867350

Comparação com o Modelo de Objectos

- O modelo de invocação de Web Services é muito semelhante ao do RMI
  - Mas existem diferenças importantes
- Referências
  - Uma referência para um objecto remoto designa a instância de uma classe que implementa uma interface
    - Um objecto pode ser instanciado remotamente
  - Uma referência para um WS designa uma interface e o WS não pode ser instanciado
  - Não podem ser passadas referências para objectos na invocação de um WS
- Estado
  - Um objecto remoto pode guardar estado entre a execução de métodos
  - Um WS não guarda estado normalmente, precisa de utilizar extensões para o fazer (WSRF)
- Execução
  - Um objecto remoto pode ser executado em modo stand-alone desde que contenha uma classe que implemente um servidor
  - Um Web Service tem de ser executado no contexto de um *container*

Contexto de Execução de um WS

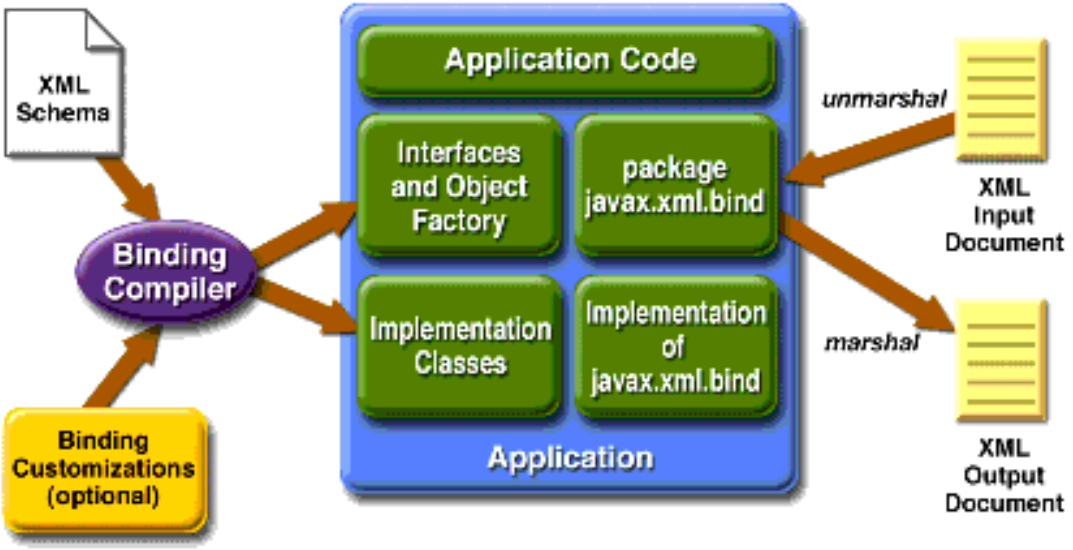
- Um Web Service é geralmente executado no contexto de um Servidor aplicacional
  - Pode correr dentro ou fora do servidor Web
- Exemplos:
  - Servlet Engine: Tomcat
  - App Server: JBOSS, Sun Java Application Server
  - Apache Server: módulo php
  - Microsoft .NET



Java e Web Services

- Embora o XML seja um formato literal, a programação das interfaces e do acesso a Web Services não é fácil de fazer directamente
- A linguagem Java fornece uma série de APIs que permitem:
  - Implementar Web Services e exportar as suas interfaces, através da geração automática de WSDL
  - Invocar directamente Web Services através do conhecimento do respectivo WSDL através da geração de
    - *Proxy* cliente e *skeleton* servidor
    - Mensagens em SOAP para envio do pedido e descodificação da resposta
- JAX-RPC:
  - Extensão do RMI que permite criar e invocar Web Services
  - Utiliza SOAP e XML based RPC
  - Suplantado pelo JAX-WS
- JAX-WS:
  - Nova plataforma integrada baseada em JEE 5 e no projecto GlassFish (Open Source)
  - Permite a descrição e *binding* de Web Services na própria linguagem Java
  - Cria uma correspondência directa entre documentos XML e objectos Java através da utilização do **JAXB** (Java Architecture for XML Binding)

JAXB: Java Architecture for XML Binding



- Framework genérica que permite a uma aplicação realizar o *parsing* de XML obedecendo a um *schema* determinado
- O *binding compiler* gera as classes Java a partir do schema ao qual obedece o documento, e estas são invocadas a partir da aplicação
- Através do package javax.xml.bind, o documento é convertido numa árvore de objectos com o conteúdo proveniente dos elementos XML
- Os dados podem ser modificados pela aplicação validados de acordo com o *schema*, e de novo serializados para um novo documento XML
- A fonte de XML pode ser qualquer, como por exemplo uma mensagem SOAP ou um ficheiro WSDL

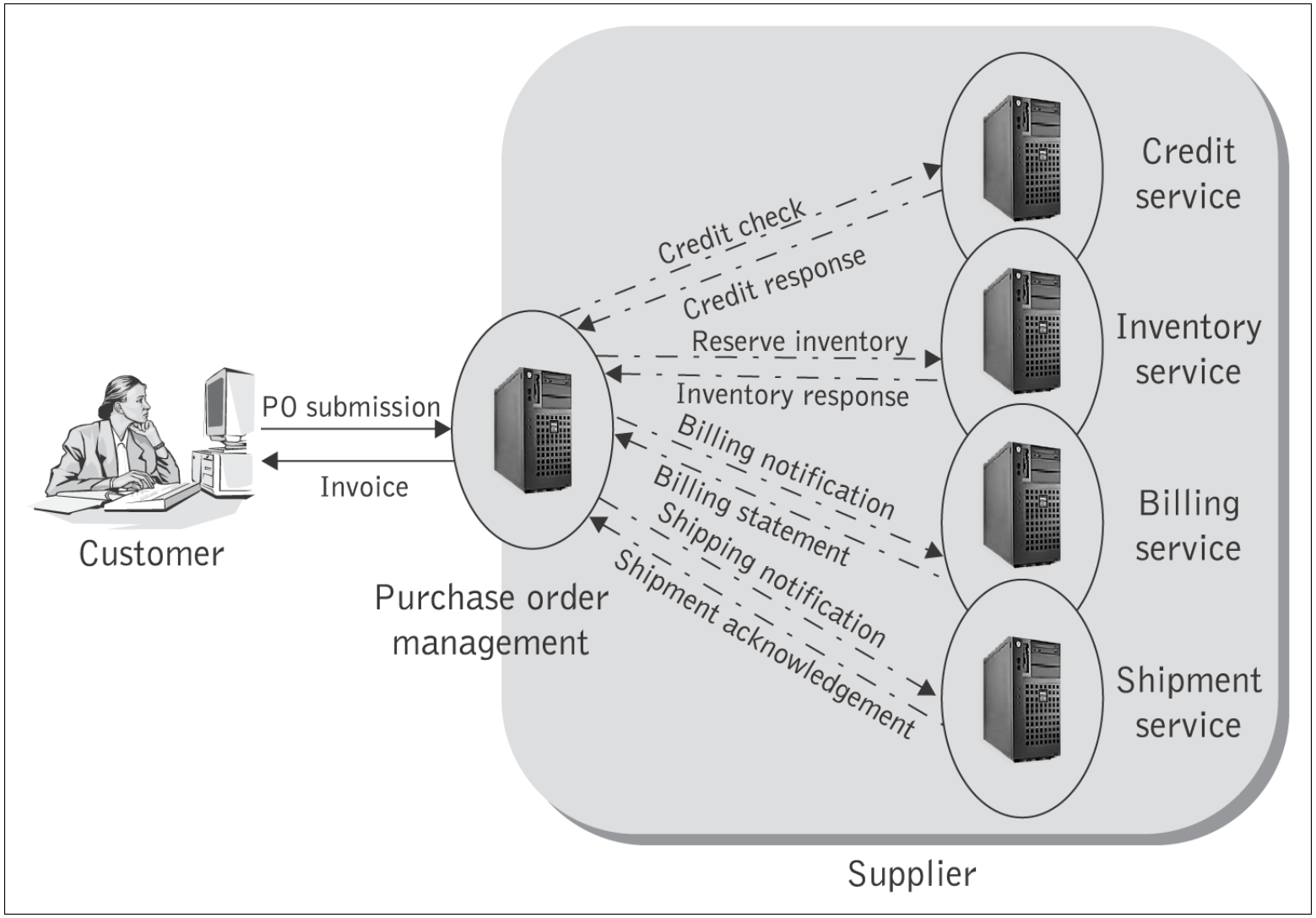
JAX-WS: Java Web Services

- Java API for XML WebService
- Utiliza anotações inseridas no código para a definição das interfaces no servidor e das referências no cliente
- O gerador de interfaces (*wsgen*) utiliza as anotações para gerar o código de invocação do serviço no cliente e no servidor assim como a descrição do mesmo em WSDL
- @WebService()
  - Indica que a classe anotada implementa um Web Service
- @WebMethod
  - Indica que o método anotado é a implementação de uma das funções do web Service
- @WebServiceRef
  - Permite a um cliente referenciar a descrição do serviço em WSDL

Utilização de Integrated Development Environments

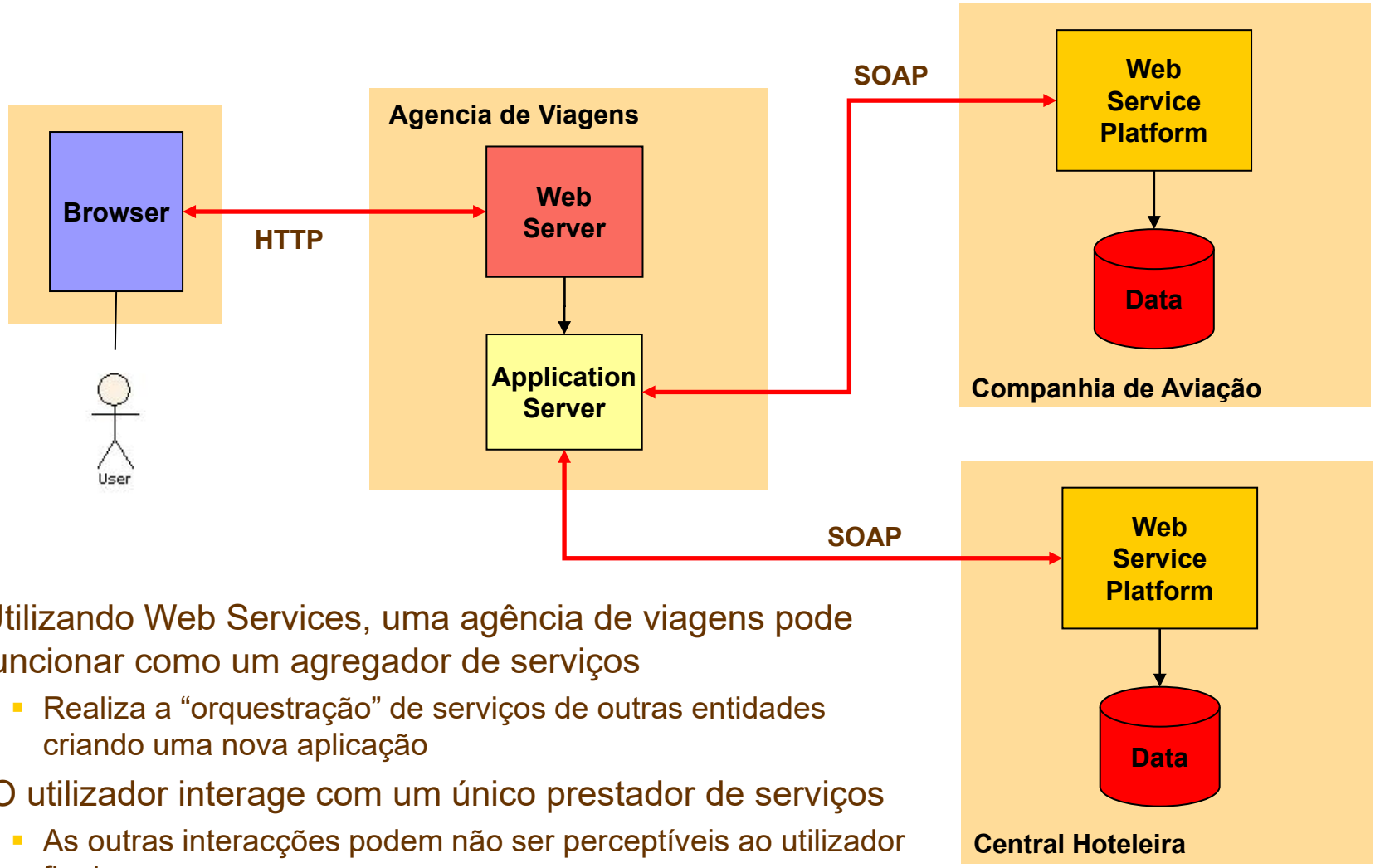
- A implementação de Web Services é muito facilitada pela utilização de IDEs - Integrated Development Environments
- Integram ainda mais os aspectos programáticos da linguagem com as funcionalidades fornecidas pelos geradores de interfaces
- Todos os aspectos ligados à utilização de XML são escondidos pela ferramenta
- Exemplos
  - Eclipse, NetBeans, …
- Tutorials
  - Java EE 5 e Web Services
    - http://java.sun.com/javaee/5/docs/tutorial/doc/index.html
  - NetBeans e Web Services com Java EE 5
    - http://www.netbeans.org/kb/55/websvc-jax-ws.html

Aplicações de Web Services: Integração entre empresas distintas



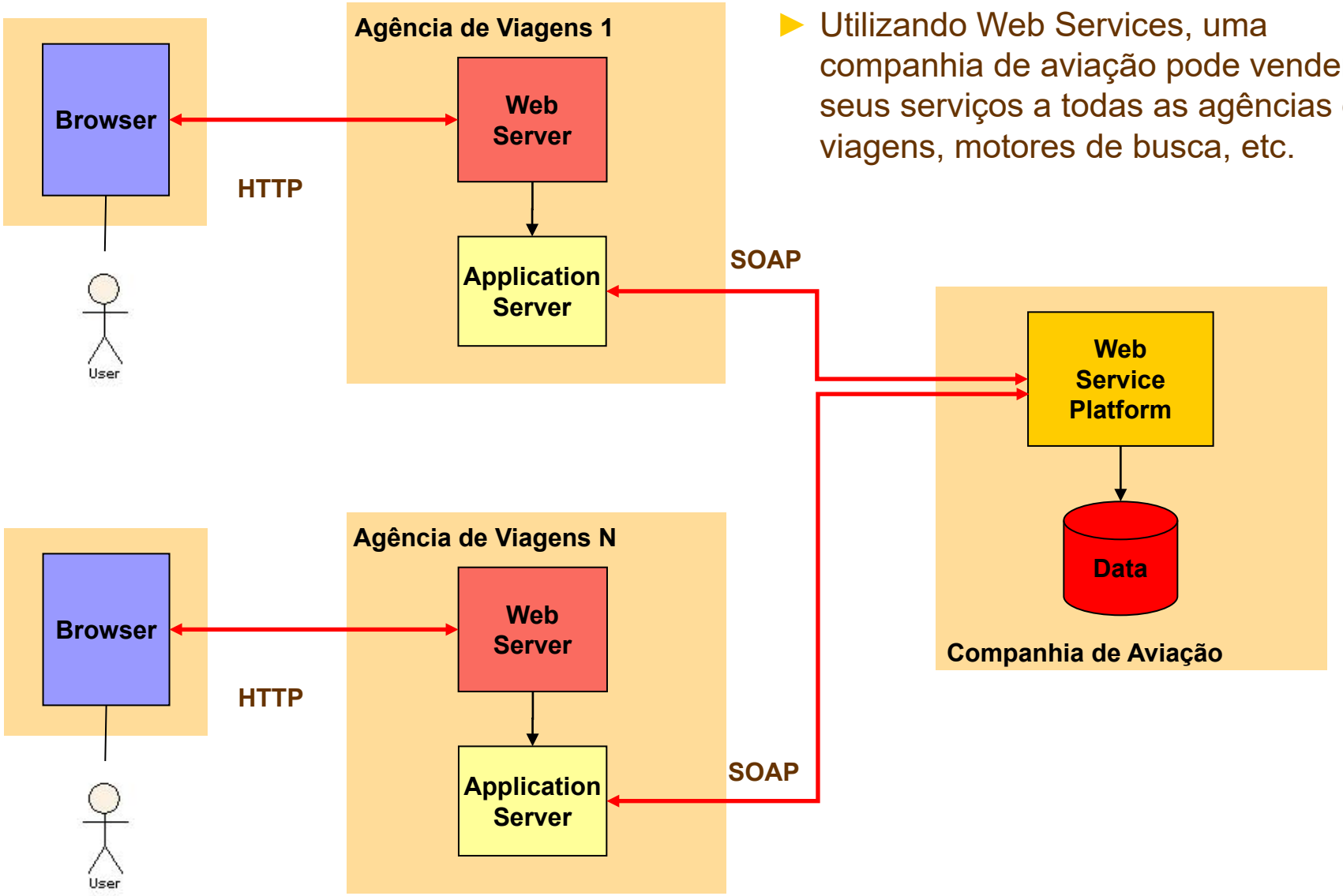
From: Michael P. Papazoglou, *Web Services*, 1st Edition, © Pearson Education Limited 2008

Aplicações de Web Services: Integração entre empresas distintas



- Utilizando Web Services, uma agência de viagens pode funcionar como um agregador de serviços
  - Realiza a "orquestração" de serviços de outras entidades criando uma nova aplicação
- O utilizador interage com um único prestador de serviços
  - As outras interacções podem não ser perceptíveis ao utilizador final

Aplicações de Web Services: Integração entre empresas distintas



- Utilizando Web Services, uma companhia de aviação pode vender os seus serviços a todas as agências de viagens, motores de busca, etc.

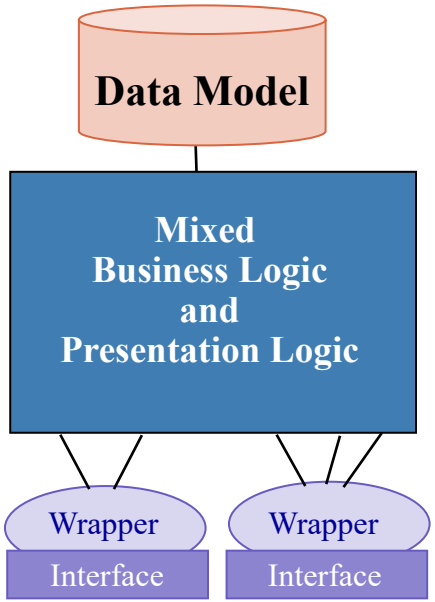
Aplicações de Web Services: SOA - Service Oriented Architecture

- O termo SOA designa um conceito arquitectural que consiste na implementação de funcionalidades através do conceito de serviço
  - Baseado em componentes independentes e facilmente agregáveis
- Um serviço é caracterizado por
  - Uma forma de nomeação e localização
  - Uma interface que fornece um conjunto de métodos específicos
  - Uma semântica de invocação que define os aspectos dinâmicos
- O conceito SOA é agora utilizado para permitir novas formas de interacção na Web utilizando protocolos existentes
  - Aumentar a sofisticação da interacção entre Clientes e Servidores
  - Definir interacções e colaborações directas entre servidores de forma a permitir a agregação de funcionalidades
  - Permitir a descoberta e utilização de novos serviços de forma dinâmica e imediata

Aplicações de Web Services: SOA empresarial

Wrapping

Use of screen scraping to package "pseudo services"



Pros:

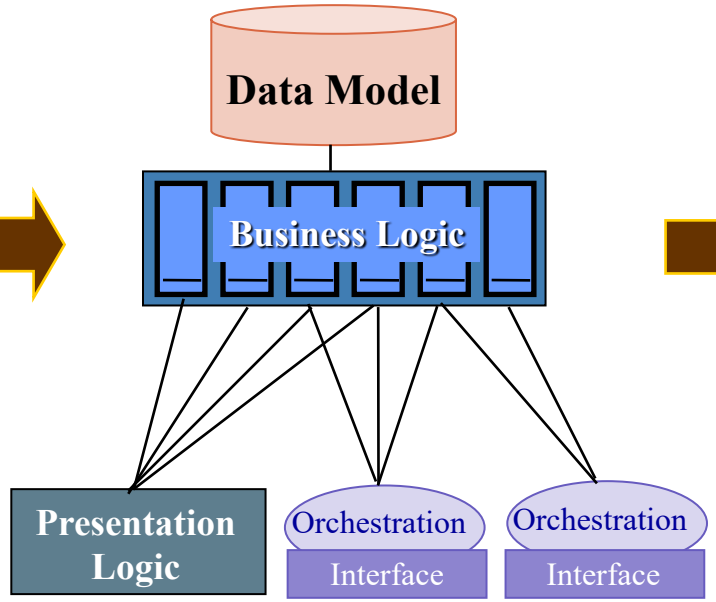
- Non-invasiveness
- Low cost/fast

Cons:

- Suboptimal granularity
- Hard to maintain

Re-engineering

Business logic is modularized and separated from presentation



Pros:

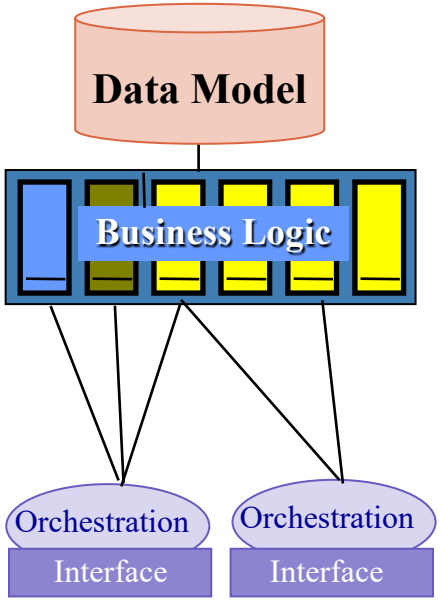
- Easier to maintain
- Better performance/scalability

Cons:

- Invasive/high cost
- Granularity might be suboptimal

Replacement

Business logic of services is redesigned from scratch or replaced with packages



Pros:

- Optimal granularity/reuse
- Enables technology change

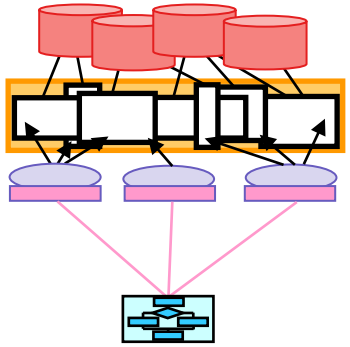
Cons:

- Risk is higher
- High cost

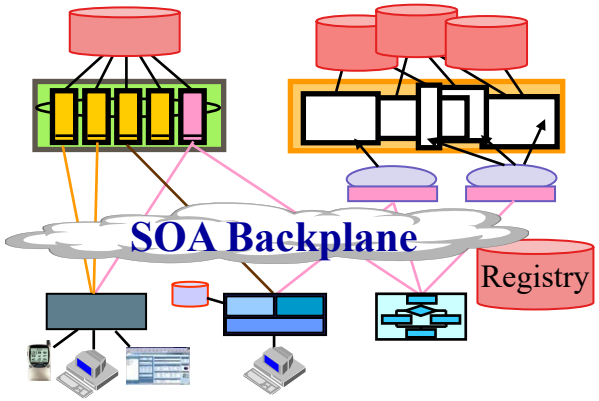
Fonte: Gartner

Aplicações de Web Services: SOA empresarial

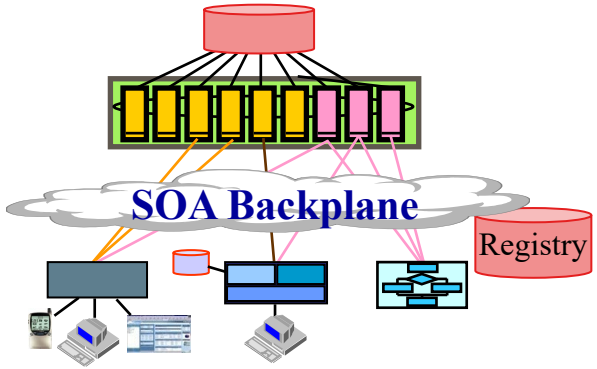
Wrapped-SOA



Hybrid-SOA



Full-SOA



Fonte: Gartner

Perguntas a que devo ser capaz de responder

- Para que serve o UDDI ?
- Existem elementos semelhantes ao UDDI no Sun RPC e no Java RMI ? Se sim, quais ?
- Indique um exemplo de binding de Web Services com SOAP a um protocolo de transporte
- Como é que um IDE (Integrated Development Environment) pode facilitar o desenvolvimento de aplicações com Web Services ?

Fim do Capítulo V

Resumo dos conhecimentos adquiridos:

- Modelo de Execução Web Services (WS)
  - Register, Discover, Invoke
- O protocolo SOAP
  - Envelope, Headers e Body
- Linguagem de Definição de Serviços
  - Estrutura e exemplos de WSDL
- Serviço de Directório de WS
  - UDDI
- Comparação dos modelos de invocação WS e RMI
- Integração com Java
  - JAX-WS e Utilização de um IDE
- Aplicações de Web Services
  - Integração entre empresas distintas
  - SOA