XIII. Service Oriented Computing

Outline

- Enterprise Application Integration (EAI) and B2B applications
- Service Oriented Architecture
- Web Services
- WS technologies

further reading:

G. Alonso, F. Casati, H. Kuno, V. Machiraju

" Web Services – Concepts, Architectures and Applications"

Spinger-Verlag 2004

Enterprise Application Integration (EAI)

- Middleware make easier the development of distributed applications
- The emergence of "the Web" EAI and B2B vision
  - Different organisations want to cooperate to satisfy a user request
  - selection and booking of flights
  - ordering and delivering of goods
  - Search and booking of hotels
  - etc...

Flight Booking System

Primary Actor: Customer

Secondary Actors: airline companies, banking system and credit card management system, web portal company...

Precondition: the booking system is running

Postcondition: The customer has a receipt describing flight detail and the flight has been correctly booked in the airline booking system and the right amount of money has been withdrawn from the customer credit card account and transferred to the airline company account. If no seat are available on the selected day the user is informed and the initial interface is provided

Main Scenario

1. the customer loads the booking system page
2. the customer provides information concerning the journey
3. the system returns a set of possible options for the same day the previous one and the next one
4. the customer selects preferred flights and submit the data to the system
5. The system store the information concerning the journey and asks the customer to chose a way of payment
7. The customer select the way of payment
8. The system asks details for the selected way of payment
9. The customer provides payment details

Flight Booking System ...ctnd

...Main Scenario

10. the system checks and stores payment details and provides confirmation details to the customer

Extensions

...

3a. no seat available for the selected dates

1. the system communicate to the customer the event and shows the first page (the UC terminates)

...

Flight Booking System characteristics

- Distributed Application
- It is composed of many different components
- **The computation cross the boundaries of many different organisations**
  - Web Portal, airlines companies, banks, credit card organisation
- The different software system that compose the whole application are in general already available and running
- The problem that we need to solve is more related to integration than to development

Middleware for the Flight Booking System

- How can we implement such kind of system?
- Can "traditional" middleware be used to easily accomplish the integration task?
- Which are the main issues raised by the use of the "traditional" middleware?

Why not using "traditional" middleware?

- Inter-organization interoperability based on traditional middleware, many drawbacks:
  - Traditional middlewares foresee the development of distributed applications on top of a centralised (at least logically) system
    - **Centralization points** – given more partners, who should manage middleware?
  - The same organization could be obliged **buying and managing more middleware** to cooperate with different partners in different context
    - **Middleware-to-middleware bridges are really expensive** (and not always feasible)

- EAI in B2B domain can be seen as a step forward in the evolution of middleware
  - Crossing inter-organisation borders raise new problems
    - Data sharing among different organisation
    - Control on the execution is not "centralised"
    - Cross-organisational interactions last much longer
      - Support for asynchronous interaction mechanisms becomes much more relevant
    - Time and quality parameters are much more difficult to derive
    - Transaction management (2PC requires centralised transaction coordinator)

A possible solution

- Solution to B2B application integration involves different ingredients:
  - Adoption of a Service Oriented Approach
  - Revision of Middleware protocols (distributed transaction management)
  - Standardisation

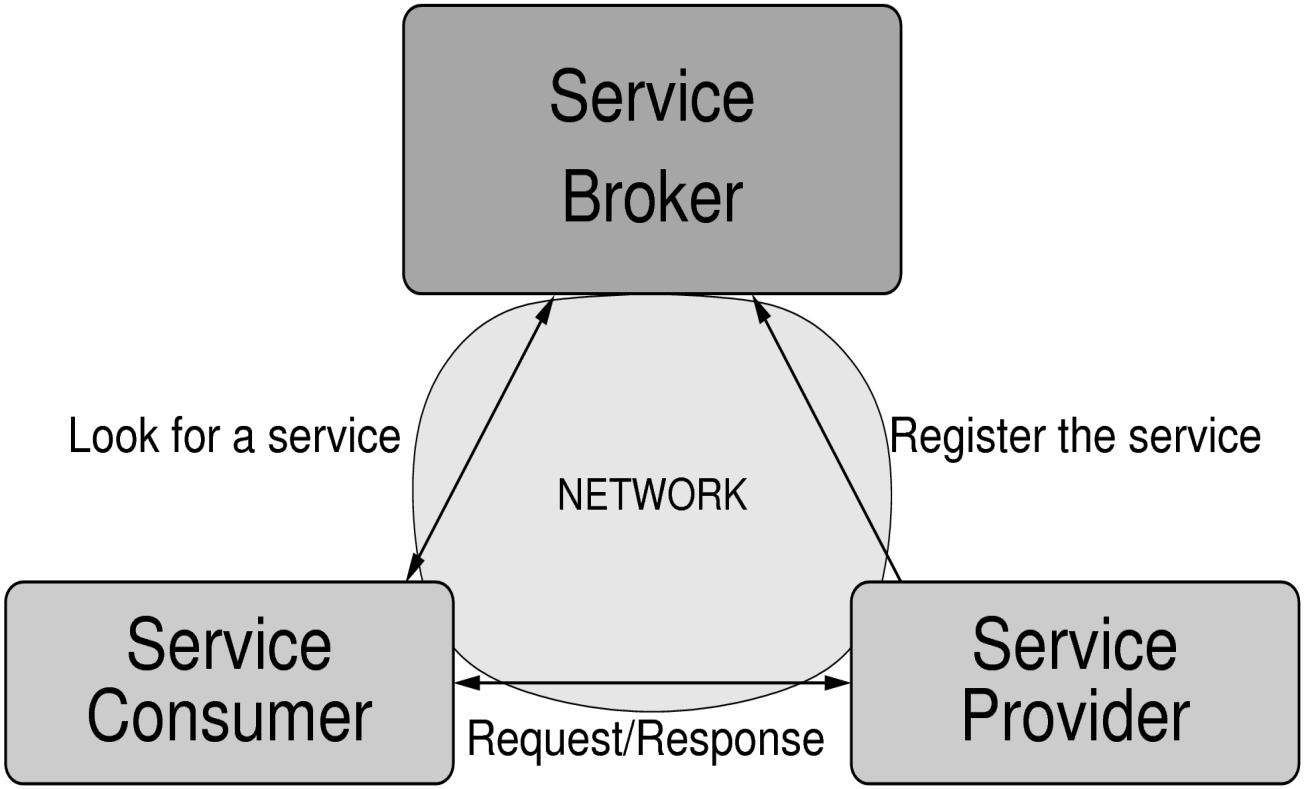
What is a Service?

- A software service is a software component providing a functionality to other software component
- A service defines a **precise interface** to be used by clients in order to access the service. To each service it is associated one or more **access points** defining the location where the provided functionality can be accessed
- the interface and the access point can be **dynamically retrieved and used** by other software components
- Service creation is not related to the existence of users

- Service is an **abstract concept** asking for the creation of a more complex architecture to make it real

Service Oriented Architecture (SOA)

- SOA is a conceptual architecture (**does not define a particular technology**) making service characteristics real



- SOA is a kind of distributed systems

Services in the SOA context

- A service can be dynamically **discovered** and **accessed**
- Service must be **modular** and **self-contained**
- Service must show an interface hiding implementation details
- Service must be **loosely-coupled**
- Service must be **published to be retrievable**
- Service must provide **coarse-grained interface**
- Service can be a **unit of composition**

Web Services

- Different technologies logically reflect this architecture (e.g. CORBA with services)
- What makes Web Services (WS) different?
- **WS adopts open and XML based standards and is "Web centered"**

W3C – WS definition

- Definition [W3C – WS Architecture]:

  "A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."

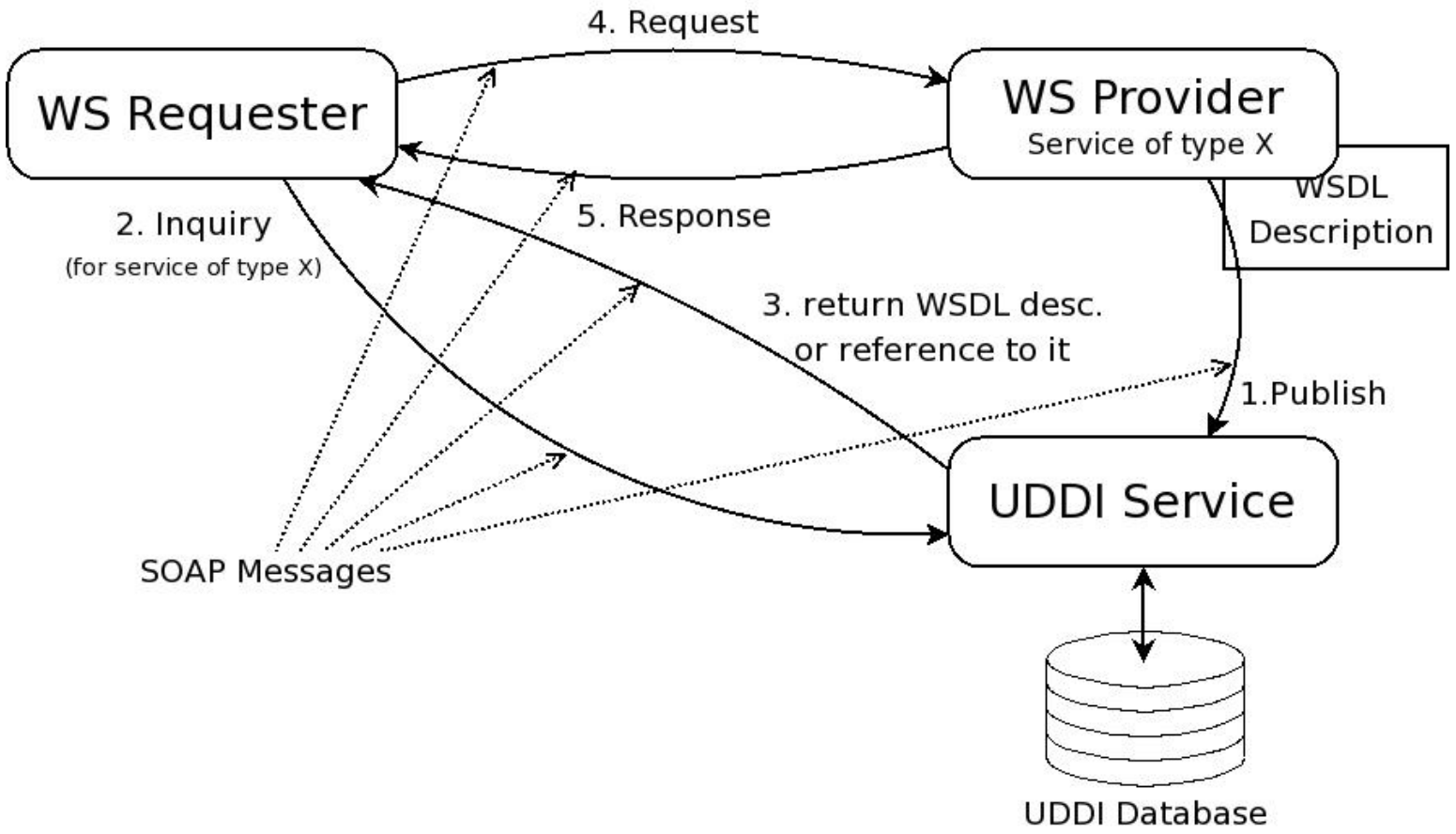
Which are the characteristics of WSs?

- **Logical view**: The service typically described in terms of what it does, and at an abstract, logical level
- **Message orientation**: The service only defines the messages exchanged between provider agents and requester agents. Any detail about how an agent implementing a service is constructed is deliberately abstracted away
- **Description orientation**: To support the public nature of the SOA, a service must describe, by means of machine-processable metadata, all and only those details that are important for the public use of the service. The semantics of a service should also be documented, either directly or indirectly, by its description
- **Granularity**: Services generally provide a small number of operations with relatively large and complex messages
- **Network orientation**: Services tend to be oriented towards use over a network
- **Platform neutral**: Messages are sent through the interfaces in a platform-neutral, standardized format, most often XML

Which are the basic WS bricks?

- Web Services (WS) constitute the most important implementations of SOA concepts. Core elements of the technology are:
  - **eXtensible Mark-up Language** (XML – W3C)
  - **Web Service Description Language** (WSDL – W3C): this is a XML based language **used to describe a Web Service**. Information in a WSDL file specify offered services, access points, format of in/out parameters and accepted mechanisms used to exchange messages;
  - **Simple Object Access Protocol (**SOAP – W3C**):** this is a simple protocol that can be **used to exchange XML based messages,** and it is the natural choice to establish communications among Web Services.
  - **Universal Description and Discovery Integration** (UDDI – OASIS): this open technology, which specification is developed by the OASIS consortium, defines a **common mechanism to publish and retrieve information about available Web Services**.

WS as a SOA



- Everything use XML format to communicate and store information

eXtensible Mark-up Language (XML)

- XML is a **meta-language** derived in 1998 from the SGML
- It provides a powerful **mean to exchange data**
- It is based on a text format so can be **exchanged among different platforms**
- Each time one or more software programs need to exchange data define a common format
- The presence of tags make easier handling and retrieving the data from the document

eXtensible Mark-up Language (XML)

- Data are inserted in a tree based structure using defined tags and attributes

```
<?xml version="1.0"?>
<anagrafica>
  <cittadino maggiorenne="si">
    <nome>Mario</nome><cognome>Rossi</cognome>
    <indirizzo>
      <residente>Via Repubblica</residente><civico>2</civico>
      <citta>Roma</citta>
    </indirizzo>
  </cittadino>
  ...
</anagrafica>
```

attribute

tag

- well-formedness defines a set of rules to be followed to have a correct XML file (e.g. tags cannot be nested)
- HTML is not XML nevetheless it has been defined a XML based format for HTML (XHTML)

How to define XML document format

- Tools to define agreed format and check for correctness:
  - Rule based languages (e.g. Schematron): define correct document structures through the specification of logical statements
  - Structural based languages (e.g. Relax NG, XML Schema, DTD): define correct document structures through the specification of "template"

- XML Schema certainly the most relevant proposal, it permits to define:
  - Allowed elements and attributes,
  - Elements structure (mandatory or optional attributes and child elements)
  - Orders and cardinality of child elements
  - Type of elements and attributes
  - Mandatory or optional content
  - Default values and constant values for elements and attributes

XML Schema example

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="anagrafica">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="cittadino" type="xs:string">
    <xs:complexType>
      <xs:sequence>
       <xs:element name="nome" maxOccurs="1" type=”xs:string”/>
       <xs:element name="cognome" maxOccurs="1" type=”xs:string”/>
       <xs:element name="Indirizzo" maxOccurs="2" type=”xs:string”>
         <xs:complexType>
           <xs:element name="residente" maxOccurs="1" type=”xs:string”/>
           <xs:element name="civico" maxOccurs="1" type=”xs:integer”/>
           <xs:element name="citta" maxOccurs="1" type=”xs:string”/>
         </xs:complexType>
       </xs:element>
     </xs:sequence>
   </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

XML tools

- Accessing XML documents Simple Access XML vs. Document Object Model

- Editors: XMLSpy, oXygen ...
- Java libraries: Apache Xerces, Java API for XML Processing (JAXP) ...

Simple Object Access Protocol (SOAP)

- SOAP application protocol standardised by W3C (ver. 1.2 May 2003)
- Difficulties in integration across the Internet: firewalls, lack of standardised protocols, need for loosely-coupled interactions – existing protocols were not suitable

- SOAP specifies:
  - A message format for one-way communication describing how information can be packaged into XML document (**Message Format**)
  - A set of conventions for using SOAP messages to implement RPC interaction pattern (**Conventions for RPC**)
  - A set of rules that any entity that processes a SOAP message must follow (**Understanding SOAP messages**)
  - A description of how a SOAP message should be transported on top of HTTP and SMTP (**Binding**)

SOAP

- SOAP specifies:
  - A message format for one-way communication describing how information can be packaged into XML document (**Message Format**)
  - A set of conventions for using SOAP messages to implement RPC interaction pattern (**Conventions for RPC**)
  - A set of rules that any entity that processes a SOAP message must follow (**Understanding SOAP messages**)
  - A description of how a SOAP message should be transported on top of HTTP and SMTP (**Binding**)
- SOAP is a **stateless and one-way protocol**
  - Any further complexity in communication pattern requires SOAP to be combined with the underlying protocol
- To define correct message format SOAP 1.2 uses XML Schema recommendation

SOAP message structure

SOAP envelope

SOAP header

Data to be processed by intermediate nodes that process the message and route it to the receiver

Three different roles for intemediate nodes:
none, next, ultimateReceiver (defaults)

Processing: *must Understana*

SOAP body

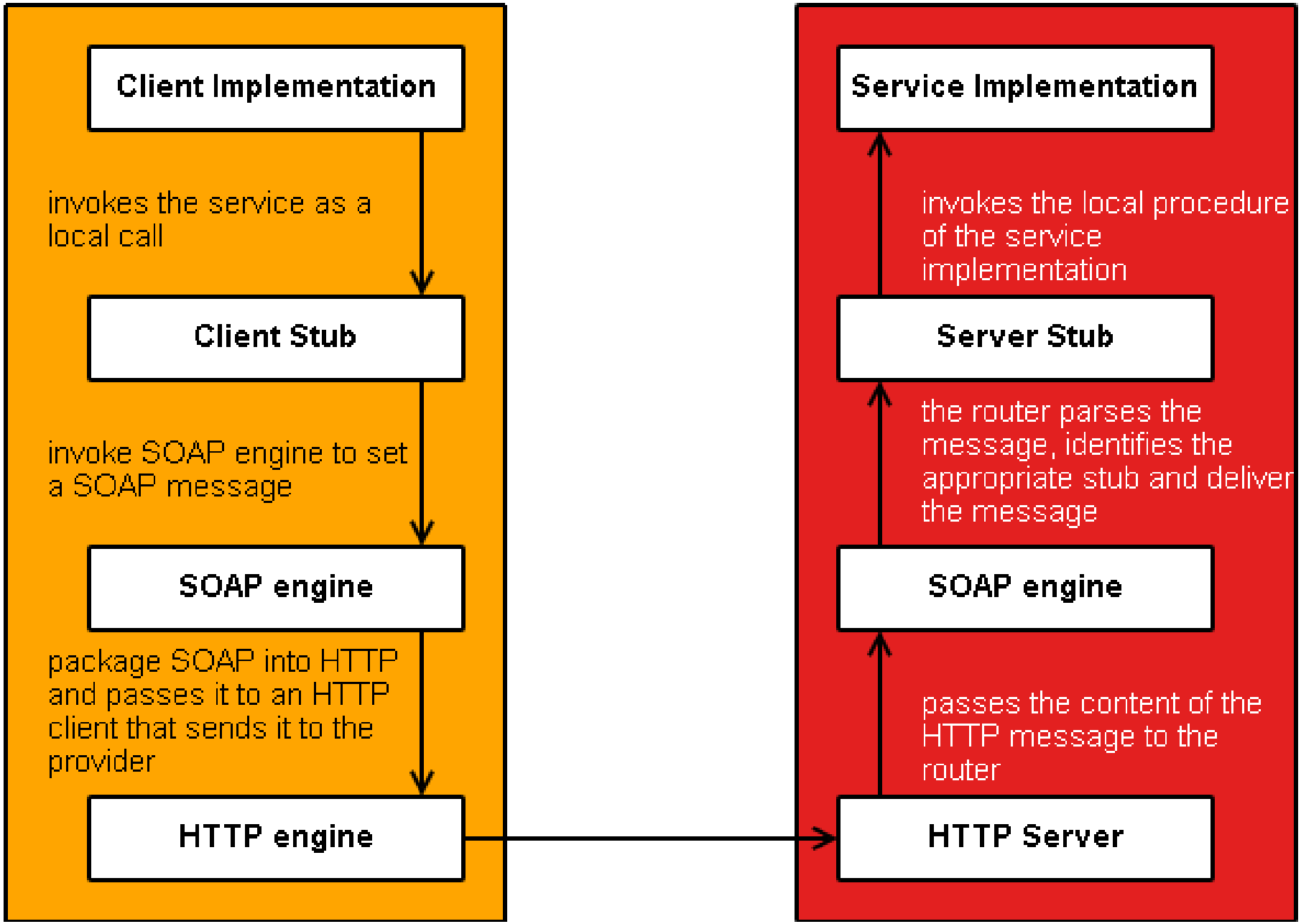
Document Style vs. RPC style

SOAP Binding

- Synchronous SOAP bound to HTTP
  - Use of POST to reproduce RPC interaction pattern (sender wait for an answer)
- Asynchronous SOAP generally bound to SMTP (e-mail)

- Used to describe addressing of a message
- Routing?
  - The concept appear in the spec but currently message routing policy relies on underlying protocols

SOAP Simple Implementation

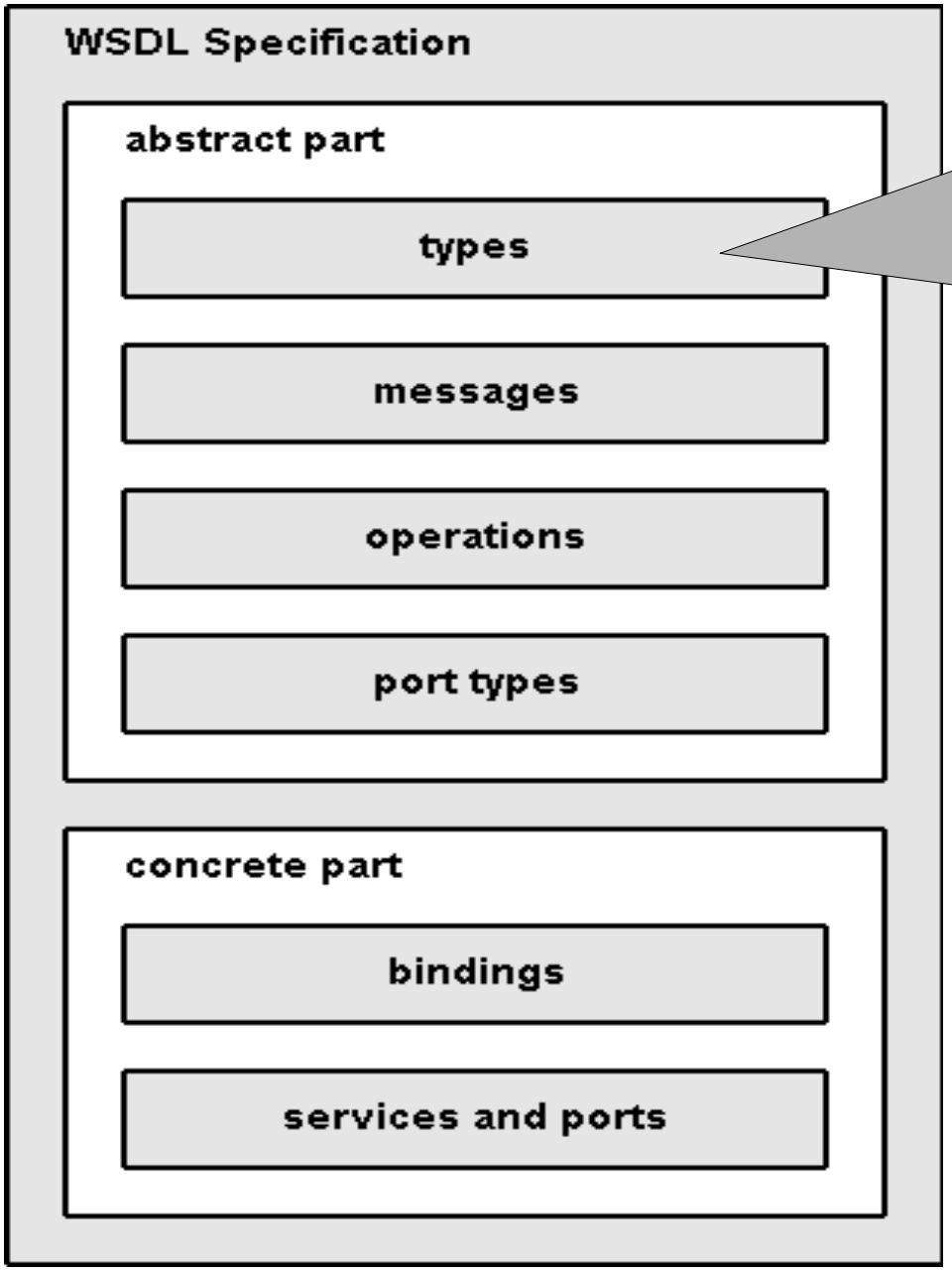


Web Service Description Language (WSDL)

- WSDL is an XML based language used to describe a web service interface
- Play the same role of an IDL in "traditional" middleware
- Standardized by W3C currently version 2.0

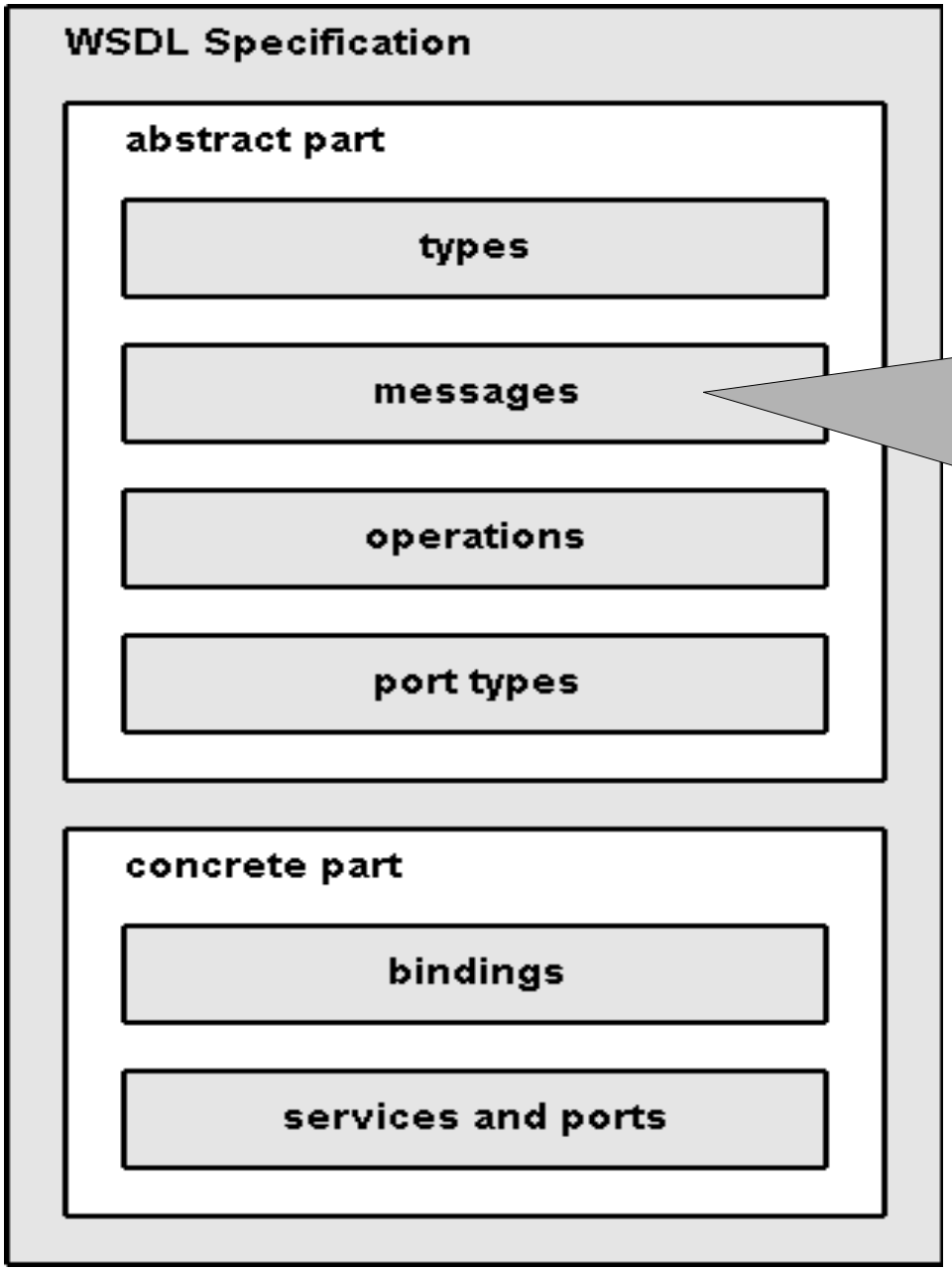
- Who can be interested in defining a WSDL?

WSDL – Types



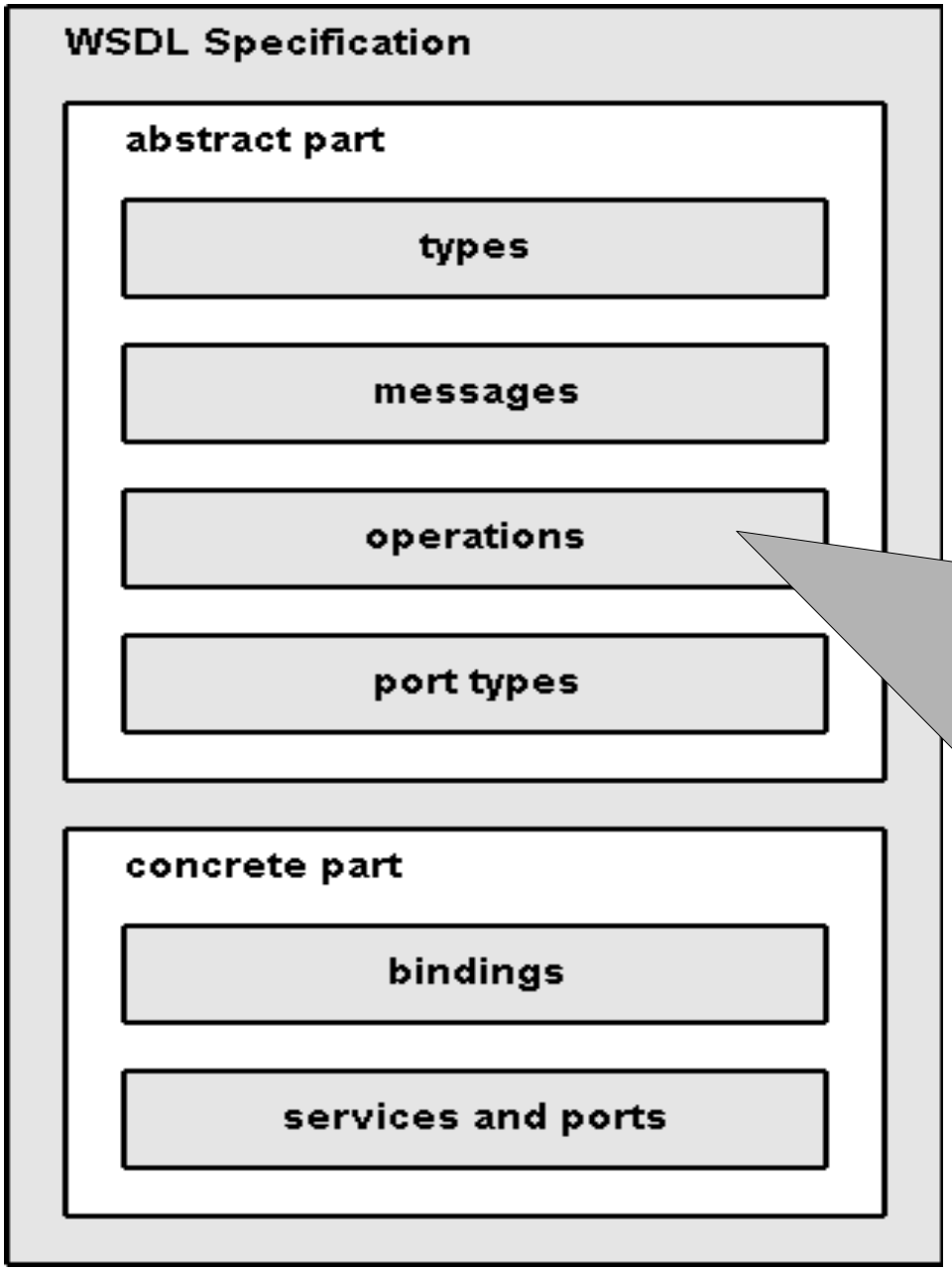
Here all the data structures that will be exchanged as parts of messages between applications
WSDL "reuses" the type system defined in the XML schema spec

WSDL – Messages



A message is a typed document exchanges among applications. It is defined on the base of data type defined in the previous section and it is divided in parts

WSDL – Operations



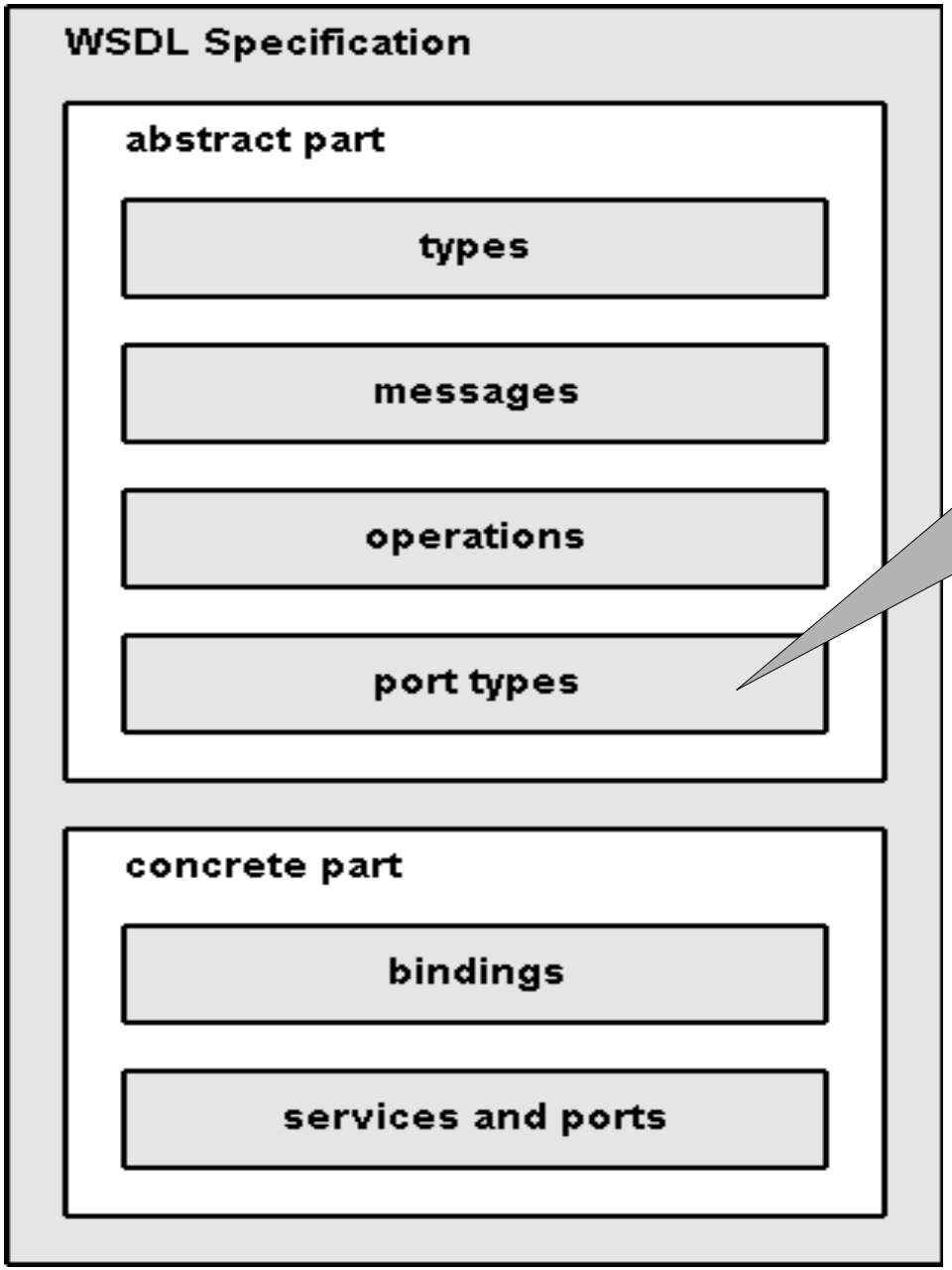
Group messages to define Operations

Four basic operations are defined:

1. one-way
2. request-response
3. solicit-response
4. notification

synchronous: 2,3
asynchronous: 1,4

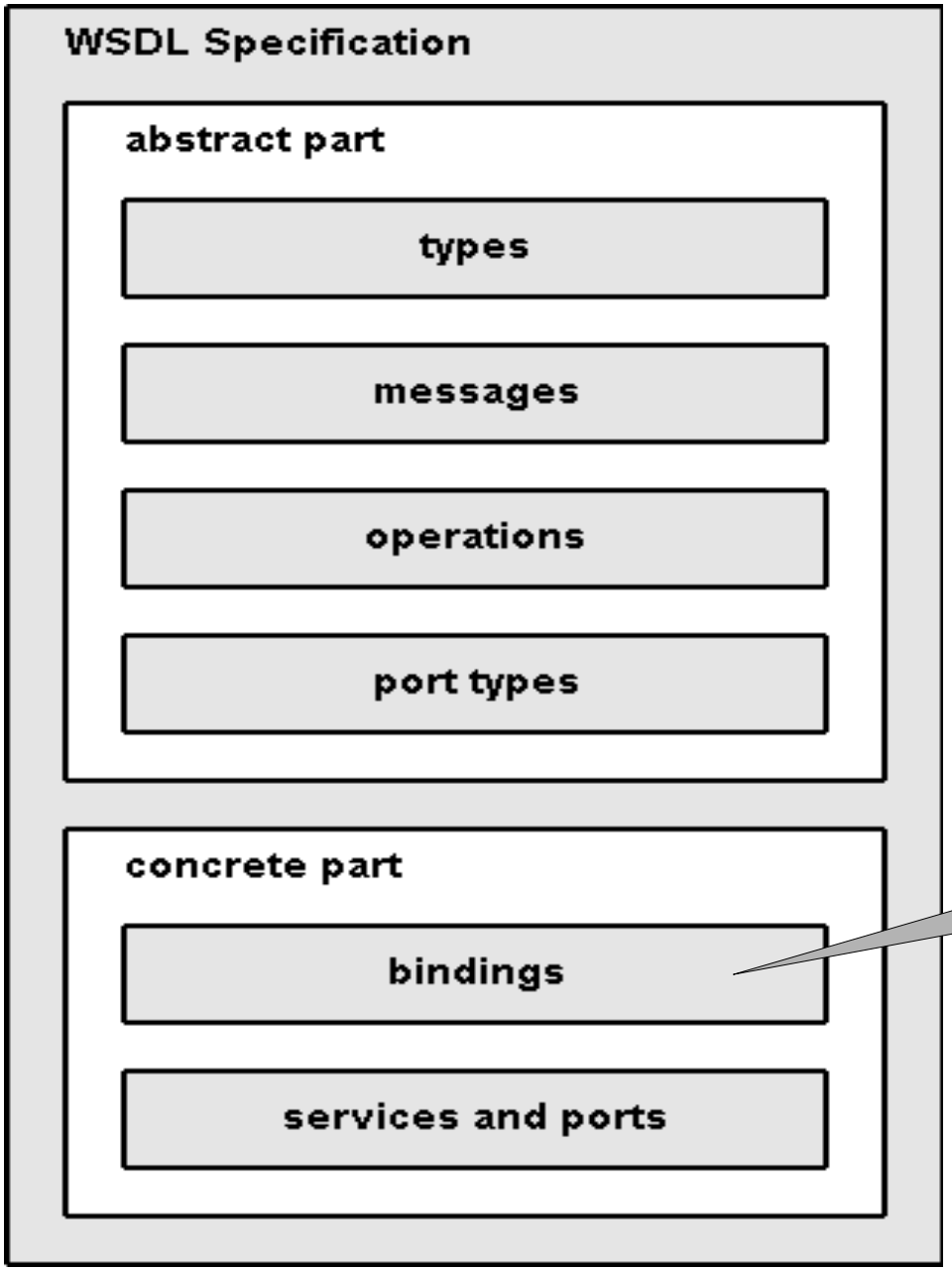
WSDL – Port types



Group operations into port types (roughly the interface to be implemented by a service)

Abstract part does not provide a real implementation for a service but only an abstract description

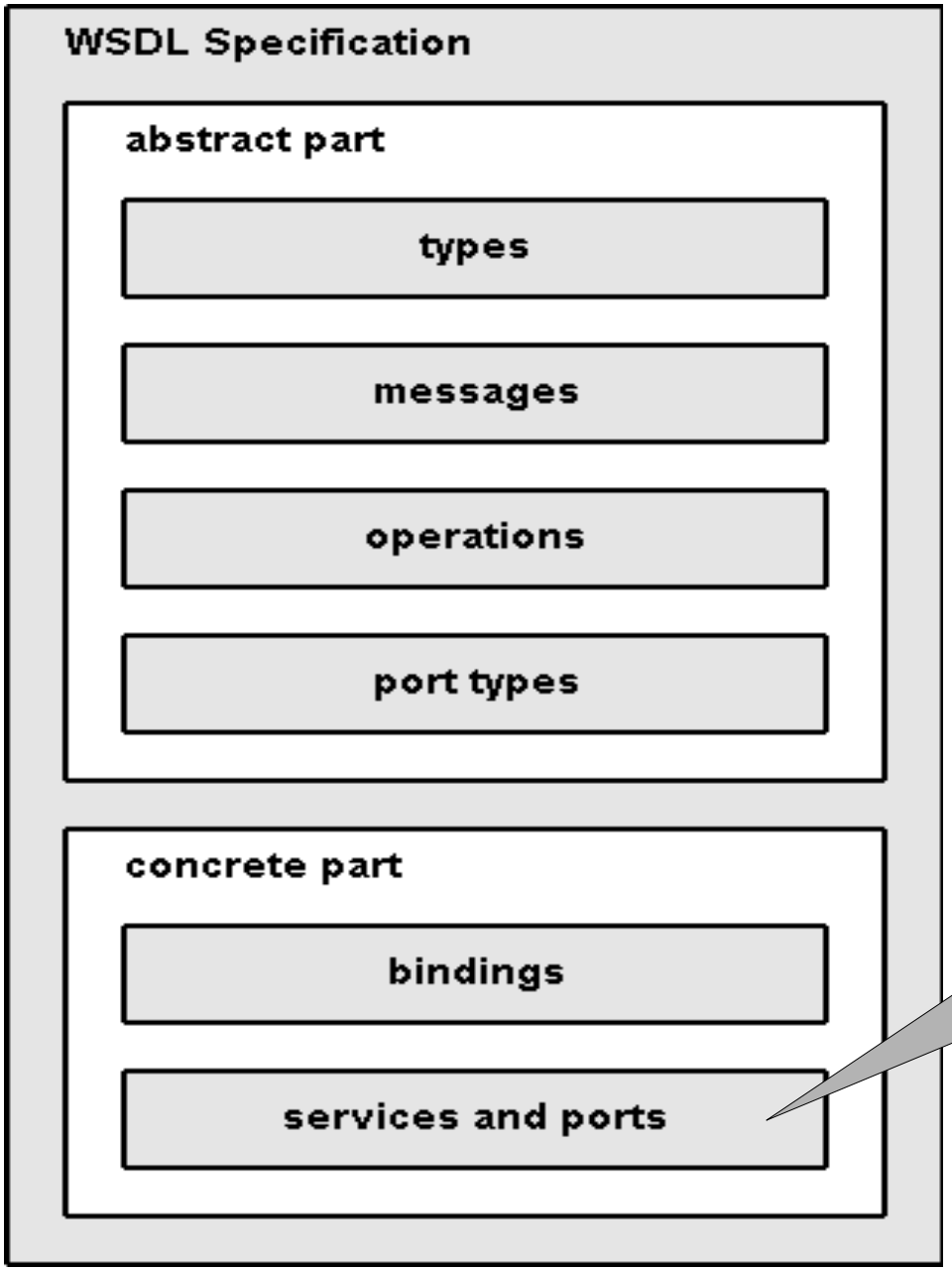
WSDL – Bindings



Concrete part wants to provide detail on a real implementation of a service. Where it can be accessed and which port type it implements

Specifies the message encoding and protocol binding (rpc, document) for a message.
It can specify also the transport protocol to be used

WSDL – Services and ports



Services logically group a set of ports. A port (also knows as endpoint) defines an access point (URI) for an interface binding.
Different ports in a service definition could provide different addresses for the same binding

WSDL final considerations

- Separation of abstract and concrete part intend to make easy reuse of definitions
- It is possible to define services that proactively start the interaction
- WSDL contains information that directly map on SOAP concepts so it can be used to directly derive SOAP messages
- How to use WSDL:
  - To describe and document an interface
  - To directly derive stubs
  - To provide info to semantically reason about the service (current standard is rather poor in this respect)

Demo

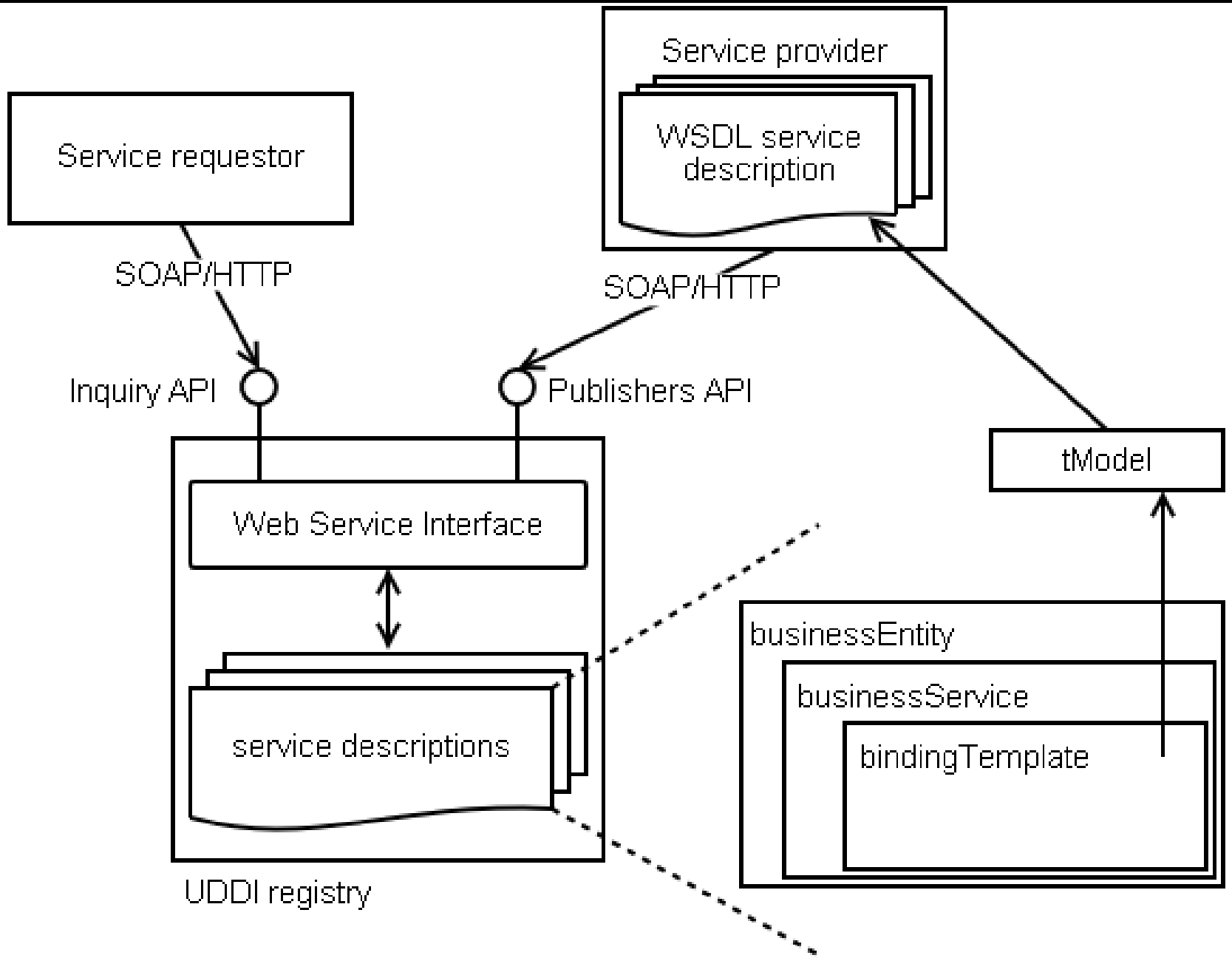
Univ. Description Discovery and Integration (UDDI)

- UDDI specifies a framework for describing and discovering services (generic not only related to Web Services)
  - Data structures
  - Application Programming Interface (API)
- UDDI can be considered as a particular service providing information on other services. A WSDL for UDDI service is defined in the UDDI spec
  - API divided in 6 groups (**Inquiry**, **Publishers**, Security, Custody, Subscription, Replication)
- Information can be organised and queried as:
  - White pages: providers
  - Yellow pages: categories
  - Green pages: characteristics

UDDI - data structures

- Four main groups of information and corresponding data structures are defined:
  - Business entity: it contains a list of organisations providing services and corresponding information
  - Business service: it contains the description of related web services offered by a business entity
  - BindingTemplate: it contains information on how a service can be used. In particular provide information concerning the address and references to tModels describing the service
  - tModel (technical Model): generic container for any kind of information. It is generally used to represent WSDL service interface

UDDI and WSDL



WS the future

- Standards
  - WS Security
  - WS Addressing
  - WS Transaction
- Orchestration: composition of services to provide a more complex service
- Choreographies: description of the interactions among a set of services to accomplish a given task
  - Open interesting scenarios for the future

Service Engineering

- Still a lot of research is required no prepacked solutions are available yet.
- Many challenging features for the different engineering activities (Requirements, Design, Coding, Testing, Management)
  - Distributed development process
  - Dynamic discovery and binding
  - No unique control on the application

Web Services Solutions (Java)

- Axis
  - SOAP container to be installed as a servlet on Tomcat
  - Java2WSDL and WSDL2Java
- jUDDI
  - Provides a servlet based implementation of UDDI2.0 (it requires the availability of a DB server and resides on top of Tomcat)
- UDDI4J
  - Provides a library that make easier to interact with UDDI servers
- Glassfish – opensource project from SUN (previously JWSDP) provide a whole platform for WS development and deployment