SOA and Services Web

Walid GAALOUL
Département INFormatique
TELECOM SudParis
http://www-inf.it-sudparis.eu/cours/WebServices

References

- Web
  - http://www-inf.it-sudparis.eu/cours/WebServices/
  - Site de W3C (normes) : www.w3.org
  - Site de Zvon (tutoriel XML) : http://www.zvon.org/
- Books
  - Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju : Web Services: Concepts, Architecture and Applications, Springer-Verlag, New York, 2004
  - Jorge Cardoso and Amit P. Sheth : Semantic Web Services, Processes and Applications (Semantic Web and Beyond: Computing for Human Experience), Springer-Verlag, New York, 2006

Plan

- Origin of service orientation

- SOAP Web Services
  - Origins and definition
  - WSDL : Web Service description Language
  - SOAP : Simple Object Access Protocol
  - Axis
  - WCF
  - Standards WS-*

- RESTFull Web Services

Service orientation

Origin, Definition et Architecture

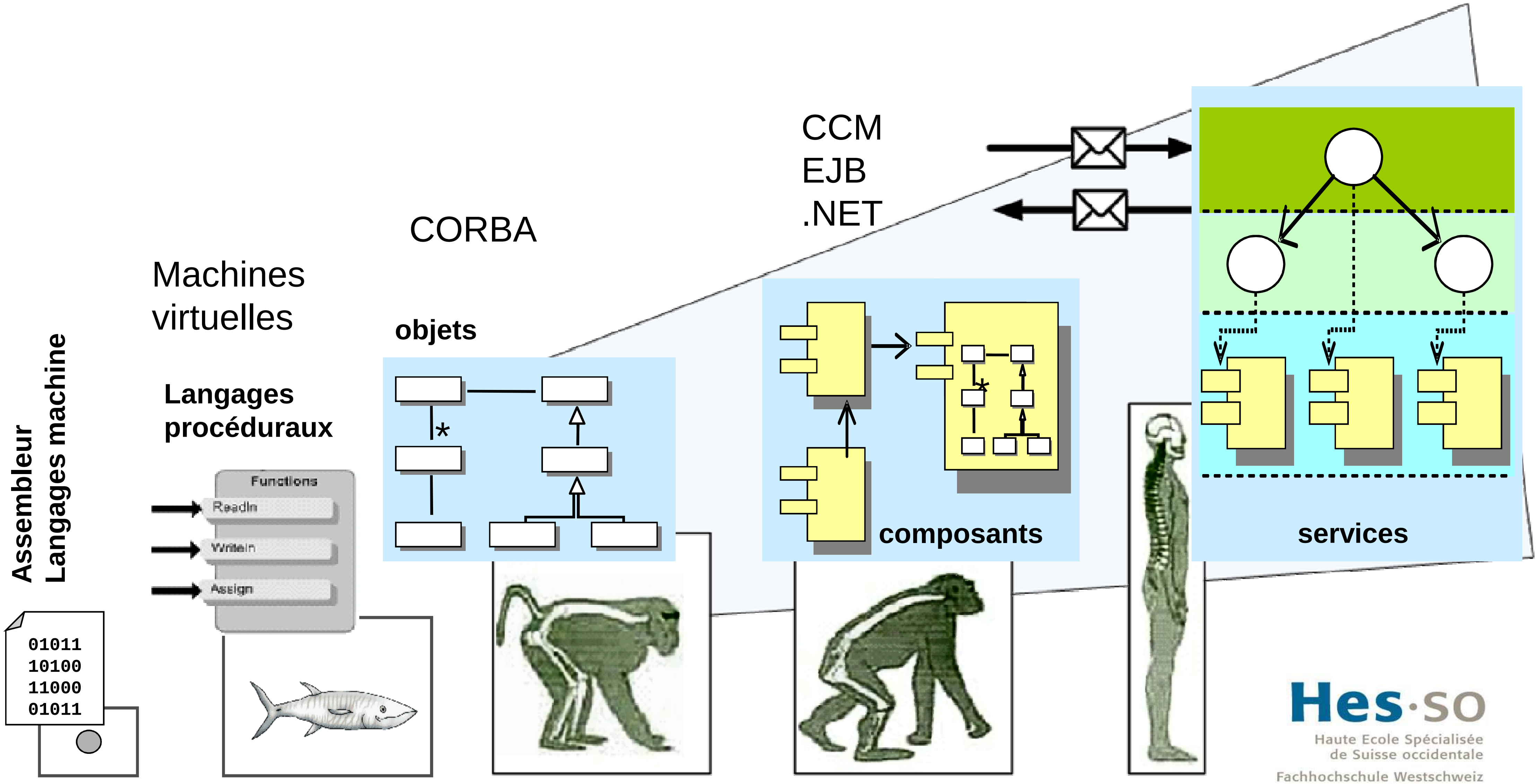
Origin (1/3)

- Structural programming
  - Procedures, Functions and Data
  - Monolithic, isolated server applications
- Object programming
  - Classes (fonctions and structured data grouped)
  - Depended on the programing language
  - Monolithic, isolated c/s applications
- Component programming
  - component (interfaces grouped)
  - Implementation agnostic
  - Depended on the compenent model and platform

Origin (2/3)

- SP, OOP, COP
  - Paradigms ?
    - They are about code not architecture
    - Architecture follows the programming model
- Needs?
  - Support heterogeneity of platforms
  - Access and manipulation of data from anywhere
- Services ?
  - Piece of software
    - Code, platform and location are irrelevant

Origin (3/3)

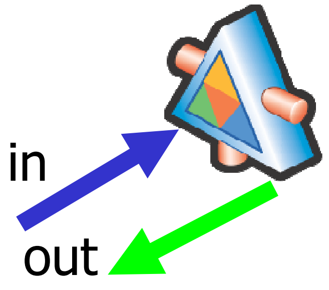


➢ ***Niveaux d'abstraction grandissant***

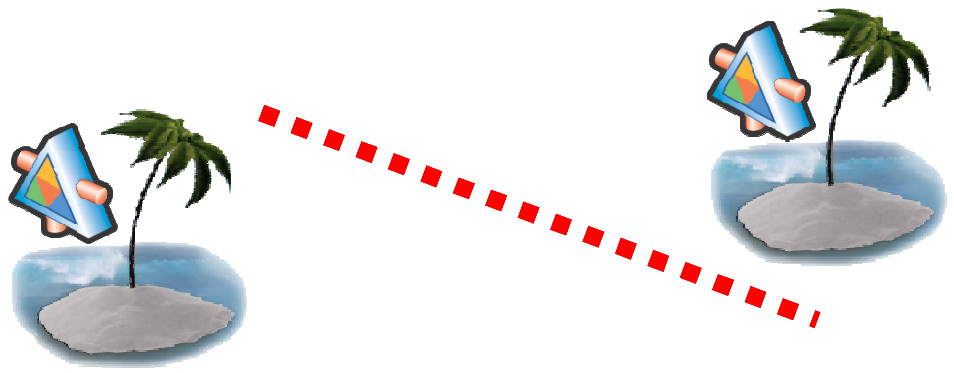
Definition

- Service is autonomous

- Service exposes <u>Contracts</u>



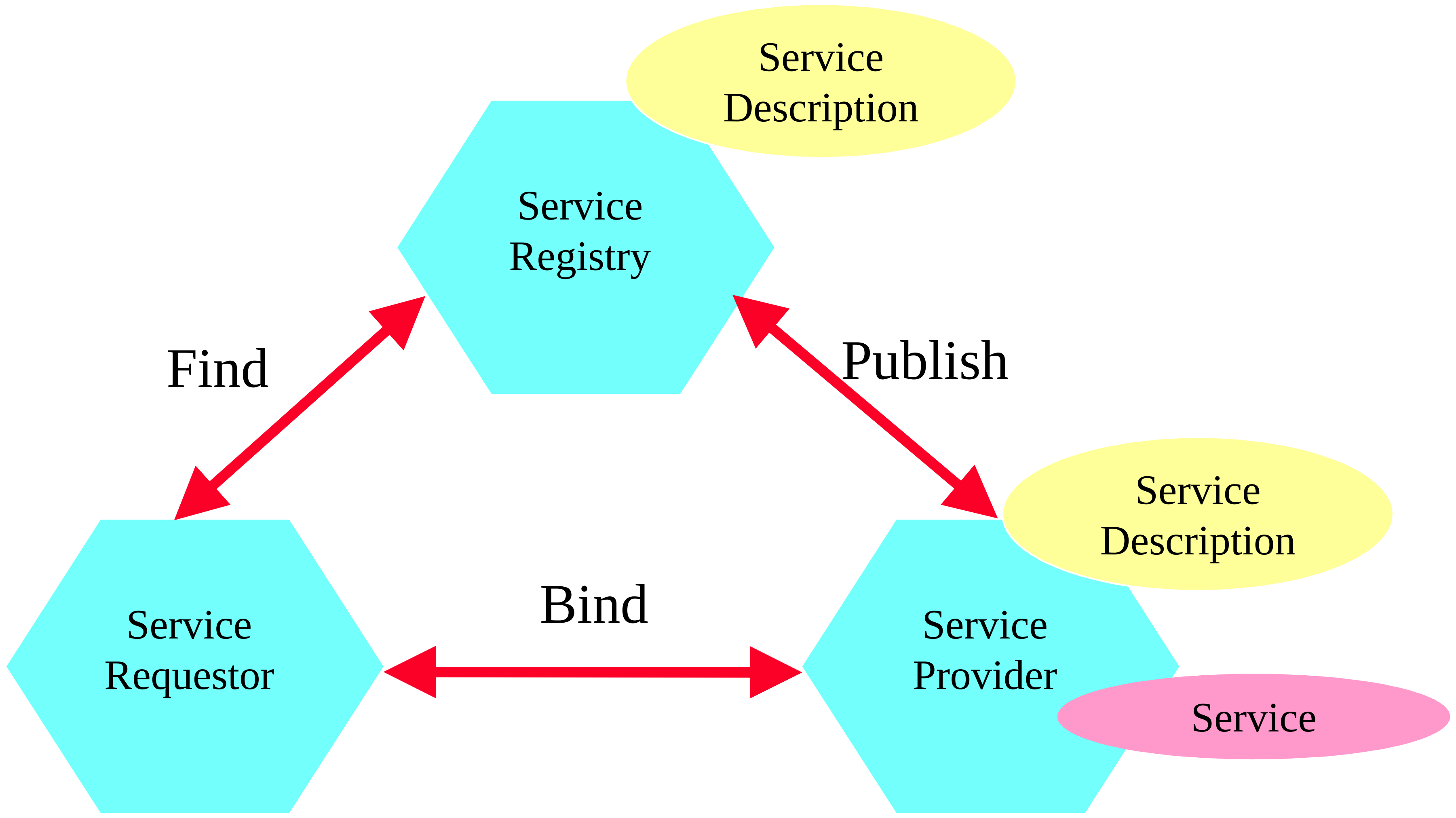
- Frontiers between services are explicit



- Services communicate using messages

Source A. Occello

Architecture



SOAP Web services

Origins et Definitions

SOAP WS: définition

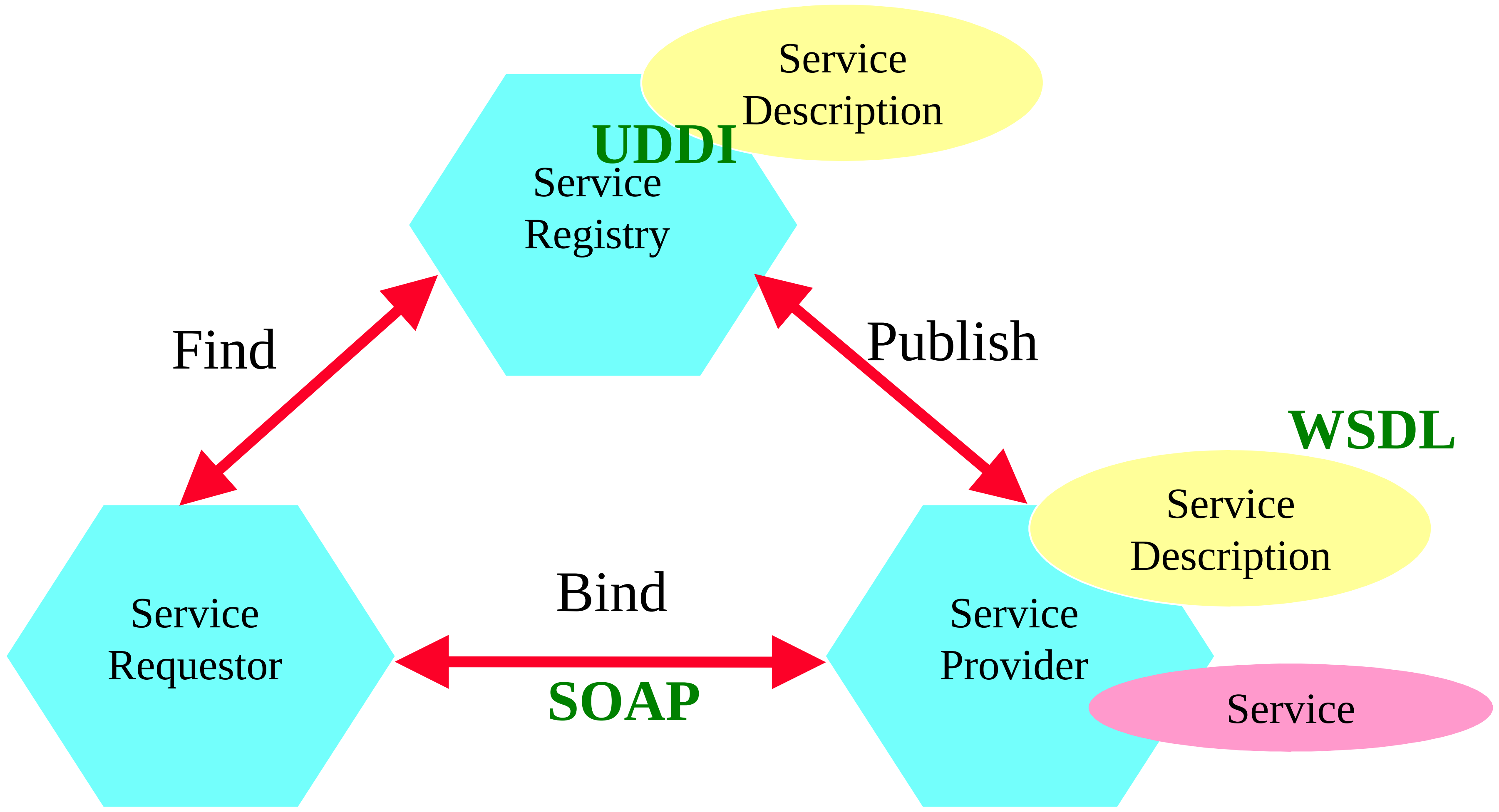
A Web service is a *software application* identified by a *URI*, whose *interfaces* and *binding*(1) are capable of being *defined, described and discovered* by XML artefacts and supports direct interactions with other software applications using *XML based messages* via *Internet-based protocols*. (W3C definition)

(1) An association between an Interface, a concrete protocol and a data format

SOAP WS: characteristics

- SOAP Web services
  1. are autonomous
  2. expose contracts
  3. have explicit frontiers
  4. communicate using messages
  5. communicate using a Web protocol
  6. are identified by URIs
  7. have messages, interfaces, bindings described in XML

SOAP WS architecture



Main SOAP WS Standards

WSDL

W3C

Web Services Description Language

Contract description

SOAP

W3C

Simple Object Access Protocol

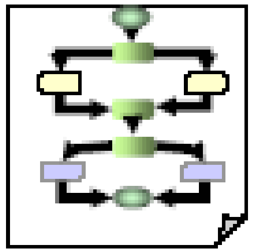
Transport

UDDI

Microsoft, IBM, HP

Universal Description Discovery and Integration

Spec for Repository/Registry



BPEL

Oasis

Business Process Execution Language

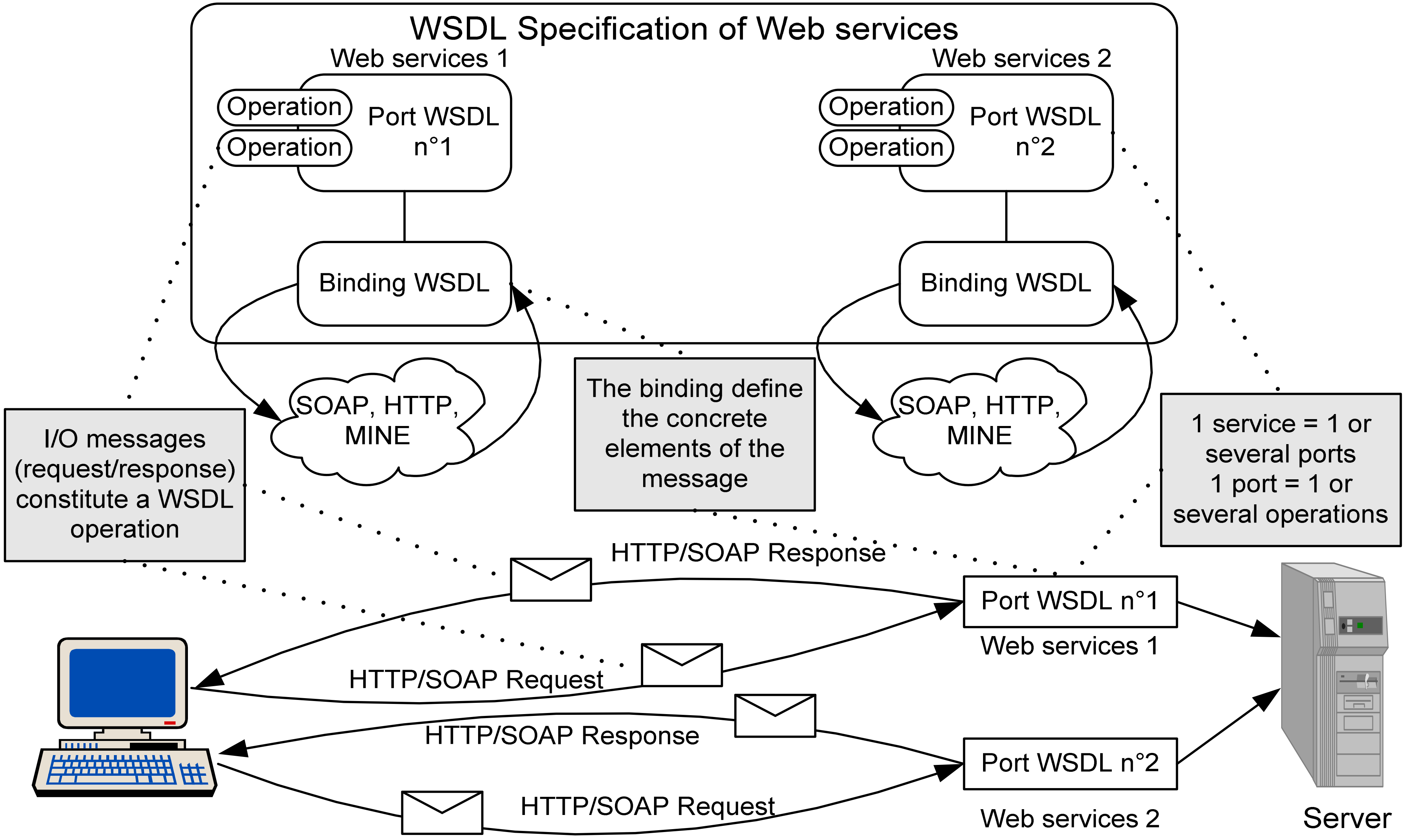
WS business processes

Source A. Occello

SOAP Web Services

WSDL : Web Service description Language

Web service description using WSDL



Source : Hubert Kadima & Valérie Monfort

Introduction

- What is WSDL ?
  - stands for Web Service Description Language
  - is an XML document for describing web services
  - represents the behavior of a resource on the Web

- What can one knows from WSDL ?
  - What kind of message is exchanged ?
  - How are the message related ?
    - (e.q operation input or output)
  - How SOAP messages are exchanged ?

WSDL structure

```
<definitions>
  <types>… </types>
  <message> … </message>
  <portType> … </portType>
  <binding> … </binding>
  <service> … </service>
</definitions>
```

Example: Address Book

- Operations
  - Add new entry
    - Input :
      - Last name: Tata
      - First name: Samir
      - Address: 9 rue Charles Fourier 91011 Evry France
  - Look for an address
    - Input: name
    - Output: Entry or Error message

The <types> element

WSDL

```
<types>
   <xsd:schema targetNamespace="urn:xml-soap-address-demo”
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
      <xsd:complexType name="phone">
         <xsd:element name="areaCode" type="xsd:int"/>
         <xsd:element name="exchange" type="xsd:string"/>
         <xsd:element name="number" type="xsd:string"/>
      </xsd:complexType>
      <xsd:complexType name="address">
         <xsd:element name="streetNum" type="xsd:int"/>
         <xsd:element name="streetName" type="xsd:string"/>
         <xsd:element name="city" type="xsd:string"/>
         <xsd:element name="state" type="xsd:string"/>
         <xsd:element name="zip" type="xsd:int"/>
         <xsd:element name="phoneNumber” type="typens:phone"/>
      </xsd:complexType>
   </xsd:schema>
</types>
```

The <message> element

```
<message name="AddEntryRequest">
   <part name="name" type="xsd:string"/>
   <part name="address" type="typens:address"/>
</message>

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

<message name="GetAddressFromNameResponse">
   <part name="address" type="typens:address"/>
</message>
```

The <porttype> element

- **One-way**
  - the endpoint receives an (<input>) message
- **Request-response**
  - the endpoint receives an (<input>) message and returns the related (<output>) message or one or several (<fault>) messages
- **Solicit-response**
  - the endpoint sends an (<output>) message and receives an (<input>) message or one or sevral (<fault>) messages.
- **Notification**
  - the end point sends a notification message (<output>)

The <porttype> element: example

```
<portType name="AddressBook">

     <!– One way operation -->
     <operation name="addEntry">
            <input message="AddEntryRequest"/>
     </operation>

      <!– Request-Response operation -->
      <operation name="getAddressFromName">
             <input message="GetAddressFromNameRequest"/>
             <output message="GetAddressFromNameResponse"/>
      </operation>

</portType>
```

The <binding> element

WSDL

```
<binding name="AddressBookSOAPBinding" type="AddressBook">
  <soap:binding
     style="rpc”
     transport="http://schemas.xmlsoap.org/soap/http"/>

   <operation name="addEntry">
     <soap:operation soapAction=""/>
     <input>
       <soap:body use="encoded” namespace="urn:AddressFetcher2"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/”/>
      </input>
    </operation>
```

The <binding> element

```
  <operation name="getAddressFromName">

    <soap:operation soapAction=""/>

    <input>
       <soap:body use="encoded”namespace="urn:AddressFetcher2"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/> </input>

     <output>
        <soap:body use="encoded” namespace="urn:AddressFetcher2"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/> </output>

  </operation>

</binding>
```

The <service> element

WSDL

```
<?xml version="1.0" ?>
<definitions name="urn:AddressFetcher"
         targetNamespace="urn:AddressFetcher2"
         xmlns:typens="urn:xml-soap-address-demo"
         xmlns:xsd="http://www.w3.org/1999/XMLSchema"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns="http://schemas.xmlsoap.org/wsdl/">
  ...
 <!-- service decln -->
 <service name="AddressBookService">
   <port
          name="AddressBook”
          binding="AddressBookSOAPBinding">
    <soap:address
        location="http://www.mycomp.com/soap/servlet/rpcrouter"/>
  </port>
 </service>
</definitions>
```

SOAP Web Service

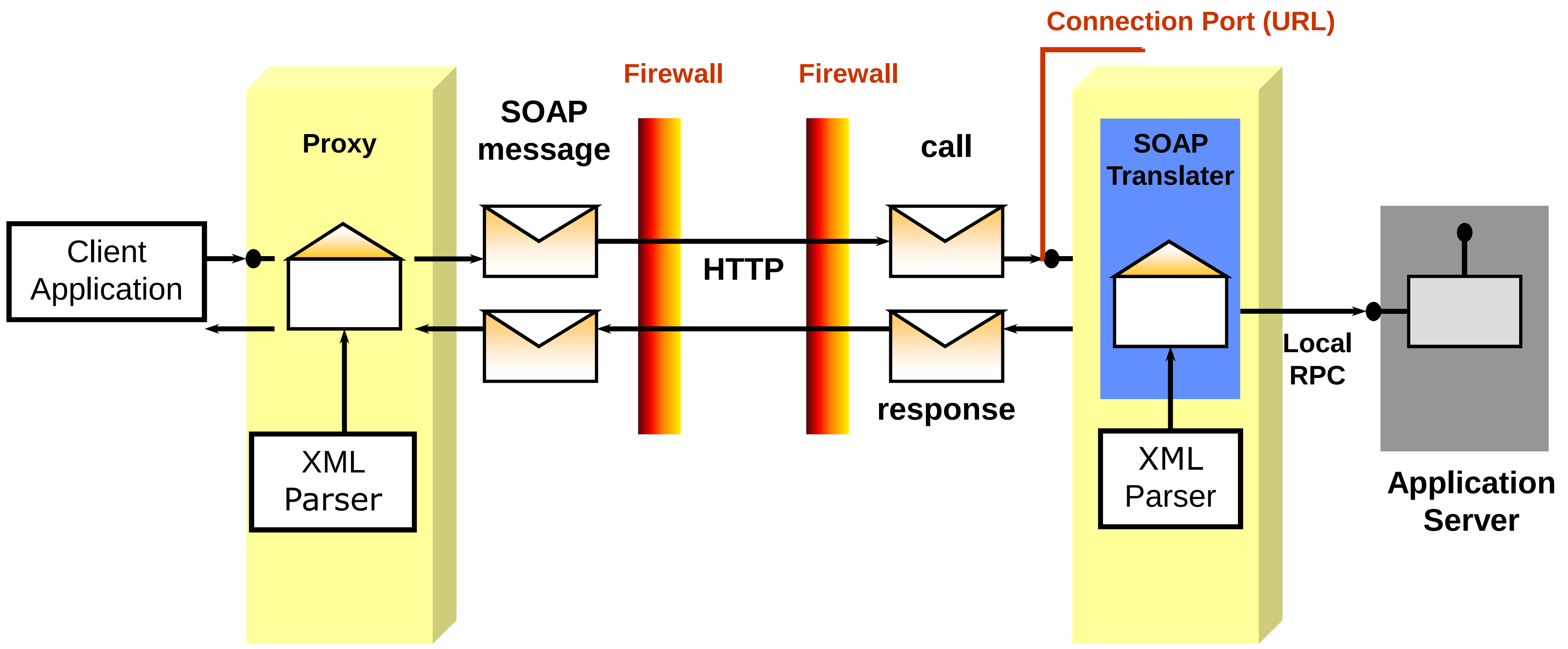
SOAP: Simple Object Access Protocol

What and why?

- SOAP
  - stands for “Simple Object Access Protocol”
  - is a communication protocol <u>specification</u> for invoking methods on servers, services, components, and objects.
  - is designed to communicate **via Internet**
- Why SOAP
  - is **platform independent.**
  - is **language independent.**
  - can be used in a large variety of systems ranging from messaging systems to RPC.
  - is **simple and extensible.**
  - is a format for **sending messages**

An exchange type

SOAP



Source G. Gardarin

SOAP Message Template

```
<soap:Envelope
                    … Envelop information goes here>
<soap:Header>
                    ... Header information goes here ...
</soap:Header>
<soap:Body>
                    ... Body information goes here ...
    <soap:Fault>
                    ... Fault information goes here ...
    </soap:Fault>
</soap:Body>
</soap:Envelope>
```

The Envelope and Header Elements

- The envelope element
  - Defines XML document as a SOAP message.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
</soap:Envelope>
```

- The header element
  - Contains user defined elements: language and currency.

```
<soap:Header>
    <m:local xmlns:m="http://www.Computer.com/local/">
      <m:language>fr</m:language>
      <m:currency>Euro</m:currency>
    </m:local>
</soap:Header>
```

The body element

- Must be present in SOAP message.
- Contains actual message

```
<soap:Envelope>
   <soap:Body>
      <m:sumRequest   xmlns:m="urn:MyFirstService">
         <param1>25</param1>
            <param2>-25</param2>
      </m:sumRequest>
   </soap:Body>
</soap:Envelope>
```

The body element

- Response

```
<soap:Envelope>
  <soap:Body>
    <m:sumResponse   xmlns:m="urn:MyFirstService">
      <return>0</return>
     </m:sumResponse>
  </soap:Body>
</soap:Envelope>
```

- Fault (errors that occurred while processing message, Only appears in answers responses)

```
<soap:Envelope>
  <soap:Body>
      <soap:Fault>
          <faultstring>Can't sum negative integers</faultstring>
      </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Implementations of SOAP

- Axis Apache
- Web Services Toolkit (IBM)
- JAXM (Sun)
- ZOAP (jBoss.org)
- HP Web Services Platform
- Microsoft SOAP toolkit (VB, etc.)
- .NET Framework (Microsoft)
- Many others…

UDDI

Universal Description, Discovery and Integration

Needs

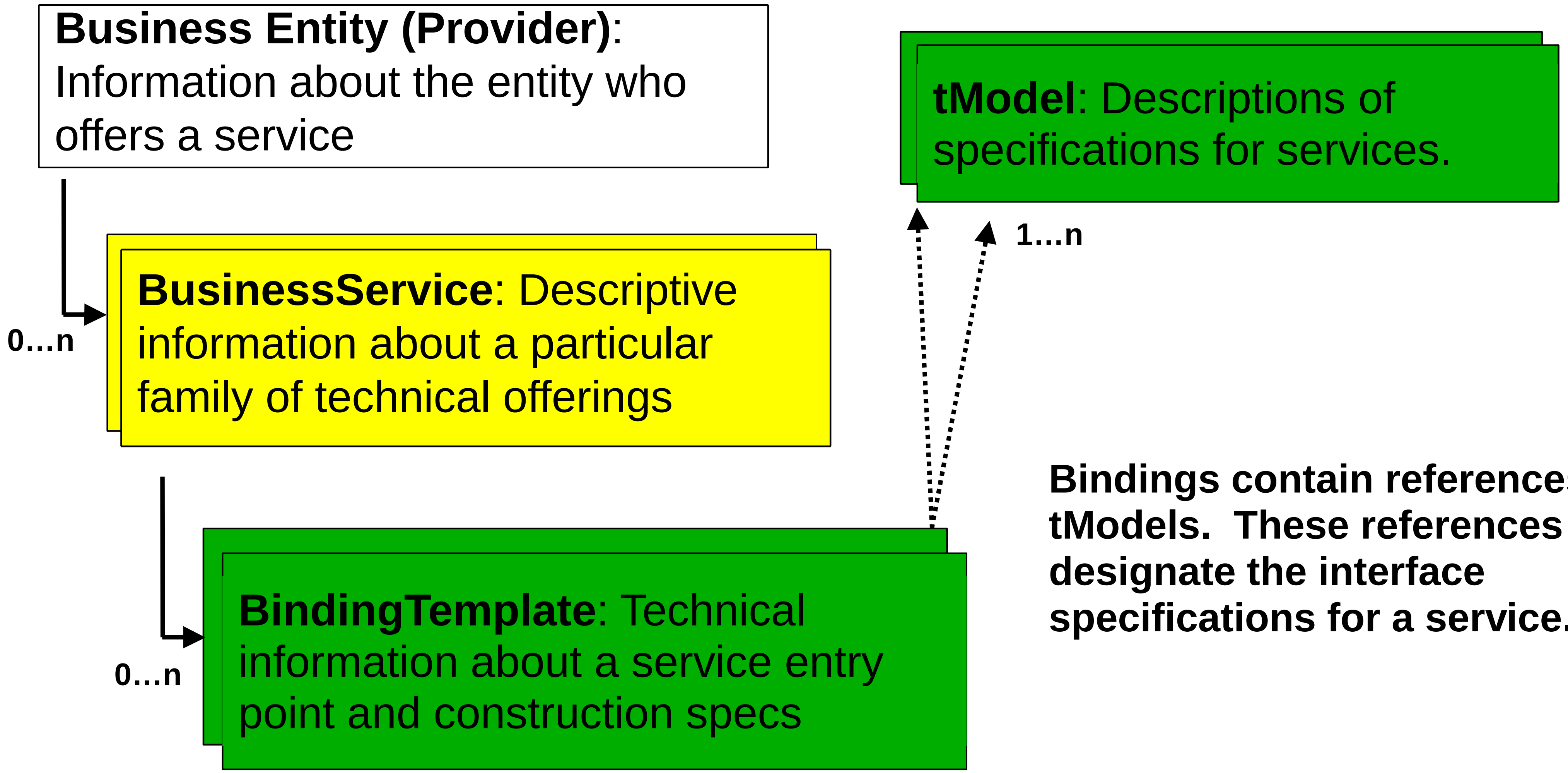
- Need to make services available
  - Which services are available?
    - Classes, Taxonomies, Locality
  - Regional, legal, trust boundaries
- Need to find services
  - Static and dynamic
- Need to negotiate capabilities
  - Security, Context, Transactions, Reliable Messaging
- Need to find ways to connect

UDDI roles

- UDDI plays three roles :White pages,Yellow pages, Green pages
  - White pages
    - address, contact, and known identifiers
  - Green pages
    - Namespace to describe how to use the service, etc...
    - Identifier of who published the service
    - Unique identifier (tModelKey) of this service for registration
  - Yellow pages
    - industrial categorisations
      - Industry
      - Product/Services
      - Location
- UDDI services are Web services

UDDI Information Model

UDDI



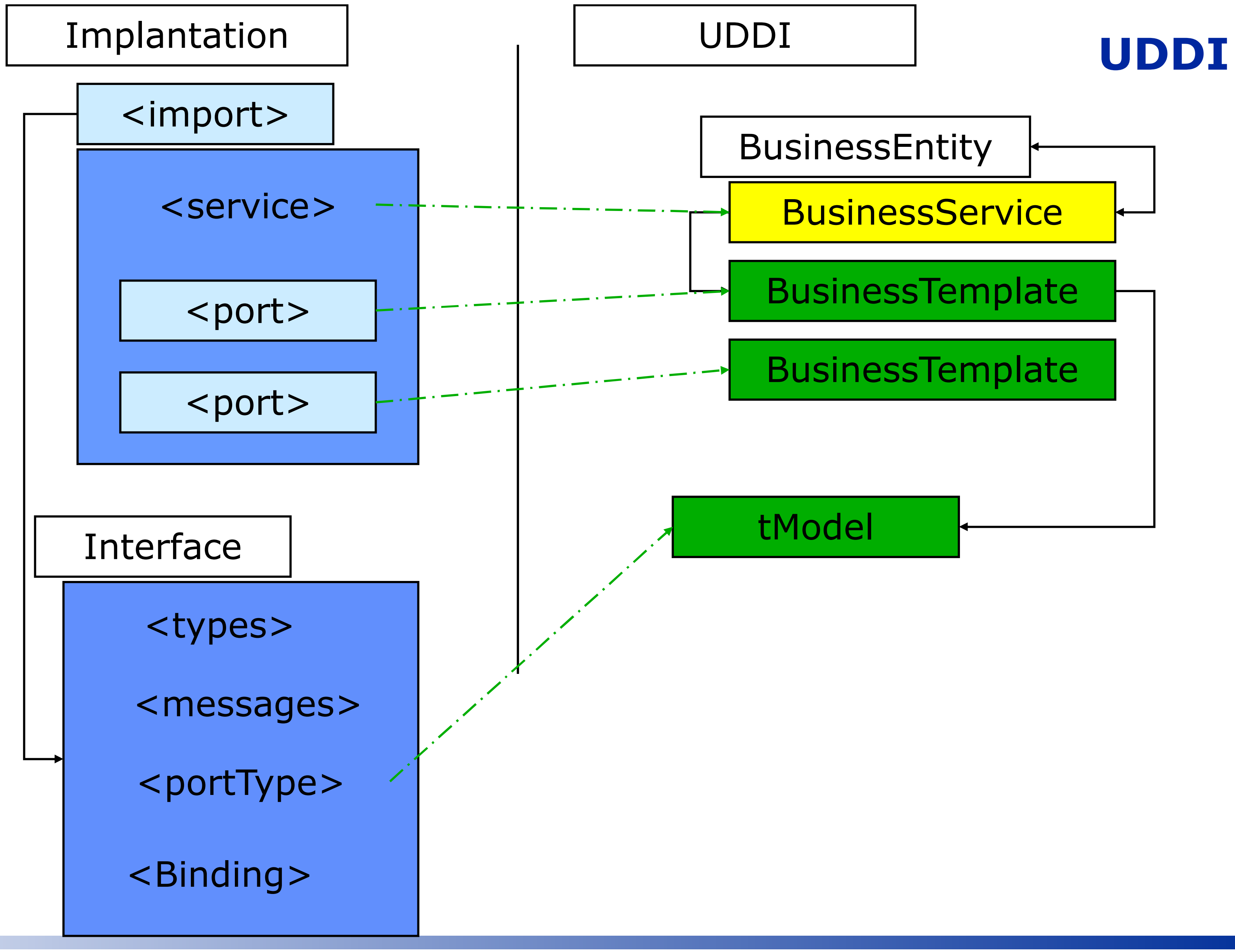
Business Entity (Provider): Information about the entity who offers a service
0…n
BusinessService: Descriptive information about a particular family of technical offerings
0…n
BindingTemplate: Technical information about a service entry point and construction specs
tModel: Descriptions of specifications for services.
1…n

Bindings contain references to tModels. These references designate the interface specifications for a service.

Providers, Services And Bindings

- Providers
  - Examples: Accounting Department, Corporate Application Server
  - Name, Description, Contact Information
  - Categorization and Identification Information
- Services
  - Examples: Purchase Order services, Payroll services
  - Name, Description(s)
  - Categorization Information
- Bindings
  - Description(s), access points, parameters
  - Examples: Access Point (http://...) for Web Service

UDDI



Implantation
UDDI
UDDI
<import>
<service>
<port>
<port>
BusinessEntity
BusinessService
BusinessTemplate
BusinessTemplate
tModel
Interface
<types>
<messages>
<portType>
<Binding>

UDDI

Implantation

```
<wsdl:definitions name="NormAdresseService"
            targetNamespace="http://…">
<import namespace="http://…"
 location="http://…" />
<wsdl:service name="DoNormeService">
<wsdl:port name="NormAdresseService "
 binding="intf:NormAdresseServiceSoapBinding">
…
</wsdl:service>
</wsdl:definitions>
```

Interface

```
<wsdl:definitions
name="NormAdresseService.interface"
targetNamespace="http://…">
<wsdl:message name="getNormeResponse">
</wsdl:message>
…
<wsdl:portType name="DoNorme">
</wsdl:portType>
<wsdl:binding
name="NormAdresseServiceSoapBinding"
type="intf:DoNorme">
</wsdl:binding>
</wsdl:definitions>
```

UDDI

```
<BusinessEntity businessKey="…"
<name> Normalisation d'adresse </name>
…
<businessService serviceKey="…">
<name> DoNormeService </name>
<BindingTemplates bindingKey="…">
<TmodelInstanceInfo tModelKey="…">
<overviewDoc>
<overviewdocURL>http://…# NormAdresseService
 </overviewdocURL>…
</BindingTemplates>
</businessService>
</BusinessEntity>
```

```
<tModel tModelKey="…"
<name> http://… </name>
…
<overviewDoc>
<overviewdocURL>http://…#NormAdresseServiceSoapBinding
            targetNamespace="http://…">
</overviewdocURL>…
<categoryBag>
<KeyedReference tModemKey="…" keyname="uddi-org:types keyvalue="wsdlSpec"/>
</ categoryBag >
</tModel>
```

SOAP Web Services

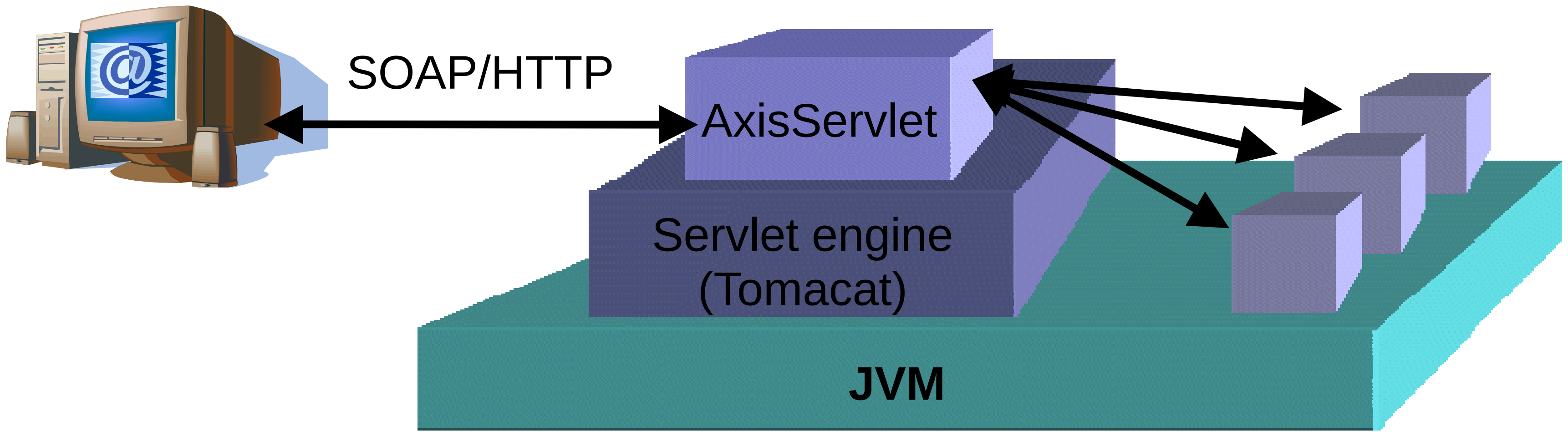
Axis

Axis

- Implantation of SOAP
  - Java
  - Open Source
- Apache community
  - Apache, Tomcat, Xerces, Struts, Cocoon
- Support “server side
  - Servlet that receives SOAP HTTP messages (SMTP)
- Support client side
  - API for sending SOAP messages over HTTP and SMTP

Axis

- Standalone
- Servlet



Axis2 with ADB

- Generation of classes to facilitate sending of SOAP messages:
  - $AXIS2_HOME/bin/wsdl2java.sh –uri file.wsdl -d adb -s

- Generated class:
  - Sub

- Client programming:
  1. Instantiate the stub
  2. Instantiate the query (subclass of stub)
  3. Initialize the query (subclass of stub)
  4. Call one of the stub method
  5. Use the output (e.g print out the output)

Services Web

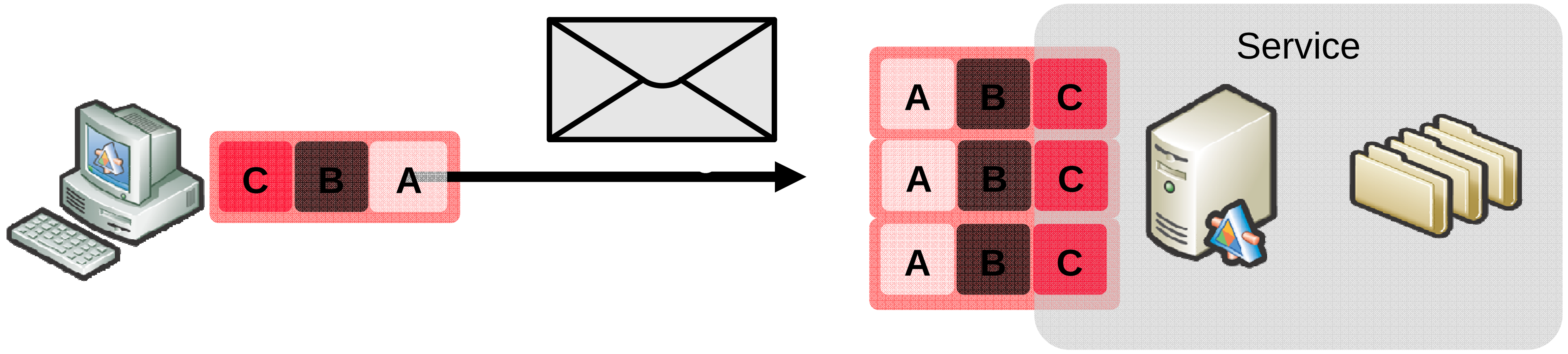
WCF

What is WCF?

- Unified programming model for building connected applications on the Windows platform
- Platform for advanced Web Services
  - Standards based
  - Building secure, reliable, transacted solutions
  - One choice for implementing SOA applications
- Unified Technologies
  - ASMX Web Services
  - Web Service Enhancements, WSE
  - .NET Remoting
  - Enterprise Services, COM+
  - MSMQ
- Interoperability with existing investments

The ABC of WCF

- Address – *where* to expose
- Binding – *how* to expose
- Contract – *what* to expose
- Defined in code or in configuration



Configuration

```
<system.serviceModel>
   <services>
      <service
          name="OrderService.OrderManager"
         <!-- use base address provided by host -->
         <endpoint         address="http://host:8080/OrderService"
                       binding="wsHttpBinding"
             contract="OrderService.IOrderManager" />
      </service>
   </services>
</system.serviceModel>
```

Address

- Defines *where* a service is located
- Specifies a URI where the service is located
  - Relative or absolute
- Address consist of
  - Scheme
    - HTTP, TCP, Named pipes, MSMQ
  - Machine
  - [Port]
  - Path
- Examples
  - http://www.mystore.com/StoreFront
  - net.tcp://mycomputer:9000/StoreFront

Binding

- Describes *how* a service communicates
- Specifies set of binding elements
  - Transport; http, tcp, np, msmq
  - Encoding format; text, binary, MTOM, ...
  - Security requirements
  - Reliable session requirements
  - Transaction requirements
- Set of predefined standard bindings
  - Can be customized
- Custom binding

A Service Contract

```
// Define a service contract.

  [ServiceContract(Namespace="http://Microsoft.ServiceModel.Samp
  les")]
  public interface IDataContractCalculator
  {
     [OperationContract]
     ComplexNumber Add(ComplexNumber n1, ComplexNumber n2);
     [OperationContract]
     ComplexNumber Subtract(ComplexNumber n1, ComplexNumber
  n2);
     [OperationContract]
     ComplexNumber Multiply(ComplexNumber n1, ComplexNumber
  n2);
     [OperationContract]
     ComplexNumber Divide(ComplexNumber n1, ComplexNumber
  n2);
  }
```

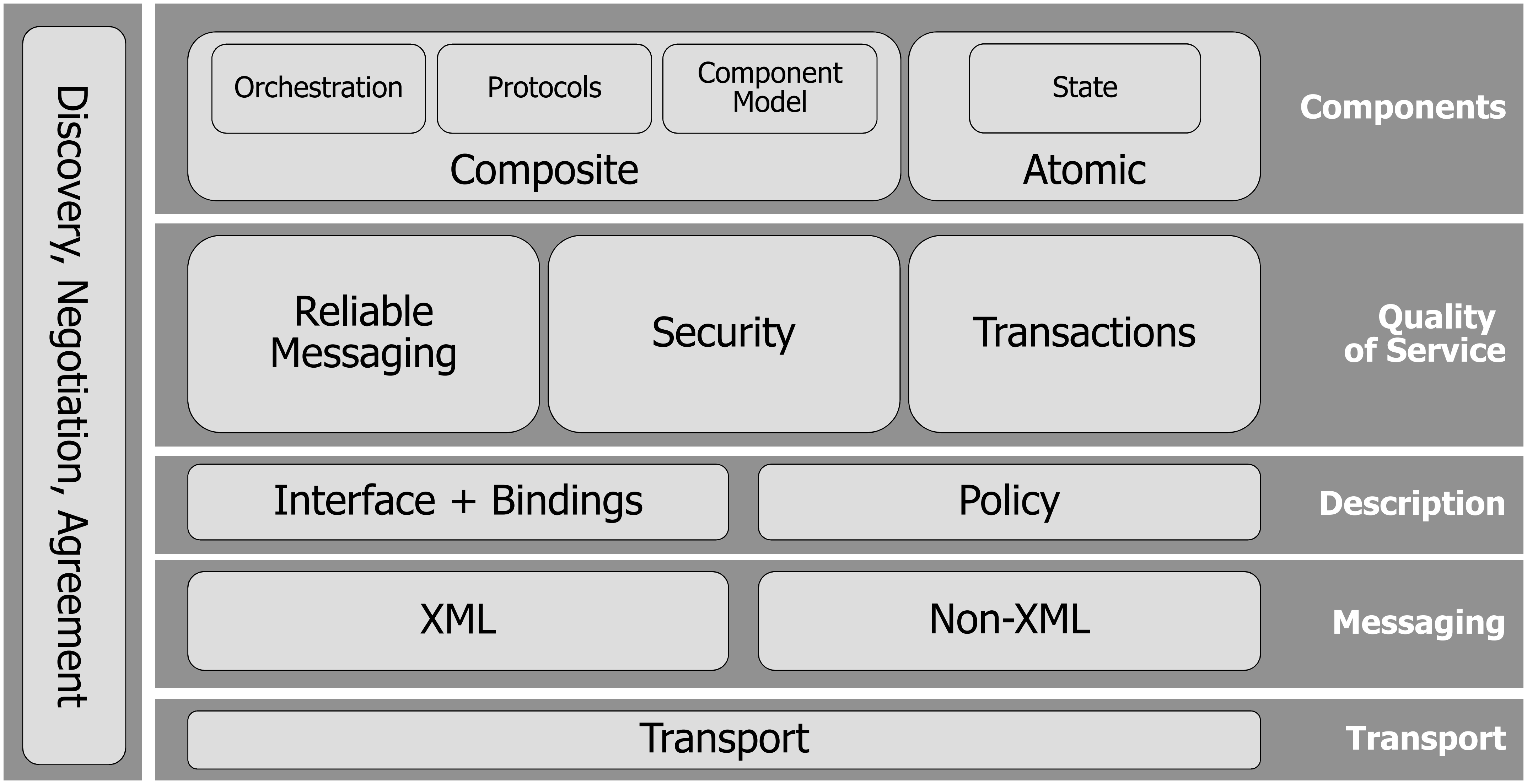
Data Contract

```
[DataContract]
  public class ComplexNumber
  {
     [DataMember]
     public double Real = 0.0D;
     [DataMember]
     public double Imaginary = 0.0D;
  }
```

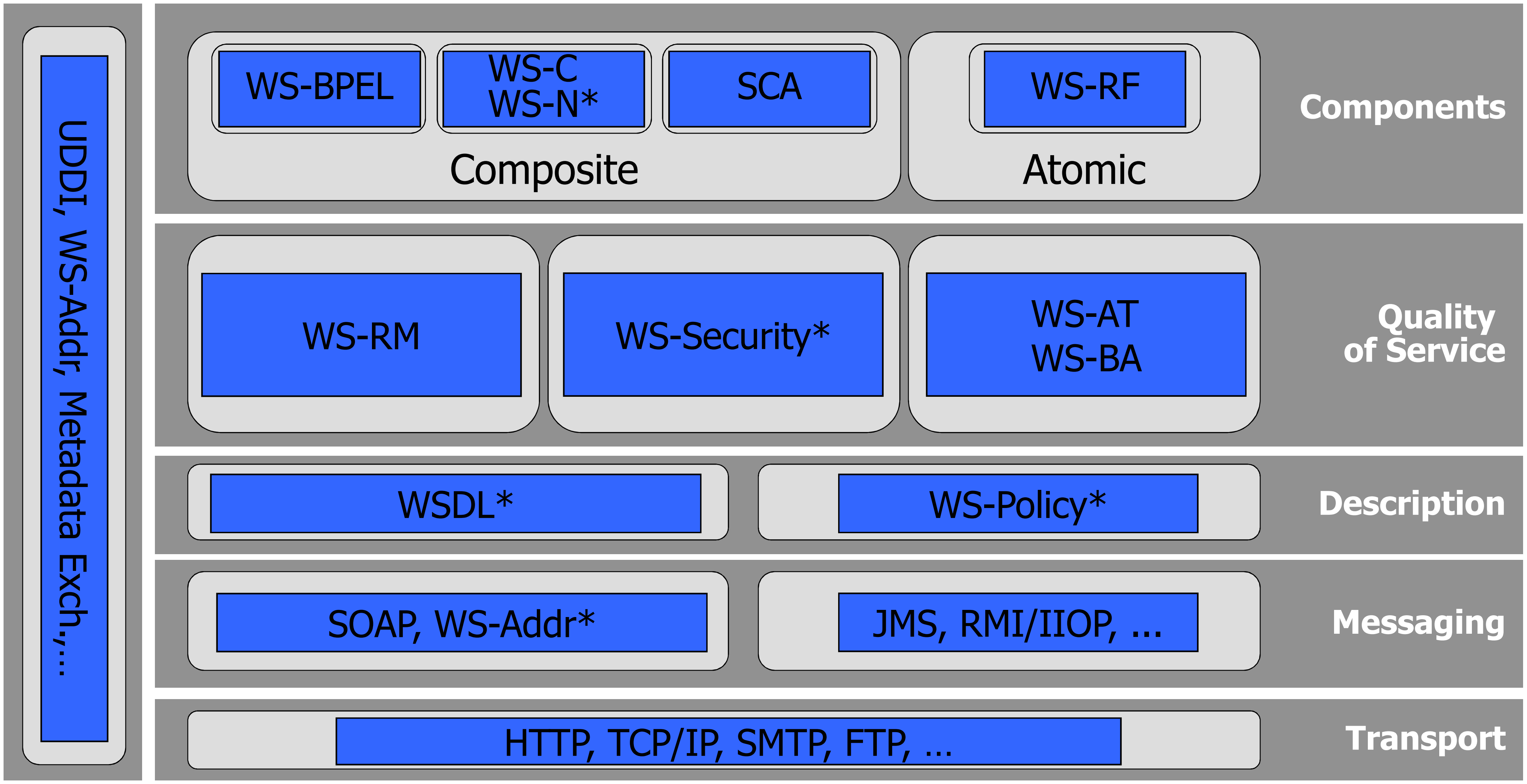
SOAP Web Services

WS-* standards

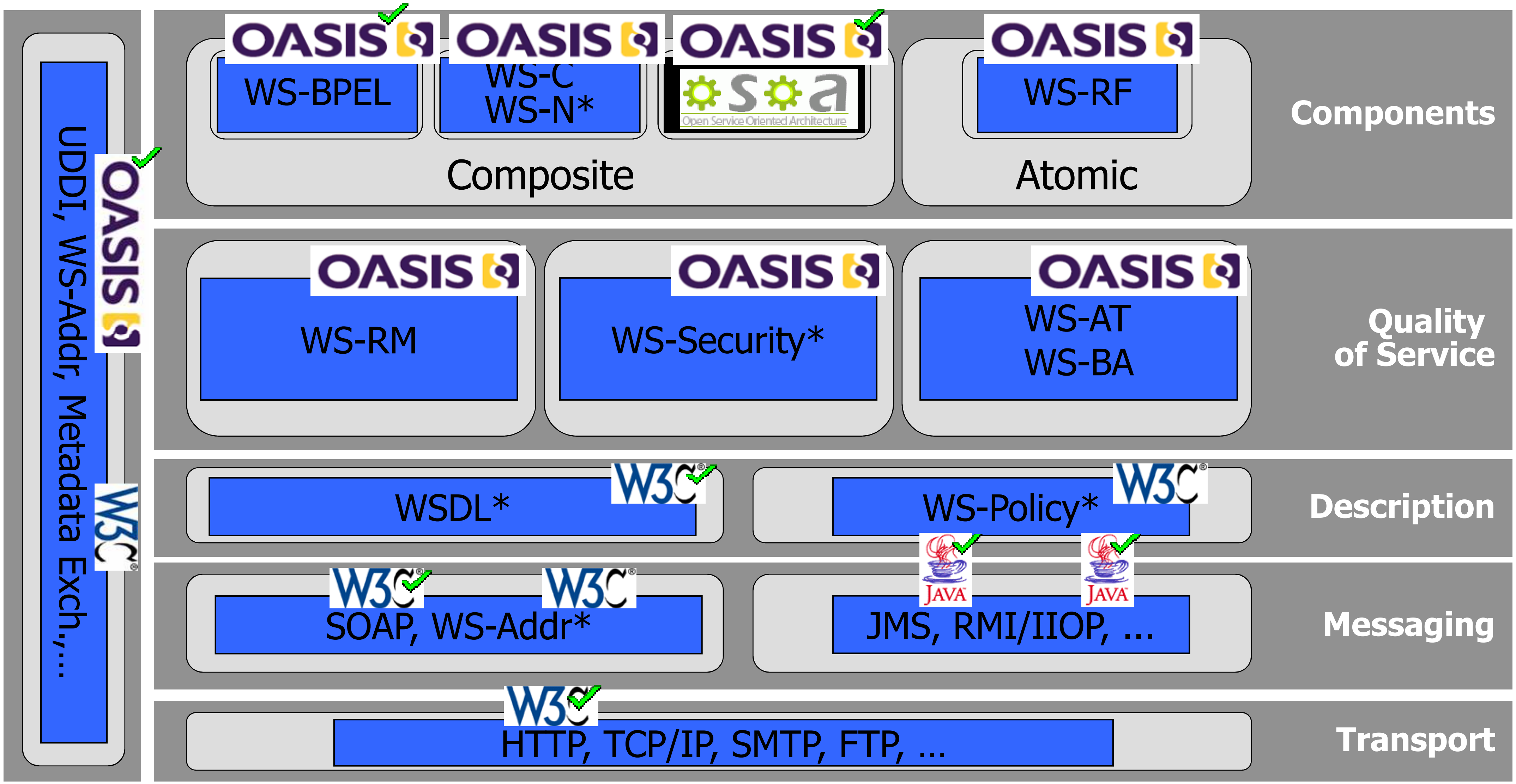
Problem with WS Standards



Problem with WS Standards



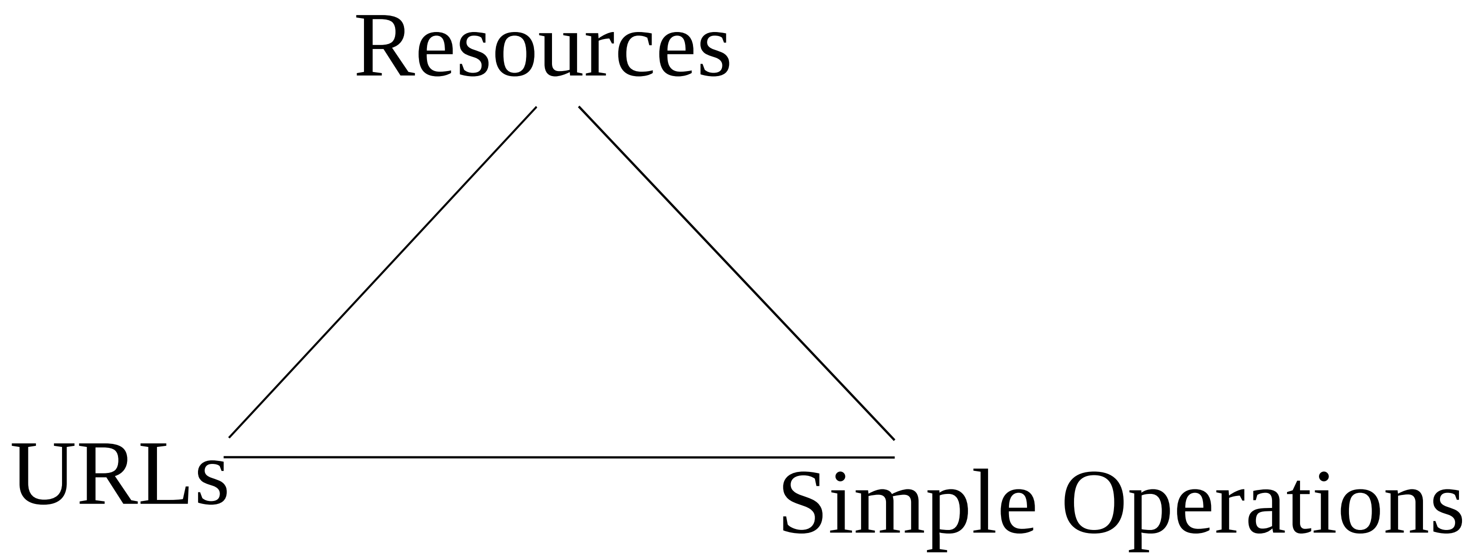
Problem with WS Standards



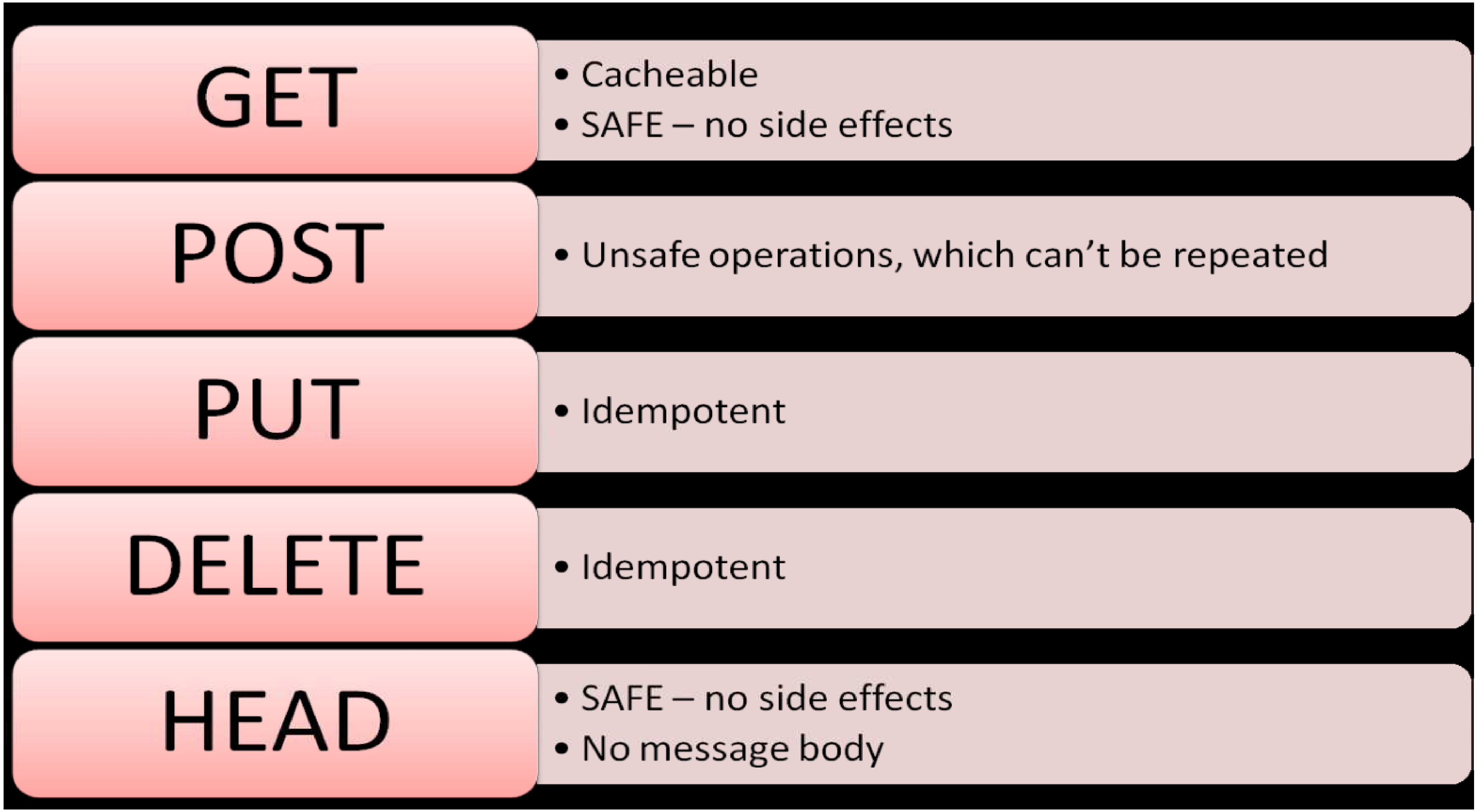
RESTFull Web Services

RESTFull Web Services

The Three Fundamental of REST



REST Operations

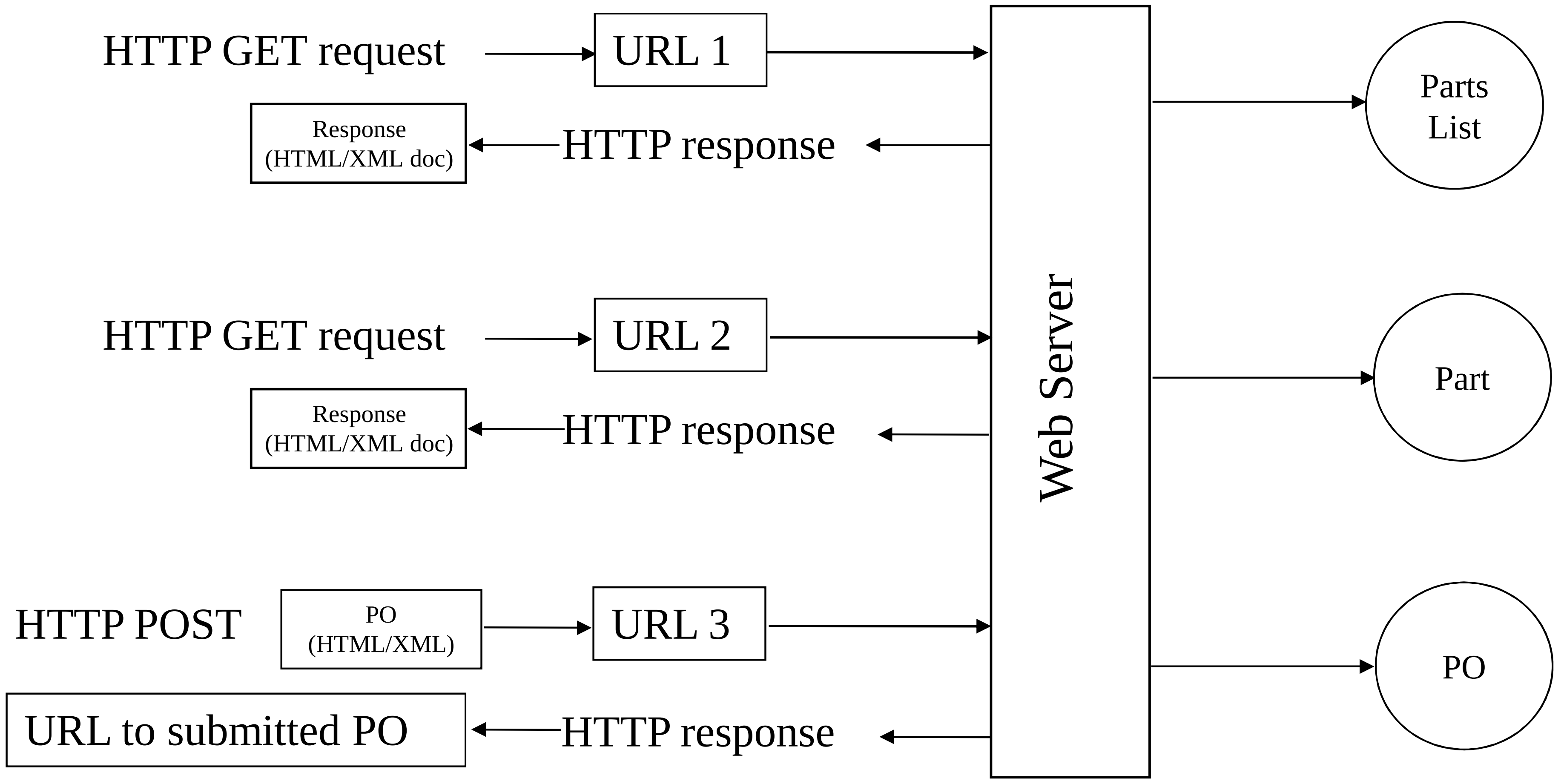


GET	• Cacheable • SAFE – no side effects
POST	• Unsafe operations, which can’t be repeated
PUT	• Idempotent
DELETE	• Idempotent
HEAD	• SAFE – no side effects • No message body

RESTFull: characteristics

- RESTFull Web services
  - are autonomous
  - expose contracts
  - have explicit frontiers
  - communicate using messages
  - communicate using a Web protocol
  - are identified by URIs
  - Only provide REST operations

RESTfull Web Service



Steps to a RESTfull Web Service

- Determine the resources (implicit objects) in the service.
  - getPurchaseOrder, getInvoice then pretty → Purchase Order and Invoice objects.
- Create a script, servlet or JSP for each kind of object.
  - a GET method that returns XML conforming to the schema.
  - a PUT method that updates the underlying database for every transaction.
  - a DELETE method for removing relevant data from the database.
- Replace each set of getXXX, setXXX and deleteXXX methods with a single hyperlink.
- Replace any method that adds a contained resource to a container with POST.
- Also replace any method that mutates the current state (e.g. increments by one or doubles, or appends) with POST
- Replace any search-like methods with GET-queries.

SOAP vs RESTfull WS

- REST web services are:
  - Lightweight - not a lot of extra xml markup
  - Human Readable Results
  - Easy to build - no toolkits required

- SOAP
  - Easy to consume - sometimes
  - Rigid - type checking, adheres to a contract
  - Development tools
  - Granularity