Computação Distribuída

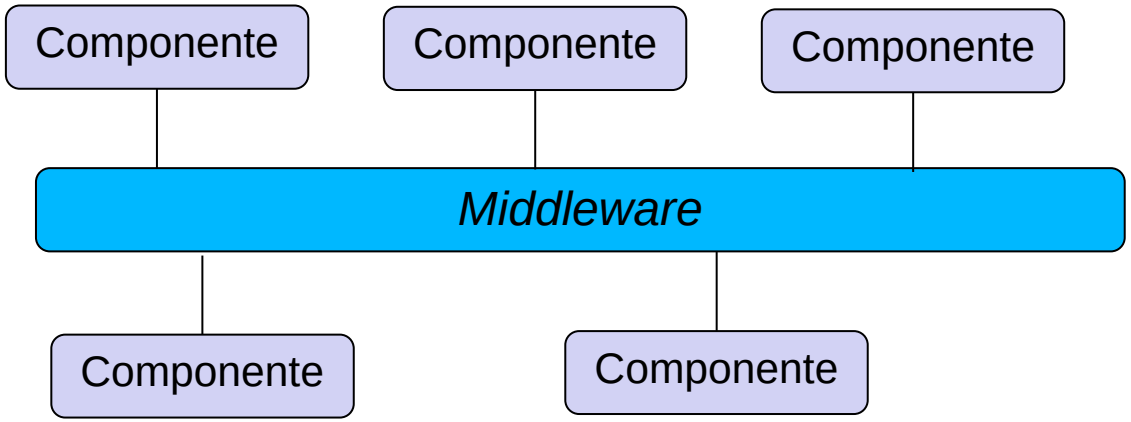
Odorico Machado Mendizabal

Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE

Arquitetura Orientada a Serviços

Motivação

- Surgimento de modelos de RPC e RMI
  - Maior abstração no modelo de comunicação em programação distribuída
  - Facilidade em reutilizar implementação já existente (invocação a métodos remotos, objetos distribuídos)



- No entanto, ainda há certa dependência à determinadas tecnologias
  - Projeto de Sistema Distribuído deve prever em que plataforma de *Middleware* a solução será empregada (ex. RPC, Java RMI, CORBA)
- Não são soluções amplamente empregadas em ambientes empresariais

Uso de Serviços Web

O que é um serviço?

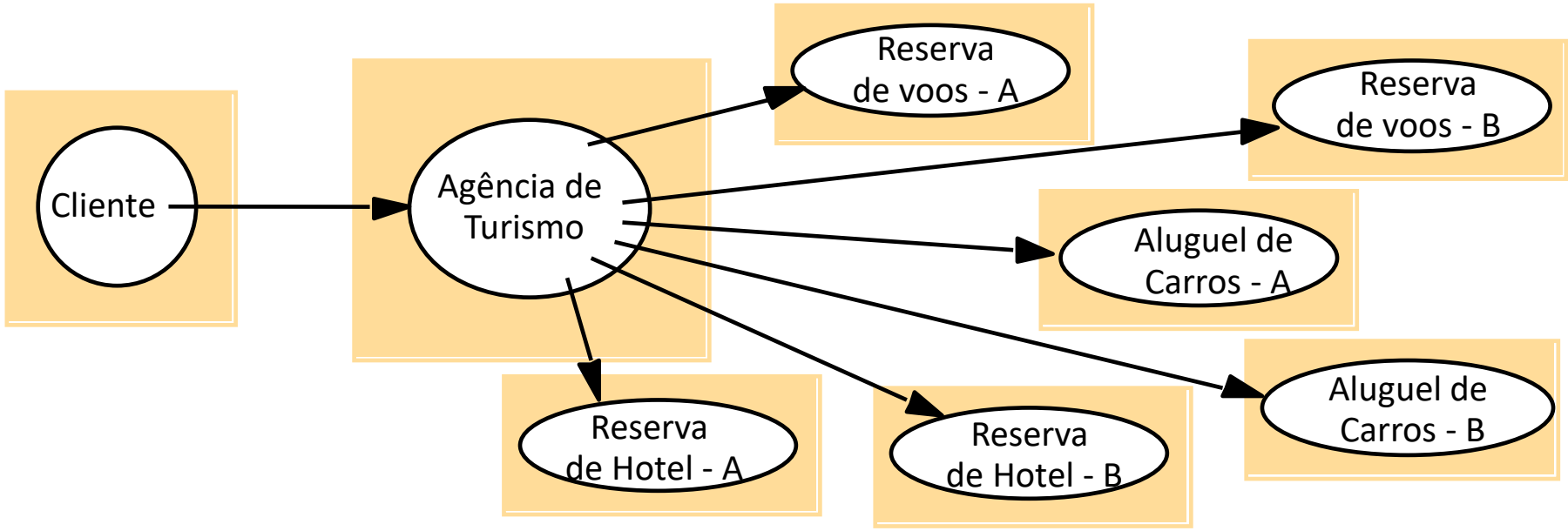
Em sistemas computacionais ..

.. Um serviço é um componente de software ***fracamente acoplado*** *que* ***implementa uma funcionalidade específica*** *e é* ***exposto*** *para o* ***(re)uso*** *de outros processos*

Orientação à Serviços

- Serviço encapsula a complexidade das regras de negócio e oferece uma interface bem definida para a sua invocação
- Acesso remoto à serviços (Arquitetura Orientadas à Serviços – SOA (*Service-Oriented Architecture*)
- Interoperabilidade e heterogeneidade

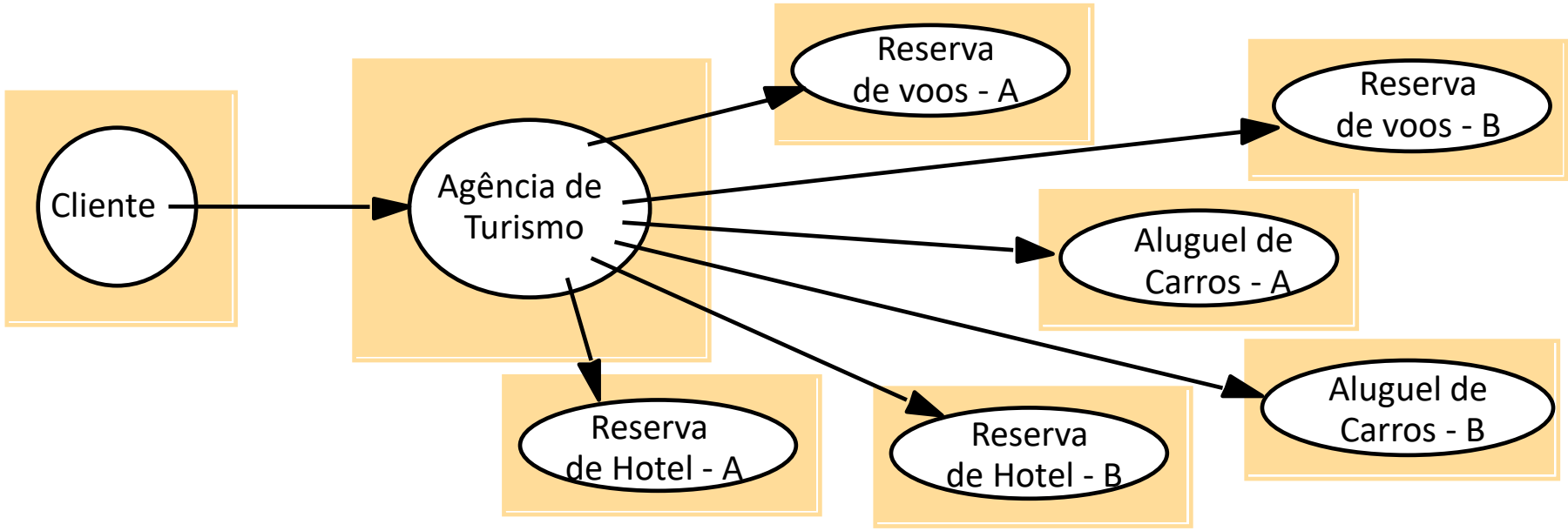
Exemplo de Aplicação: Agência de Turismo



A agência de turismo deve usufruir de serviços de várias outras empresas de viagem: reserva de hotel, companhia viagem aérea, aluguel de carro, etc.

Qual a melhor maneira de integrar estes vários serviços em um único sistema?

Exemplo de Aplicação: Agência de Turismo



Alguns desafios:

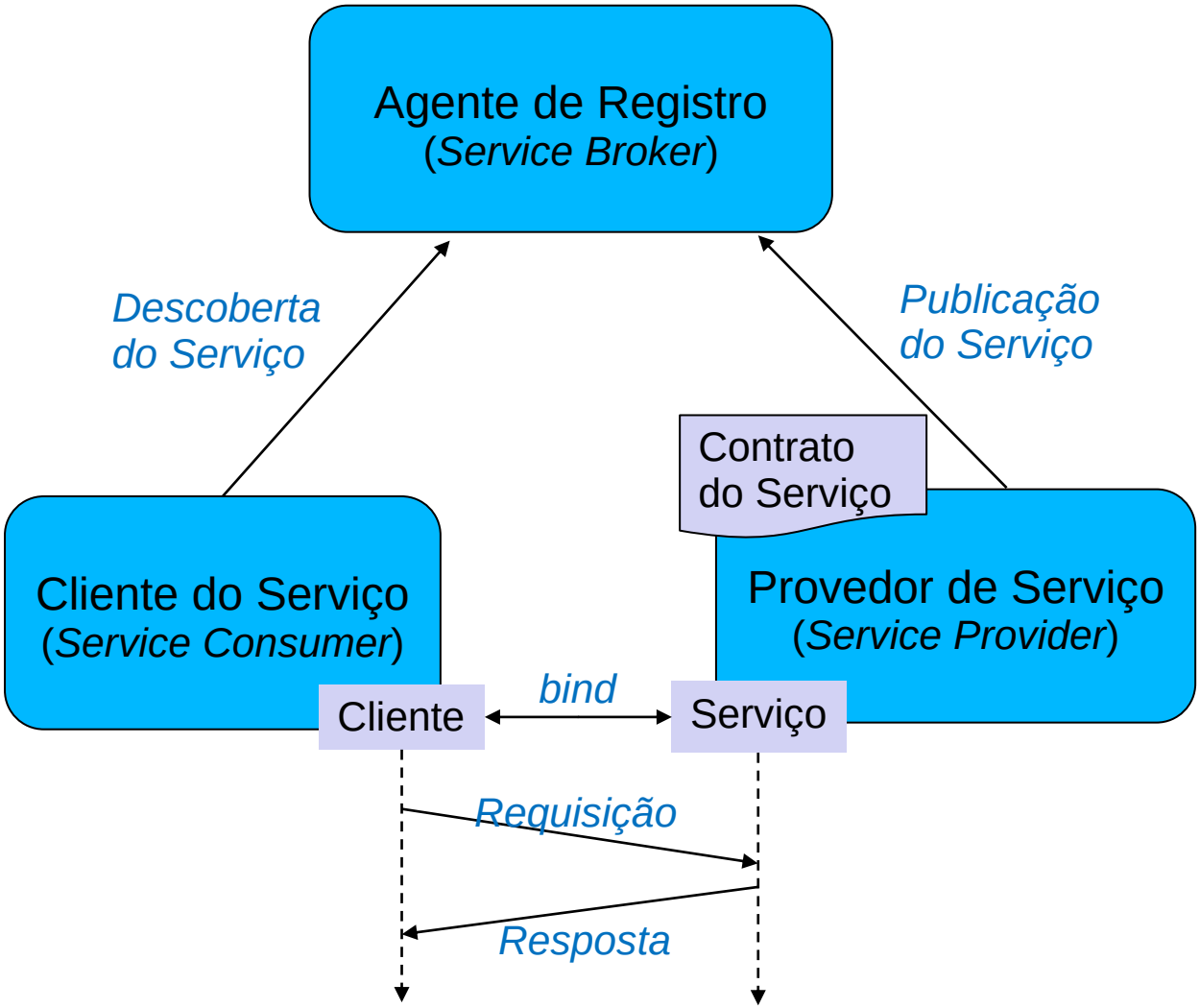
- Como integrar sistemas legados com sistemas atuais?
- Como localizar os serviços na Web?
- Como obter a interface dos serviços? (quais parâmetros devem ser enviados)
- Como acessar serviços em uma WAN? Como passar por Firewalls?

Arquitetura Orientada a Serviços (SOA) é um estilo arquitetural no qual aplicações são montadas a partir de serviços fracamente acoplados sobre uma rede. Serviços são interoperáveis e comunicam-se usando padrões abertos.

Serviços Web

Serviços podem ser chamados pela Web
Comunicação baseada em protocolos abertos, amplamente divulgados e consensuais:
HTTP, XML, WSDL, SOAP, REST, UDDI

Arquiteturas Orientadas a Serviço (SOA – *Service Oriented Architecture*)



1. Provedor (*Provider*) registra serviço no *Service Broker*
2. Consumidor (*Consumer*) obtém informação sobre serviço (*Find*) através do repositório de serviços (*Service Broker*)
3. Consumidor e Provedor estabelecem uma conexão (*Bind*) para possibilitar a troca de mensagens
4. Consumidor envia uma requisição a um serviço (*service request*)
5. Provedor executa o serviço e responde a requisição diretamente para o consumidor

Arquiteturas Orientadas a Serviço (SOA – *Service Oriented Architecture*)

- Serviços são disponibilizados via Internet
- Tipicamente implantados em ambientes de computação em nuvem
  - Serviços sob demanda
  - Elasticidade e escalabilidade

- Clientes (consumidores do serviço) podem ser desenvolvidos em diferentes tecnologias e dispositivos (*frontend*)
  - Web, desktop, smartphone, sistemas embarcados, etc.

Padrões e protocolos Web

- Arquitetura Web
  - Protocolos de rede (camadas)
    - Transporte (TCP), Rede (IP)
  - Transferência de recursos na Web
    - HTTP
    - Outros protocolos de comunicação podem ser usados (menos usual). Ex. SMTP
- Representação de Dados
  - XML
- Interface dos Serviços
  - WSDL
- Serviços
  - SOAP, REST
- Publicação e descoberta de Serviços
  - UDDI

Formato XML – *eXtensible Markup Language*

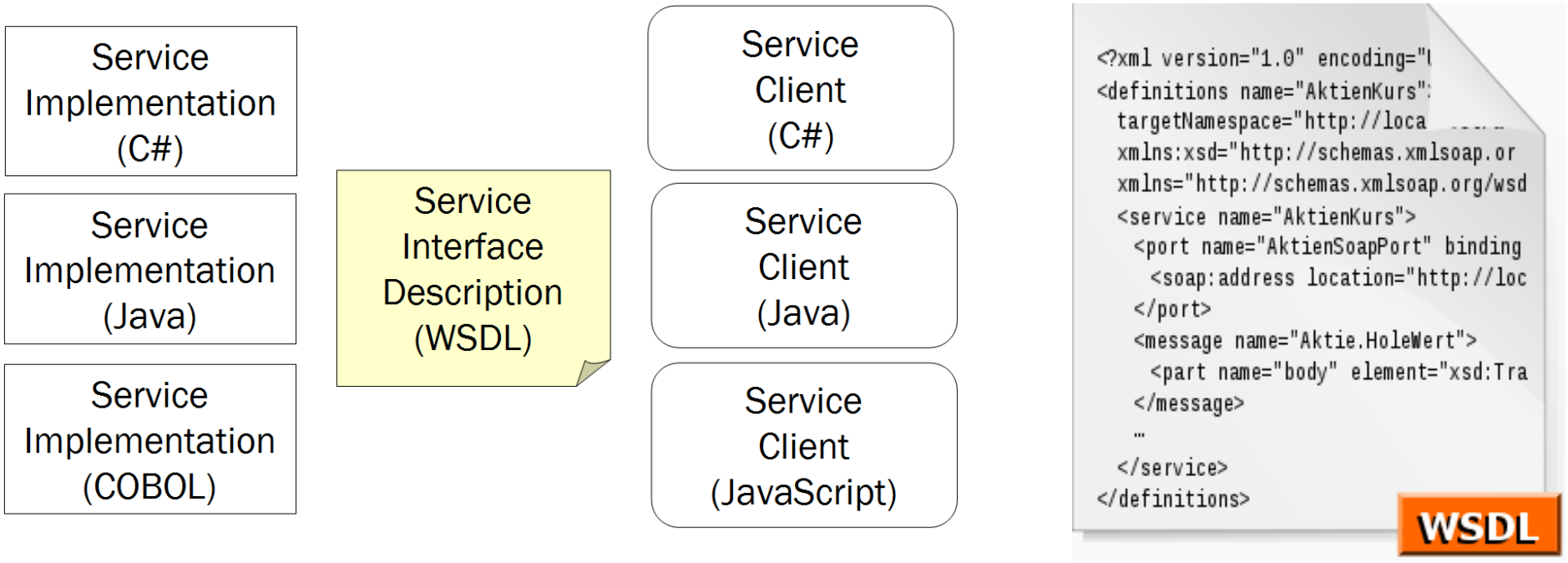
- Padrão do W3C (*World Wide Web Consortium*) para representação de dados
- Linguagem de marcação
  - Utilização de *tags* (rótulos) para criação de metadados
  - Facilita a interoperabilidade entre aplicações executadas em diferentes plataformas
  - Fácil leitura por humanos
  - Pode ser processado em qualquer linguagem por manipulação do texto

```
<?xml = version "1.0">
<article>
    <title> Prudent Engineering Practice for Cryptographic Protocols</title>
    <author><name>M. Abadi</name></author>
    <author><name>R. Needham</name></author>
    <journal>
        <jname>IEEE Transactions on Software Engineering</jname>
        <volume>22</volume>
        <number>12</number>
        <month>January</month>
        <pages>6 – 15</pages>
        <year>1996</year>
    </journal>
</article>
```

Linguagem de Descrição de Serviços Web

WSDL (*Web Services Description Language*)

- Possibilita uma representação de uma interface para o serviço, independente de linguagem de programação
- Definição das operações oferecidas por um serviço e seus parâmetros
- Descrição dos tipos usando XML



WSDL – *Web Services Description Language*

Principais elementos do WSDL:

- `<definitions>` elemento raiz
- `<type>` define os tipos de dados utilizados pelo serviço (pode referenciar um esquema XSD)
- `<message>` especifica as mensagens para a comunicação com o serviço Web
- `<portType>` define um conjunto de operações que são executadas por um serviço
- `<binding>` associa ao serviço um protocolo de transporte e o estilo de comunicação adotado
- `<service>` especifica o endereço de rede para acesso ao serviço

WSDL – Exemplo

```
<definitions name="HelloService"
     targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
     xmlns="http://schemas.xmlsoap.org/wsdl/"  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">

     <message name="SayHelloRequest">
          <part name="firstName" type="xsd:string"/>
     </message>

     <message name="SayHelloResponse">
          <part name="greeting" type="xsd:string"/>
     </message>

     <portType name="Hello_PortType">
          <operation name="sayHello">
               <input message="tns:SayHelloRequest"/>
               <output message="tns:SayHelloResponse"/>
          </operation>
     </portType>

     <binding name="Hello_Binding" type="tns:Hello_PortType">
     <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>   <operation
name="sayHello">
          <soap:operation soapAction="sayHello"/>
          <input>
               <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     namespace="urn:examples:helloservice" use="encoded"/>
          </input>
          <output>
               ....
```

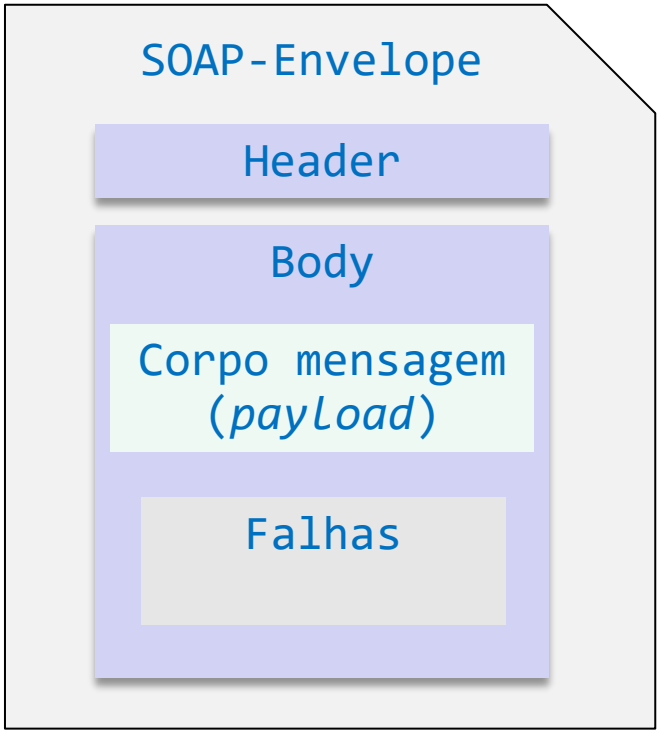
SOAP (~~*Simple Object Access Protocol*~~)

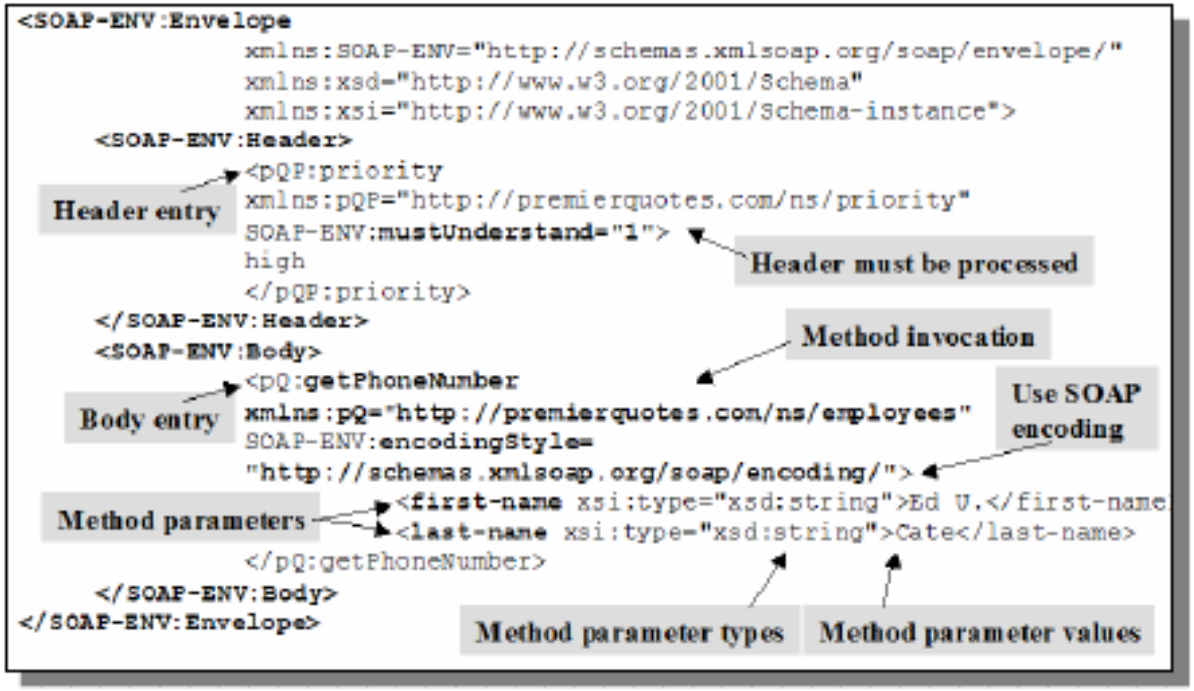
- Protocolo definido pelo W3C
- Define o formato das mensagens trocadas entre serviços Web
- Formato independente de plataforma ou linguagem de programação
    - Uso de XML e protocolos de transporte usuais da Web
- Protocolo HTTP(s) é comumente utilizado e o acesso à porta 80 é autorizado por firewalls
    - Simples adoção e configuração de infraestruturas na Internet

SOAP

Elementos das mensagens SOAP:

- Envelope,
- Cabeçalhos(*Headers*),
- Corpo da mensagem (*Body*)





SOAP – Básico

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <SOAP-ENV:Body xmlns:m="http://www.xyz.org/quotations">
       <m:GetQuotation>
           <m:QuotationsName>Geek</m:QuotationsName>
      </m:GetQuotation>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requisição

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <SOAP-ENV:Body xmlns:m="http://www.xyz.org/quotation">
        <m:GetQuotationResponse>
            <m:Quotation> There are 10 types of people in the world:
            Those who understand binary, and those who don't
            </m:Quotation>
        </m:GetQuotationResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Resposta

UDDI (*Universal Description, Discovery and Integration*)

- Serviço utilizado para registro e localização de serviços Web
  - Criado por Microsoft, IBM e Ariba
- Armazena as especificações WSDL dos provedores de serviço
  - Permite consultas e localização de serviços de interesse dos clientes
    - Páginas brancas (endereço para contato do provedor de serviço)
    - páginas amarelas (classificam provedores por categoria, de acordo com o tipo de serviço)
    - páginas verdes (disponibilizam informações técnicas sobre o serviço)

UDDI (*Universal Description, Discovery and Integration*)

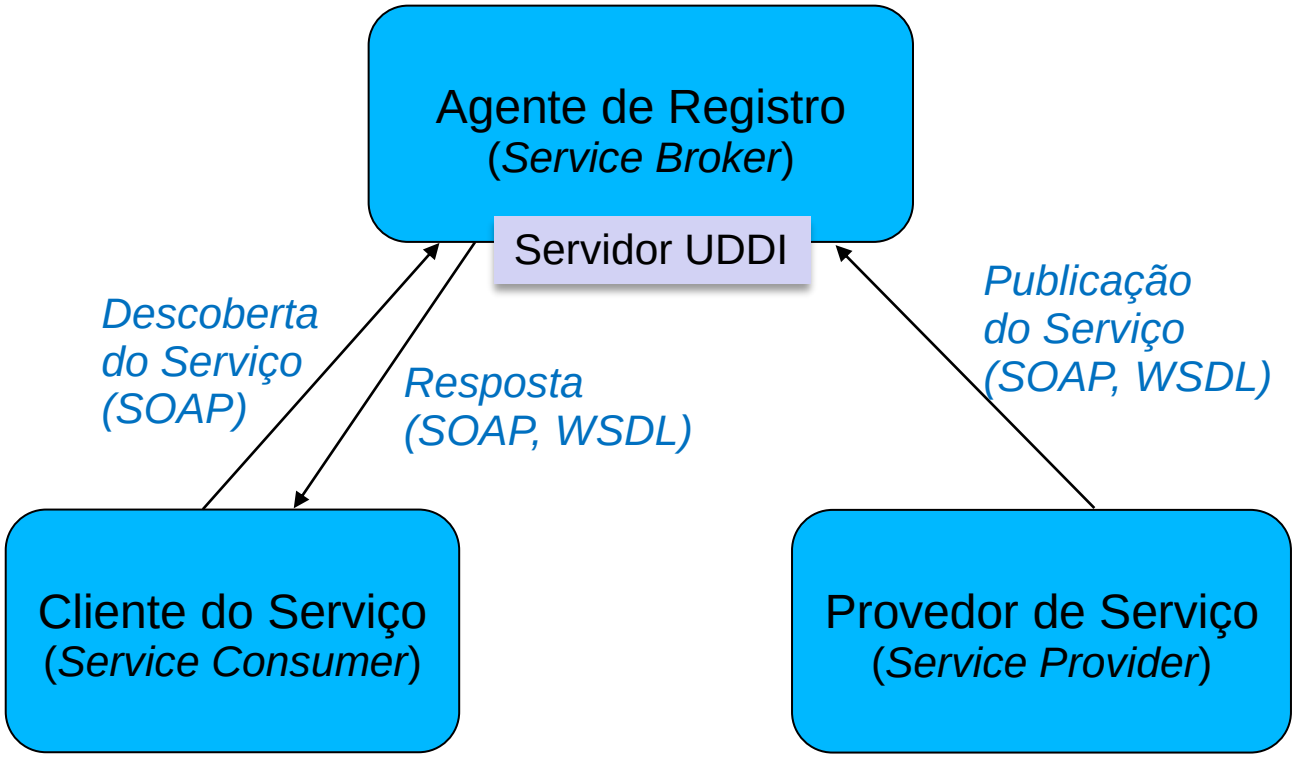
- Alguns exemplos de invocação disponíveis na API do UDDI:

Consulta:

- `find_binding`
- `find_business`
- `find_service`
- `get_businessDetail`
- `get_serviceDetail`

Publicação:

- `save_service`
- `save_binding`
- `delete_binding`
- `delete_service`



REST (*REpresentional State Transfer*)

- Padrão proposto por Roy Fielding (tese de doutorado)
  - Padrão mais informal que SOAP

- Visa simplicidade e desempenho
  - Acesso através de URLs (método HTTP sem necessidade de um envelope SOAP)
  - Recurso pode ser representado com formato mais leve que XML, ex. JSON
  - Não precisa armazenar estado no servidor

REST (*REpresentional State Transfer*)

- Operações sobre recursos
  - Operações CRUD (**C**reate, **R**ead, **U**pdate e **D**elete) a partir de métodos HTTP

  Create: POST ou PUT (cria um recurso no servidor)
  Read: GET (obtém recurso ou uma coleção)
  Update: PUT (modifica um recurso)
  Delete: DELETE (remove um recurso ou coleção)

- Parâmetros são passados na URL ou no corpo das mensagens

Exemplos:

```
GET <url>/catalogo/acao
GET <url>/catalogo/acao/<id>
```

(1) Retorna uma coleção de filmes de ação
(2) Retorna um filme específico (com identificador id) da coleção de filmes de ação

JSON (*JavaScript Object Notation*)

- Formato baseado em marcações, porém mais enxuto que XML
  - Aproximadamente 30% menor que XML
  - *Parsing* mais leve e mais rápido

- Representação de objetos no estilo chave-valor:
  Exemplo:

```
{“filme”: “Across the Universe”,
 “gênero”: [“musical”, “romance”],
 “ano”: 2007}
```

REST (*REpresentional State Transfer*)

Publicação da API de serviços

- WADL (*Web Application Description Language*)
  - Derivada de XML para descrição de serviços REST
  - Proposto pela Sun ao W3C
  - Parcialmente aceito

- *Open API Specification* (Swagger)
  - Proposta mais recente
  - Escrito em JSON ou YAML
  - Teve melhor aceitação que WADL, mas também não é um padrão definitivo
    - Usado por Linux Foundation, Google, Microsoft, IBM,...