Contemporary SOA and Web Services

Ing. Nicola Zaghini

nicola.zaghini@unibo.it

may 2006

Outline

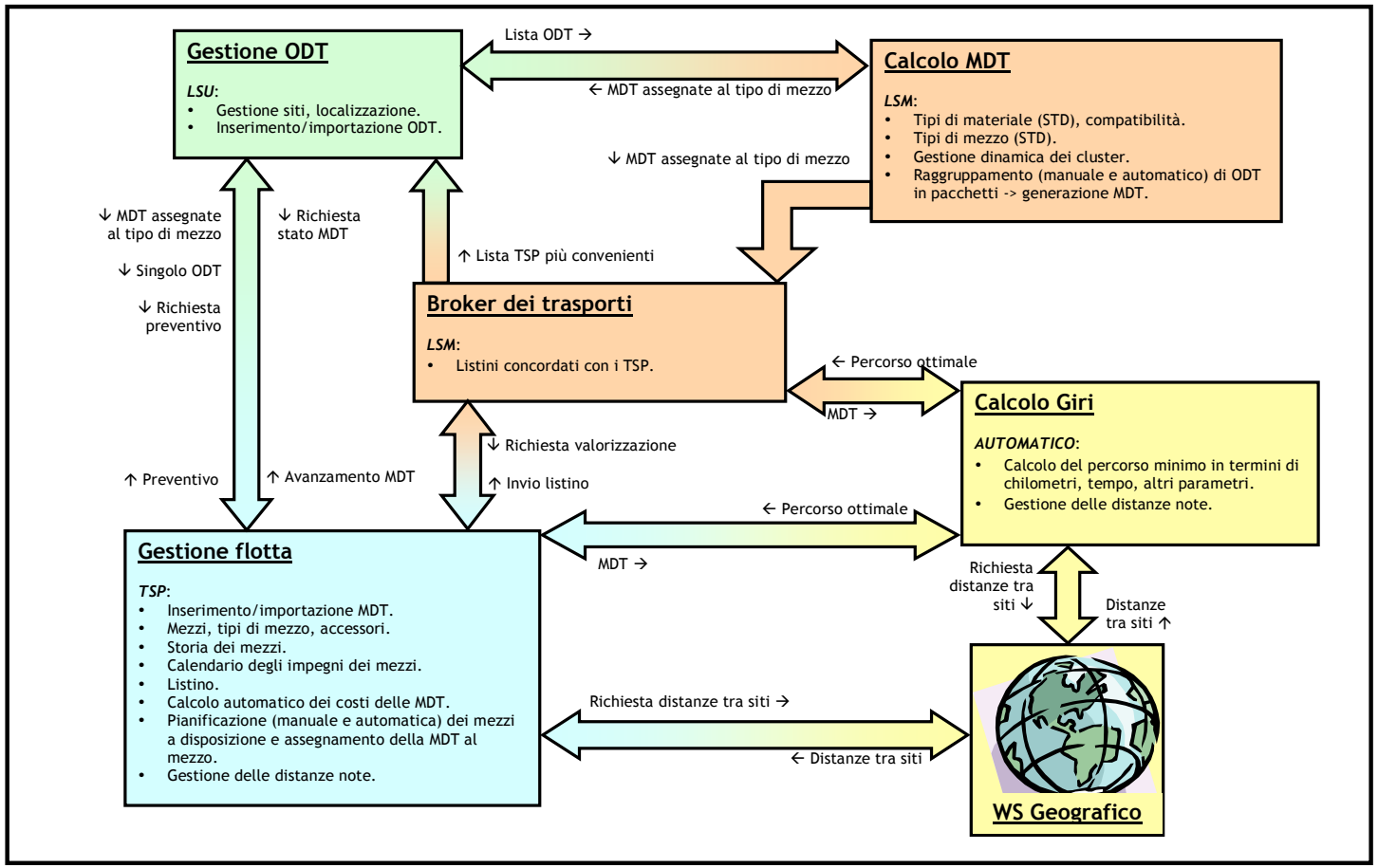
- SOA
  - Service orientation principle
  - Architecture
- Web Services
  - Proposal & framework
  - Service role
  - Service description (WSDL)
  - SOAP messaging framework
    - WS-Addressing
  - WSDL: which style?
- Message exchange patterns
- Overview on SOA Platform (J2EE)

SOA introduction

- SOA is Service Oriented Architecture
- Web Services and SOA are related but independent ...
- SOA is a new paradigm regard object oriented...
- why we need new paradigm?
- --> follow the example

SOA INTRODUCTION

WHICH DOMAIN? WHICH MODEL?



SOA ANALOGY

- THINK ABOUT AVERAGE COSMOPOLITAN CITY FULL OF BUSINESS COMPANY
- EACH COMPANY REPRESENT A SERVICE-ORIENTED BUSINESS -> SERVICE PROVIDED TO MULTIPLE CONSUMER
- COLLECTIVELY THEY ARE A BUSINESS COMMUNITY
- IT MAKE SENSE NOT HAVE A SINGLE BUSINESS OUTLET PROVIDING ALL SERVICES
- WE ACHIEVE AN ENVIRONMENT WITH DISTRIBUTED OUTLETS

SOA analogy

- Service-oriented architecture
  - a model in which automation logic is decomposed into smaller, distinct units of logic
  - collectively this units comprise a large piece of business automation logic (individually can be distributed)
- BUT We wont to
  - self-governing individual services -> independence between services (relatively)
  - MUST ensure that they adhere to certain baseline conventions

Service orientation

- Principle/1:
  - interoperability - of course
  - service contract - communication agreement
  - loose coupling - minimize dependencies, awareness of each other
  - abstraction - hiding logic form outside
  - autonomy - over the logic they encapsulate

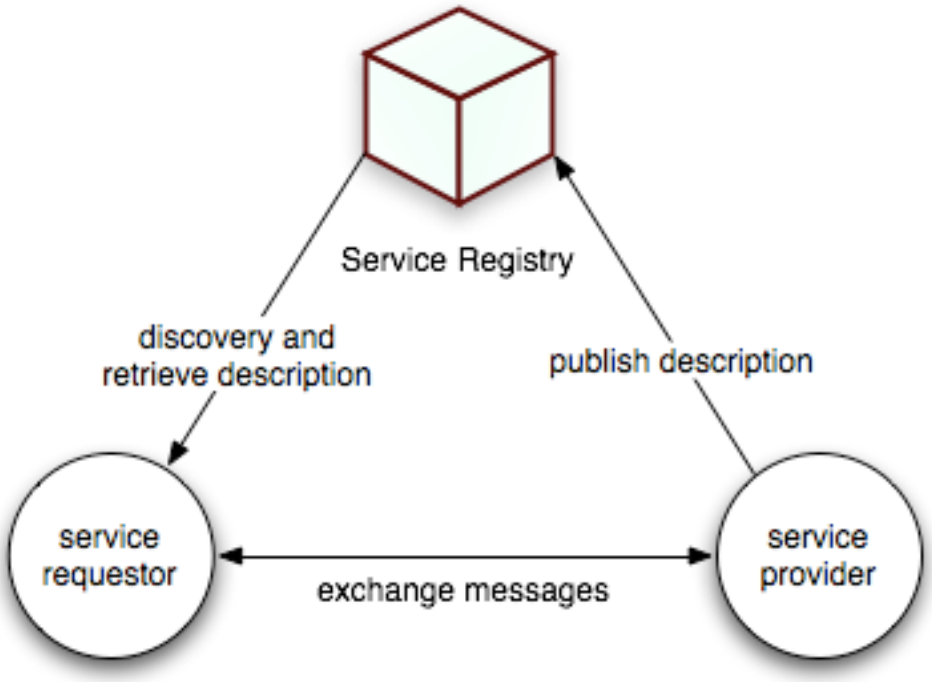
Service orientation

- Principle/2:
  - composability - collection coordinated to form composite service
  - reusability - logic divided into services to promote reuse
  - statelessness - minimize retaining info
  - discoverability - assessed by discovery mechanism
- which technology platform??
  - Web-Service! but carefully (how)

SOA vs Internet Arch.

- Client-server architecture vs. SOA
  - single-tier
  - two-tier
- Distributed internet architecture vs. SOA
  - RPC connection between components
- Hybrid Web Services architecture vs. SOA
  - wrapper encapsulating components

SOA Architecture



SOA Service

- Service as a unit of logic within a context
- service has a description
- loosely coupled relation
- we need messaging framework
- message as "independent units of communication"
- SOA KEYs: Services, Descriptions and Messages

The proposal of WS

- "Web Services provide a standard means of interoperating between different software application on a variety of platforms and frameworks"
- ... Web Services Architecture W3C working group
- they focus on Interoperability!

What is a Web Service?

- "WS is a software system designed to support interoperable machine-to-machine interaction over a network [..] using SOAP messages"
- "WS is an abstract notion that must be implemented by a concrete agent [..] the agent is the concrete piece of software that send and receive messages"
- the agent may or not be the service

Web services framework

- Web services framework is flexible and adaptable -> large in scope
- Characterized by/1:
  - an abstract (vendor-neutral) existence defined by standard implemented by (proprietary) technology platform
  - core building block that include Web services, service descriptions and messages
  - service description based on WSDL

Web services framework

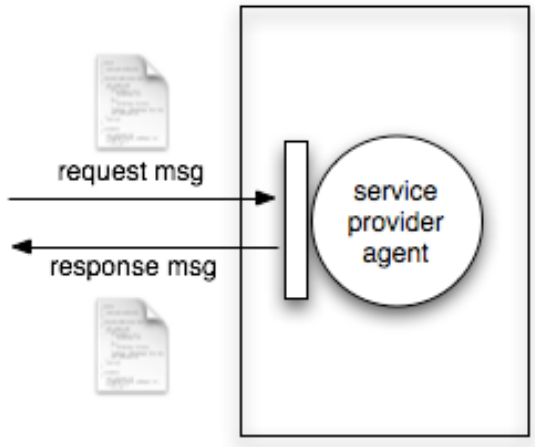
- Characterized by/1:
  - messaging framework comprised of SOAP technology and concept
  - service description registration and discovery (UDDI)
  - architecture that support message pattern
  - WS-* specifications

Service

- Services as application logic provider = implement a real world business functionality
- Service role (runtime classification)
  - depending on its processing responsibility in a given scenario (initiator - relayer - recipient of a message)

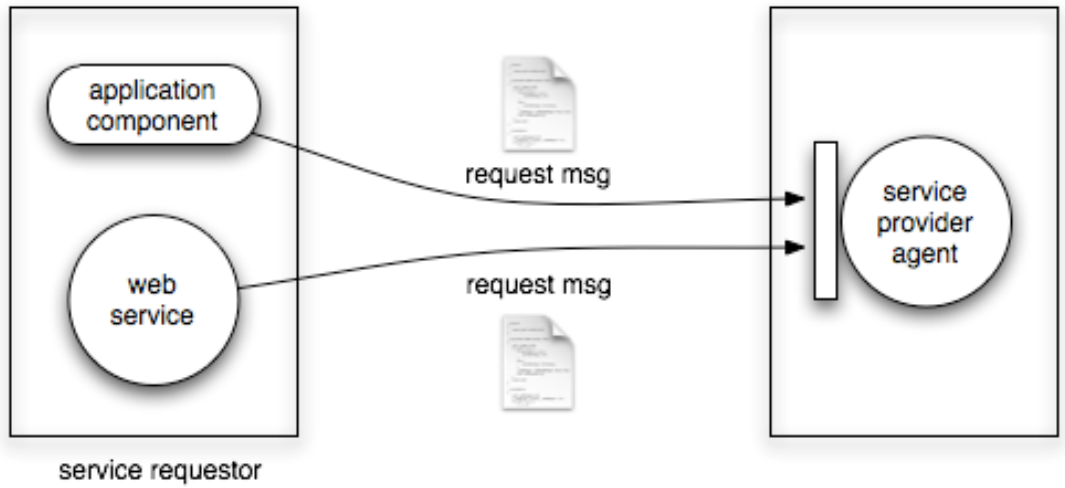
Service role

- **Service provider role**
  - is invoked via an external source
  - publish a service description (WSDL)



Service role

- **Service requester role**
  - invoke a service provider by sending msg
  - search the most suitable service provider studying available service descriptions



Service role

- **Service intermediator role**
  - also service and provider role for forwarding to destination
  - passive: without altering content
  - active: process and alter message content, typically will lock for a particular SOAP header
  - e.g.: policy rule, load balancing, ...
- **Service composition (member)**
  - Orchestration & choreography

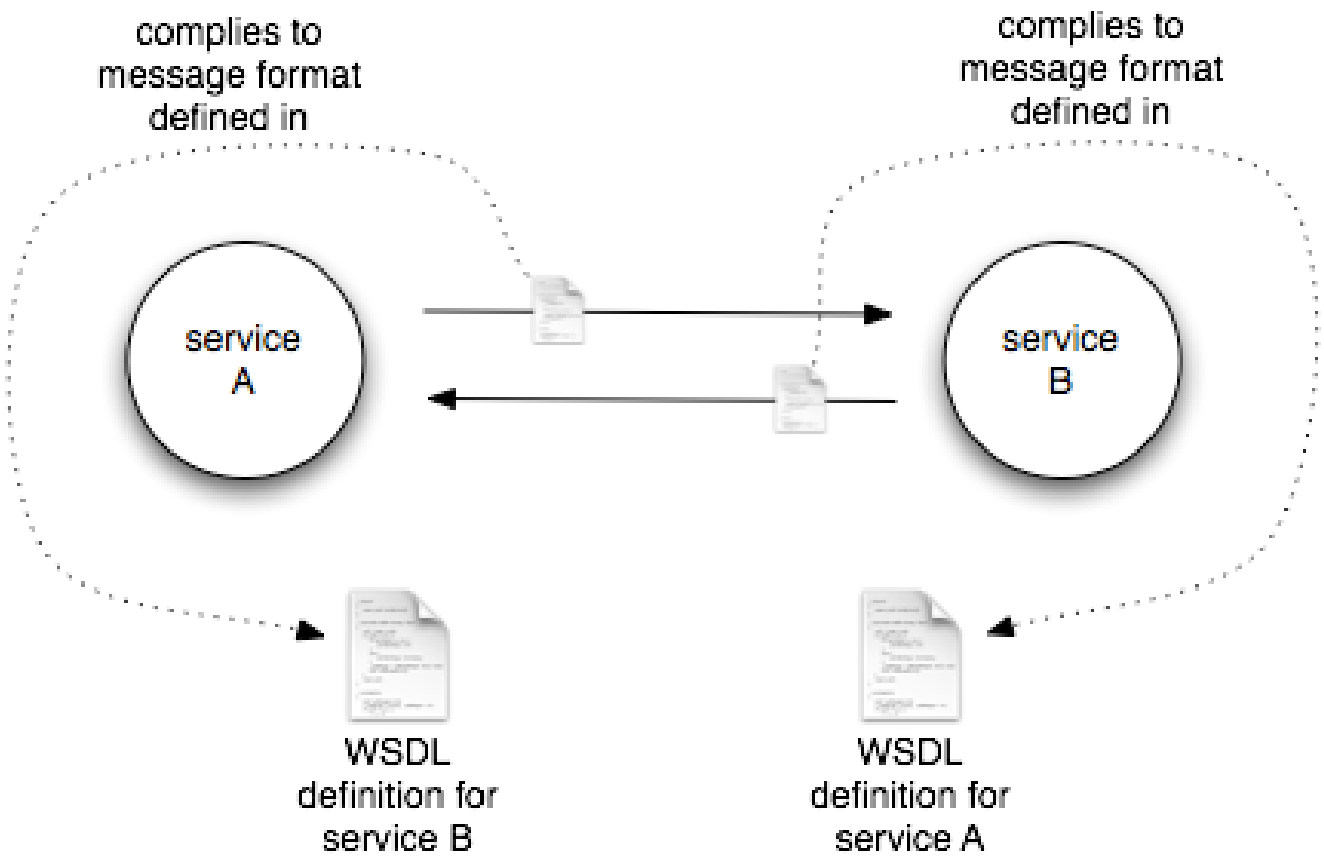
Service Models

- **Service classification based on the nature of the application logic provided**
  - Business service model: encapsulate a distinct set of business logic, is full autonomous but not limited to executing in isolation
  - Utility service model: a generic web service designed for potential reuse -generic and non-application specific nature
  - Controller service model: assembly and coordination of services

Service Description

- Service Description as "contract" that can be used to **build** and **validate** messages
  - what kind of operation can I invoke on service X? - requester role
  - what kind of operation/request can I accept? - provider role
- WSDL Web Service Description Language

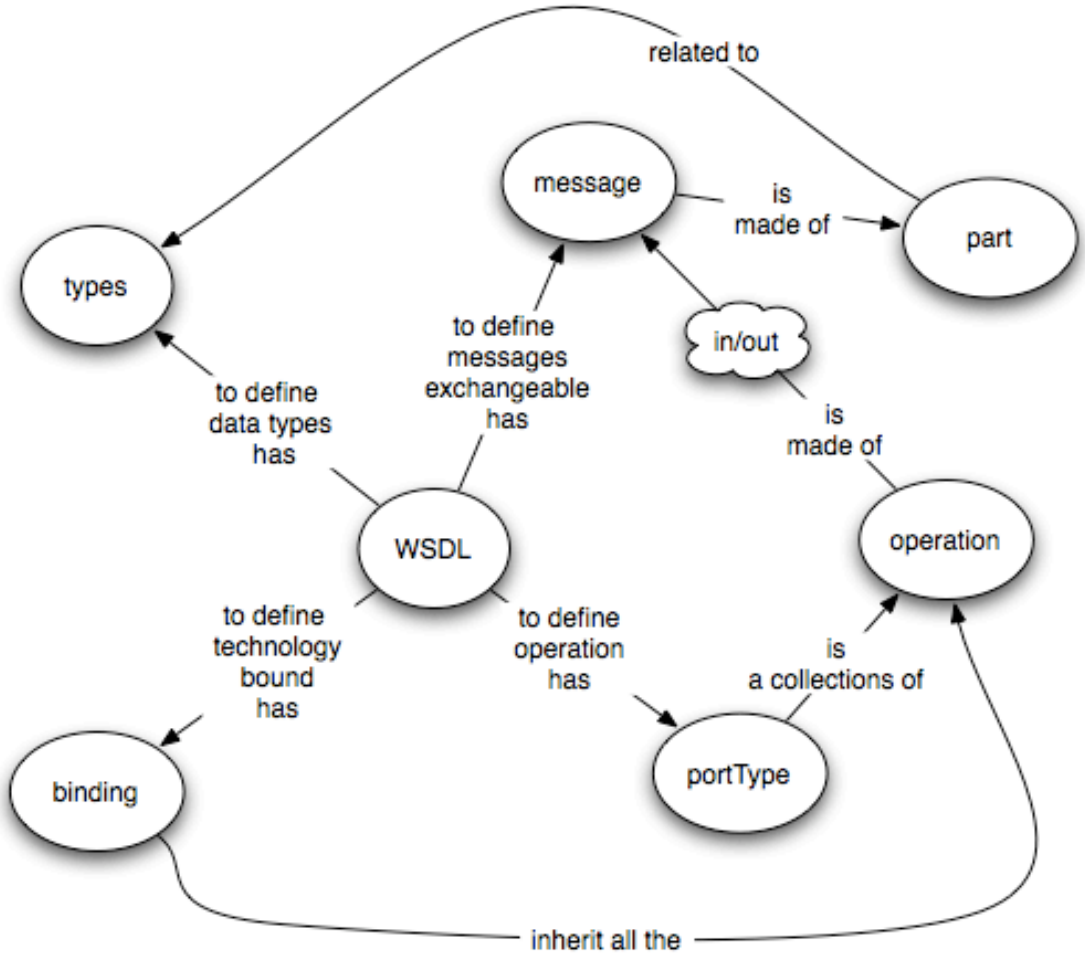
Service Description



Service Description

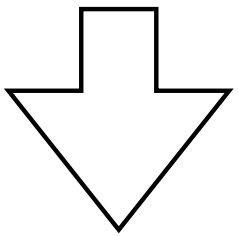
- **WSDL - Web Server Description Language**
  - Abstract description
    - interface characteristic without technology reference
  - Concrete description
    - connection to some real, implemented technology

Service Description



WSDL

- **Abstract Description - high level view of the service**
  - definition - root element declaring namespace
  - types - where XML Schema is placed, to simple data to complex business document
  - example -> echo and ping operations



WSDL

- **Abstract Description**
  - messages designed to receive or transmit

```
<wsdl:message name="echoRequestMessage">
   <wsdl:part name="part1" element="ns1:echoRequest"/>
</wsdl:message>

<wsdl:message name="echoResponseMessage">
   <wsdl:part name="part1" element="ns1:echoResponse"/>
</wsdl:message>

<wsdl:message name="pingRequestMessage">
   <wsdl:part name="part1" element="ns1:pingRequest"/>
</wsdl:message>
```

WSDL

```
<wsdl:types>
    <xs:schema targetNamespace="http://org.apache.axis2/xsd"
        elementFormDefault ="unqualified" attributeFormDefault="unqualified">

        <xs:element name="pingRequest">
            <xs:complexType>
                <xs:sequence>
                    <xs:element type="xs:anyType" name="element"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>

        <xs:element name="echoRequest">
            <xs:complexType>
                <xs:sequence>
                    <xs:element type="xs:anyType" name="element"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="echoResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element type="xs:anyType" name="return"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:schema>
</wsdl:types>
```

WSDL

- **Abstract Description - high level view of the service**

  - portType (collection of) -> operation

```
<wsdl:portType name="MyServicePort">
   <wsdl:operation name="echo">
      <wsdl:input message="tns:echoRequestMessage"/>
      <wsdl:output message="tns:echoResponseMessage"/>
   </wsdl:operation>
      <wsdl:operation name="ping">
      <wsdl:input message="tns:pingRequestMessage"/>
   </wsdl:operation>
</wsdl:portType>
```

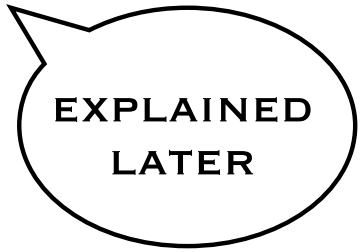
  - operation is not (only) a method mapping

WSDL

Concrete Description

binding -> concrete binding to SOAP

```
<wsdl:binding name="MyServiceBinding" type="tns:MyServicePort">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <wsdl:operation name="echo">
        <soap:operation soapAction="echo" />
        <wsdl:input>
            <soap:body use="literal" namespace="http://www.org.apache.axis2"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" namespace="http://www.org.apache.axis2"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```



WSDL

Concrete Description

- service -> physical address at which access service
- port -> location information

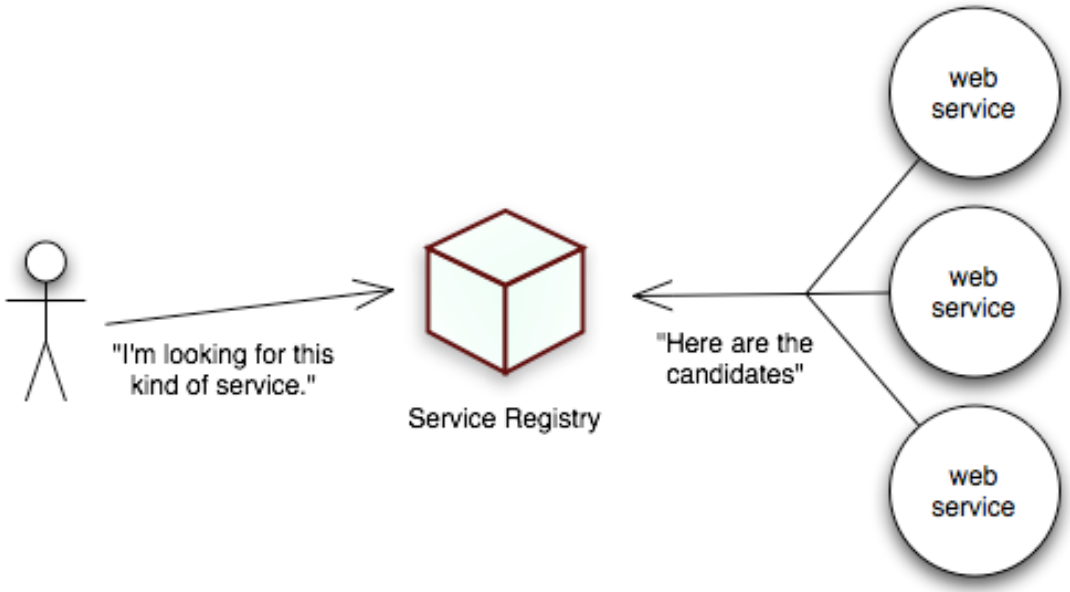
```
<wsdl:service name="MyService">

   <wsdl:port name="MyServicePortType0"
                                binding="tns:MyServiceBinding">

         <soap:address location="http://localhost:8080/MyService"/>

   </wsdl:port>
</wsdl:service>
```

WSDL Semantic (pills)

- **...and what about semantic**
  - how a service behaves under certain conditions
  - how service will respond to specific conditions
  - what specific tasks the service is most suited for
- **OWL - OWLS** (think about)
  - no standardized solution yet

UDDI (pills)

- **Service description advertisement and discovery**
  - UDDI V2.0 specifications approved as an OASIS Standard



- **Not yet commonly implemented**

SOAP

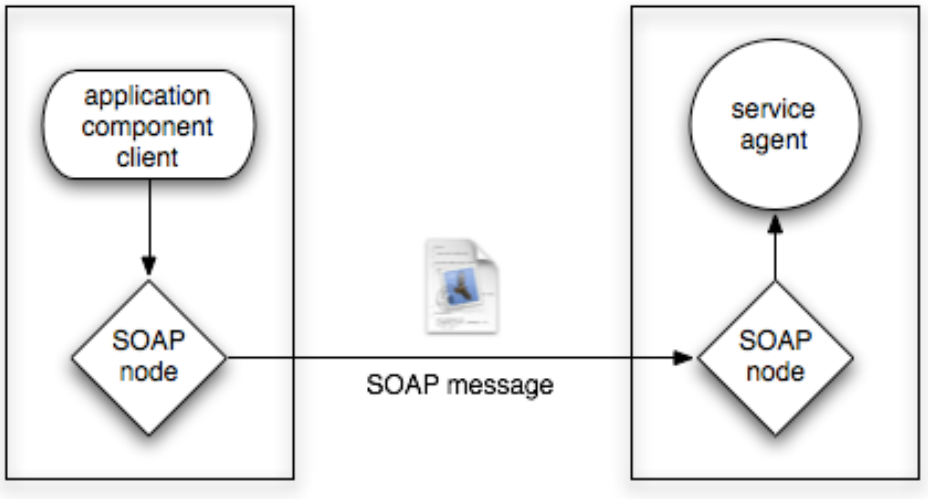
- Messaging Framework Specification
- Simple Object Access Protocol
  - originally designed to replace proprietary RPC protocols -> serialization of object
  - now the purpose is to define a standard message format !!!
  - extremely flexible and extensible
- The RPC-Style messages runs contrary to the SOA principle.

SOAP

- Each message packaged in ENVELOPE
  - Header - area dedicated to hosting meta information --> WS-*
  - Body - XML formatted data, is the message payload
- Message have high level of independence --> robustness and extensibility
- Fundamental in a loosely coupled env.

SOAP

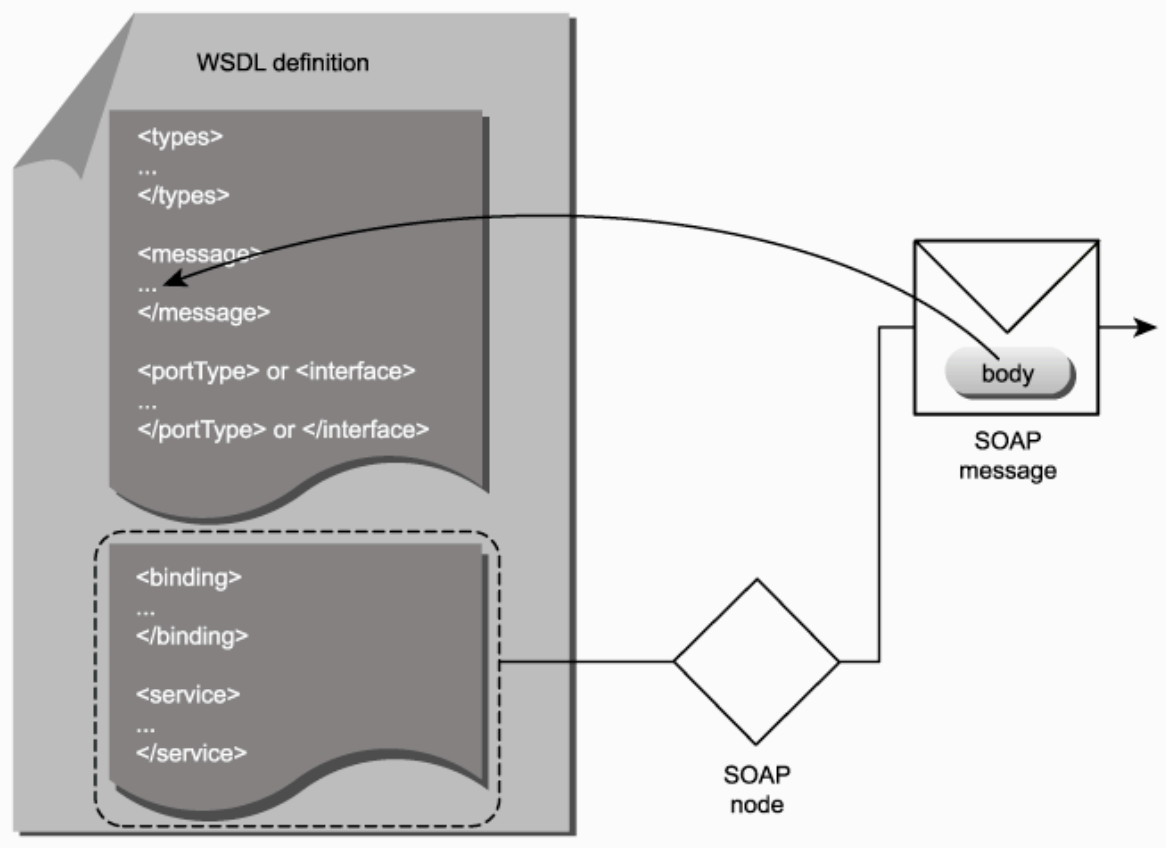
- The SOAP Nodes
  - sender
  - receiver
  - intermediary
  - initial
  - ultimate



- Remember the model!!

SOAP & WSDL

- Processing of SOAP message using concrete definition



WS-* EXTENSIONS

- **WS-Addressing**
  - standardize the representation of service endpoint locations and unique correlation values that tie together request and response exchanges
  - Relation to other WS-* extensions

WS-* EXTENSIONS

- **WS-Addressing**

WS Addressing

- **Endpoint reference element**
  - assist in providing service interface information
- **Message Information Header element**

Element	Description
MessageID	An element used to hold a unique message identifier, most likely for correlation purposes. This element is required if the ReplyTo or FaultTo elements are used.
RelatesTo	This is also a correlation header element used to explicitly associate the current message with another. This element is required if the message is a reply to a request.
ReplyTo	The reply endpoint (of type EndpointReference) used to indicate which endpoint the recipient service should send a response to upon receiving the message. This element requires the use of MessageID.
From	The source endpoint element (of type EndpointReference) that conveys the source endpoint address of the message.
FaultTo	The fault endpoint element (also of type EndpointReference) that provides the address to which a fault notification should be sent. FaultTo also requires the use of MessageID.
To	The destination element used to establish the endpoint address to which the current message is being delivered.
Action	This element contains a URI value that represents an action to be performed when processing the MI header.

WS Addressing

Case Study

```
<Envelope
   xmlns="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:wsa=
      "http://schemas.xmlsoap.org/ws/2004/08/addressing"
   xmlns:app="http://www.xmltc.com/railco/...">
   <Header>
      <wsa:Action>
         http://www.xmltc.com/tls/vp/submit
      </wsa:Action>
      <wsa:To>
         http://www.xmltc.com/tls/vp/...
      </wsa:To>
      <wsa:From>
         <wsa:Address>
            http://www.xmltc.com/railco/ap1/...
         </wsa:Address>
         <wsa:ReferenceProperties>
            <app:id>
               unn:AFJK323llws
            </app:id>
         </wsa:ReferenceProperties>
         <wsa:ReferenceParameters>
            <app:sesno>
               22322447
            </app:sesno>
         </wsa:ReferenceParameters>
      </wsa:From>
      <wsa:MessageID>
         uuid:243234234-43gf433
      </wsa:MessageID>
```

WS Addressing

```
<wsa:ReplyTo>
   <wsa:Address>
      http://www.xmltc.com/railco/ap2/
   </wsa:Address>
   <wsa:ReferenceProperties>
      <app:id>
         unn:AFJK323llws
      </app:id>
   </wsa:ReferenceProperties>
   <wsa:ReferenceParameters>
      <app:sesno>
         22322447
      </app:sesno>
   </wsa:ReferenceParameters>
</wsa:ReplyTo>
<wsa:FaultTo>
   <wsa:Address>
      http://www.xmltc.com/railco/ap-err/
   </wsa:Address>
   <wsa:ReferenceProperties>
      <app:id>
         unn:AFJK323llws
      </app:id>
   </wsa:ReferenceProperties>
   <wsa:ReferenceParameters>
      <app:sesno>
         22322447
      </app:sesno>
   </wsa:ReferenceParameters>
</wsa:FaultTo>
</Header>
<Body>
   ...
</Body>
</Envelope>
```

Which style of WSDL should I use?

- In relation to WSDL binding to SOAP
  - RPC/encoded
  - RPC/literal
  - Document/encoded
  - Document/literal
- Following the example
  - myMethod operation with parameters (integer x, float y)

Which style of WSDL should I use?

- **RPC/encoded - void myMethod(int x, float y)**

WDSL

```
<message name="myMethodRequest">
   <part name="x" type="xsd:int"/>
   <part name="y" type="xsd:float"/>
</message>


<portType name="PT">
   <operation name="myMethod">
      <input message="myMethodRequest"/>
   </operation>
</portType>

<binding .../>
```

SOAP

```
<soap:envelope>
   <soap:body>
      <myMethod>
         <x xsi:type="xsd:int">5</x>
         <y xsi:type="xsd:float">5.0</y>
      </myMethod>
   </soap:body>
</soap:envelope>
```

OVERHEAD

OP. NAME

not WS-I compliant

Which style of WSDL should I use?

- **RPC/literal - void myMethod(int x, float y)**

WDSL

```
<message name="myMethodRequest">
   <part name="x" type="xsd:int"/>
   <part name="y" type="xsd:float"/>
</message>


<portType name="PT">
   <operation name="myMethod">
      <input message="myMethodRequest"/>
   </operation>
</portType>

<binding .../>
```

SOAP

```
<soap:envelope>
   <soap:body>
      <myMethod>
         <x >5</x>
         <y >5.0</y>
      </myMethod>
   </soap:body>
</soap:envelope>
```

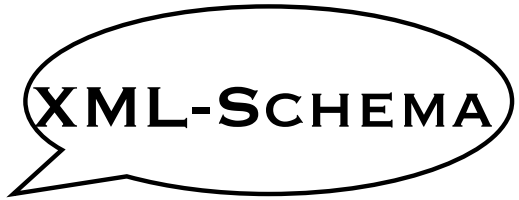
OP. NAME

WS-I compliant

Which style of WSDL should I use?

- Document/literal

WDSL




```
<types>
   <schema>
      <element name="xElement" type="xsd:int"/>
      <element name="yElement" type="xsd:float"/>
   </schema>
</types>

<message name="myMethodRequest">
   <part name="x" element="xElement"/>
   <part name="y" element="yElement"/>
</message>
```

SOAP

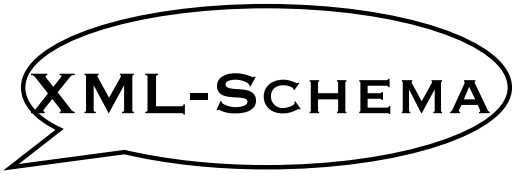

```
<soap:envelope>
   <soap:body>
      <xElement>5</xElement>
      <yElement>5.0</yElement>
   </soap:body>
</soap:envelope>
```

not WS-I compliant

Which style of WSDL should I use?

- **Document/literal wrapped**

WDSL




```
<types>
   <schema>
      <element name="myMethod">
         <complexType>
            <sequence>
               <element name="x" type="xsd:int"/>
               <element name="y" type="xsd:float"/>
            </sequence>
         </complexType>
      </element>
    </schema>
</types>

<message name="myMethodRequest">
   <part name="parameters" element="myMethod"/>
</message>
```

SOAP

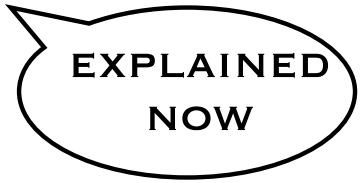
```
<soap:envelope>
   <soap:body>
      <myMethod>
         <x>5</x>
         <y>5.0</y>
      </myMethod>
    </soap:body>
</soap:envelope>
```

WS-I compliant

WSDL BINDING SOAP

- **CONCRETE DESCRIPTION**
  - BINDING -> CONCRETE BINDING TO SOAP

```
<wsdl:binding name="MyServiceBinding" type="tns:MyServicePort">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/
    >
    <wsdl:operation name="echo">
        <soap:operation soapAction="echo" />
        <wsdl:input>
            <soap:body use="literal" namespace="http://www.org.apache.axis2"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" namespace="http://www.org.apache.axis2"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```



MEPs

message exchange patterns

- **Interaction** between services
  - as result of engineering interaction
- A group of already mapped out sequence for the exchange of messages
- Simple MEPs as building block for Complex MEPs

MEPs

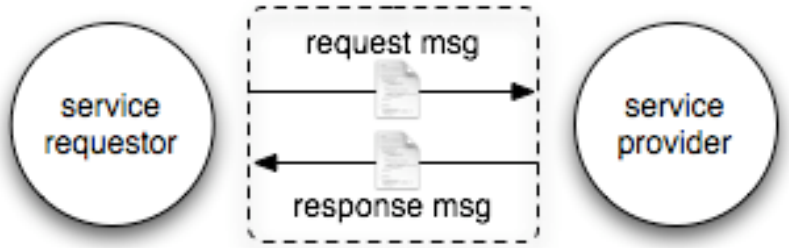
MESSAGE EXCHANGE PATTERNS

- **PRIMITIVE MEPS**
  - REQUEST-RESPONSE
    - CORRELATION CONCEPT
    - DEFINE SYNCHRONOUS COMMUNICATION (ALSO ASYNCHRONOUS)
  - FIRE AND FORGET
    - SINGLE DESTINATION - MULTICAST -BROADCAST

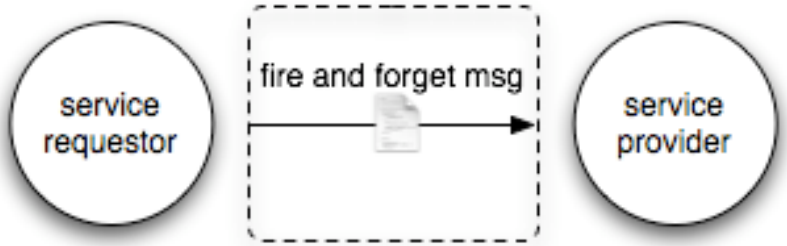
MEPs

MESSAGE EXCHANGE PATTERNS

- **PRIMITIVE MEPs**



message exchange pattern

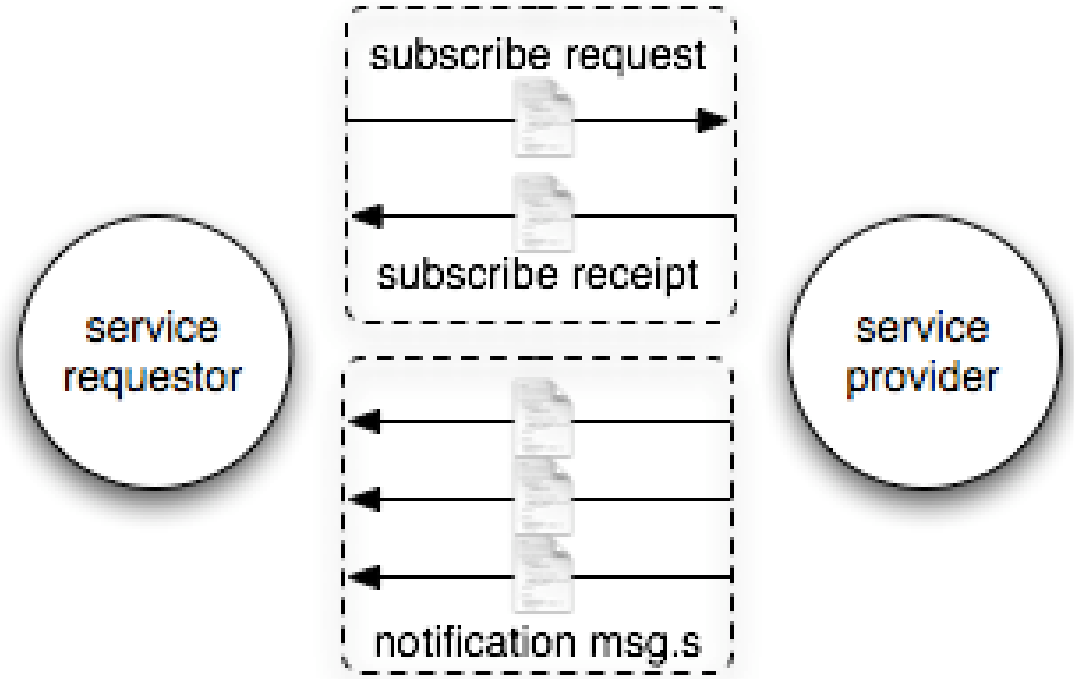


message exchange pattern

MEPs

MESSAGE EXCHANGE PATTERNS

- **COMPLEX MEPs --> E.G.: PUBLISH-AND-SUBSCRIBE**



MEPs

message exchange patterns

- Blocking or not blocking ?
  - only for request-response pattern
  - in a dual transport like Http is a client matter -> but Long Time Transaction?
  - two separate transport connection for request and response is a client and service matter --> WS-*
  - WS-Addressing

MEPs And WSDL

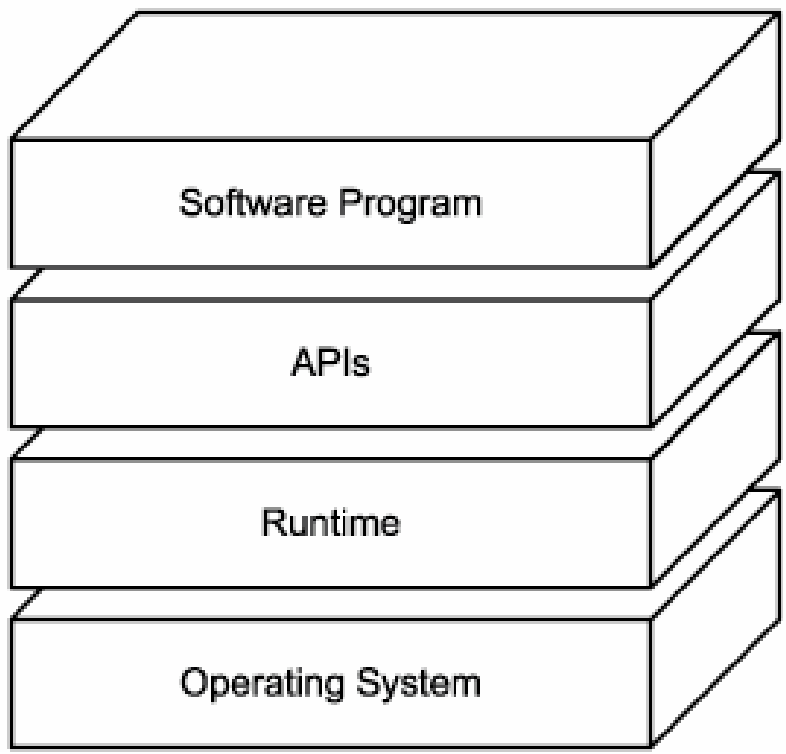
- In WSDL 1.1 terms
  - Request-Response -> WS-I ok
  - Solicit-Response -> WS-I ok
  - One-way operation -> WS-I ko
  - Notification Operation -> WS-I ko
- WS-I delivers practical guidance, best practices and resources for developing interoperable Web services solutions. http://www.ws-i.org/

MEPs And WSDL

- In WSDL 2.0 terms
  - In-out pattern = Request-Response
  - out-in pattern = Solicit-Response
  - In-only pattern = One-way operation
  - Out-only pattern = Notification Operation
  - Robust in-only -> fault message from receiver are allowed
  - In-optional-out pattern -> the response is optional

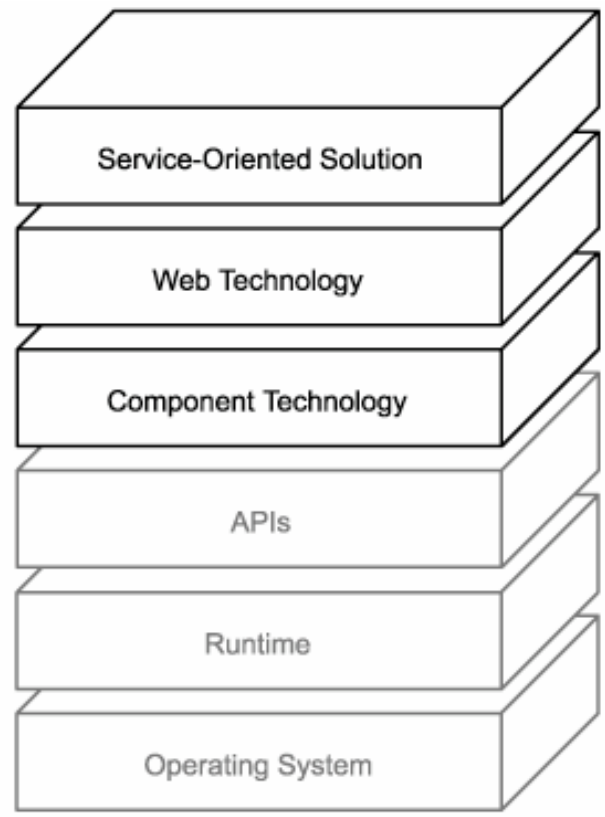
SOA Platform

- Basic platform building block

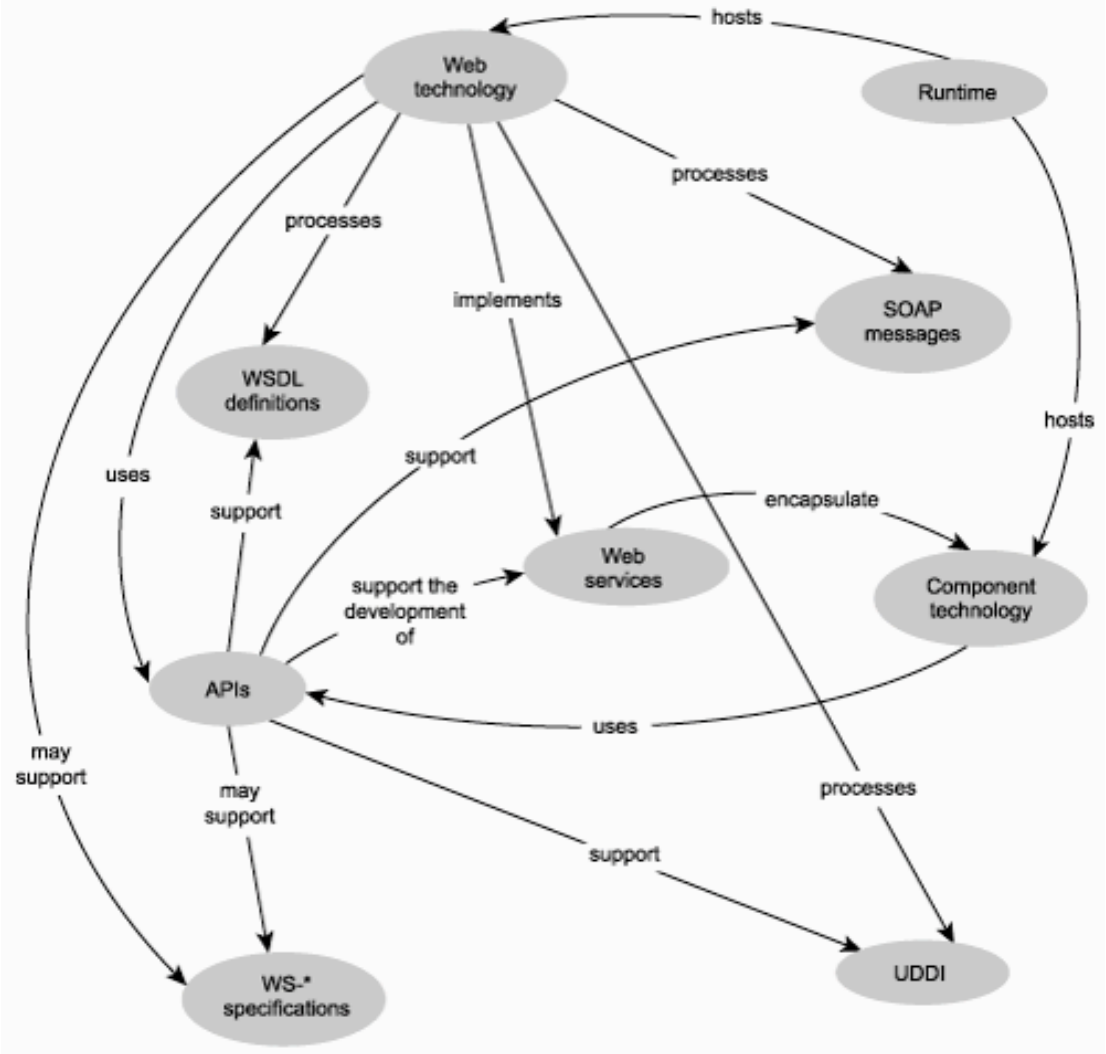


SOA Platform

- Common SOA platform layer



SOA Platform



Service Processing Task

- Service provider are expected to
  - supply a public interface (WSDL)
  - receive a SOAP message from requester
  - processing the header block within SOAP m.
  - validate and parse payload of SOAP m.
  - transform payload in a different format
  - encapsulate business processing logic

Service Processing Task

- Service provider are expected to
  - assemble SOAP message containing the response to the original request SOAP
    - WS-Addressing and correlation
  - transform the contents of the message back into the form expected by the requestor
  - transmit the response SOAP

Service Processing Task

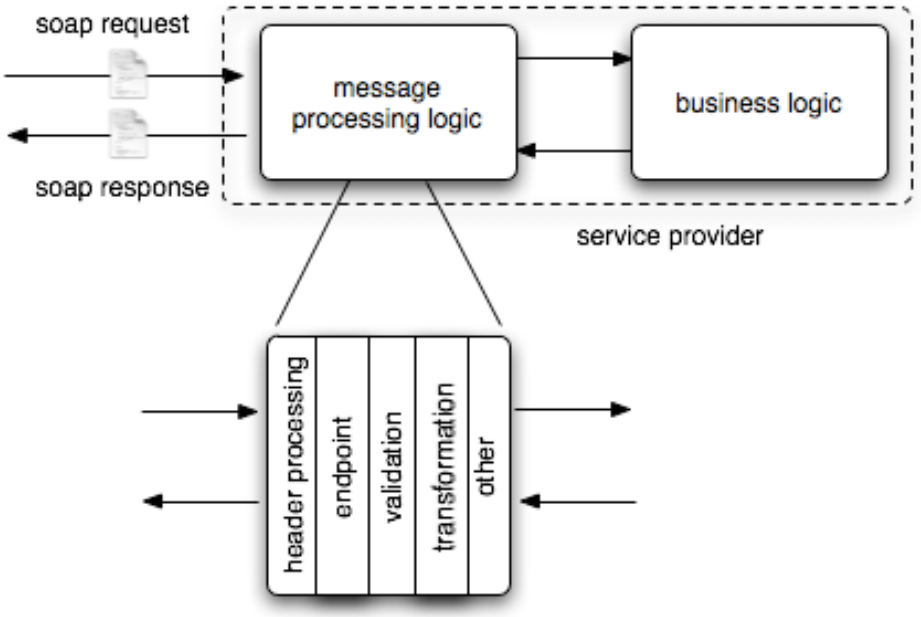
- Service requester are expected to
  - contain business processing logic that calls a service provider
  - interpret a service provider's WSDL definition
  - assemble a SOAP request in compliance with service provider WSDL definition
  - trasmitt SOAP request message to service provider

Service Processing Task

- Service requester are expected to
  - receive a SOAP response message
  - validate and parse the SOAP response
  - transform payload in a different format
  - process SOAP header block

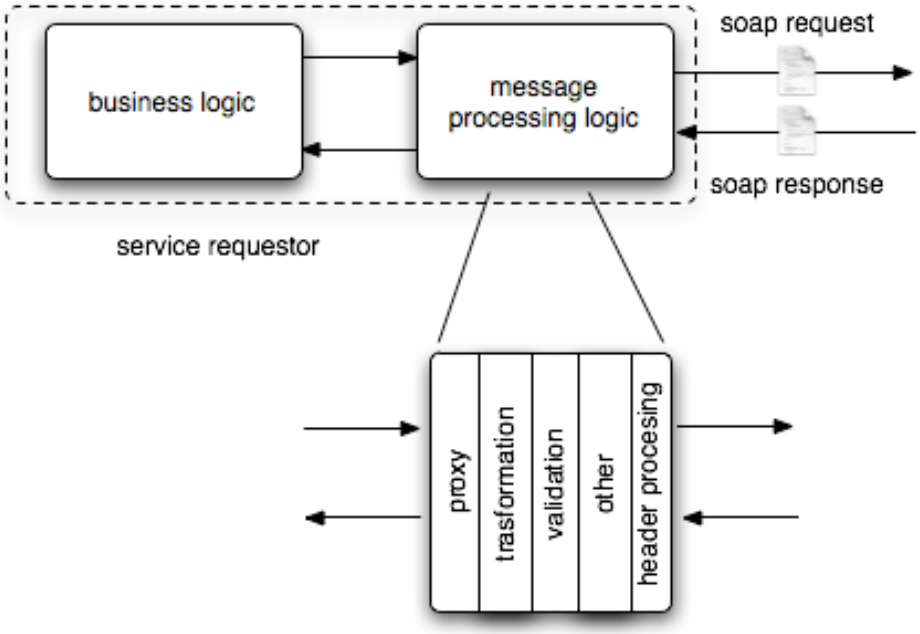
Service Processing Task

- Service provider

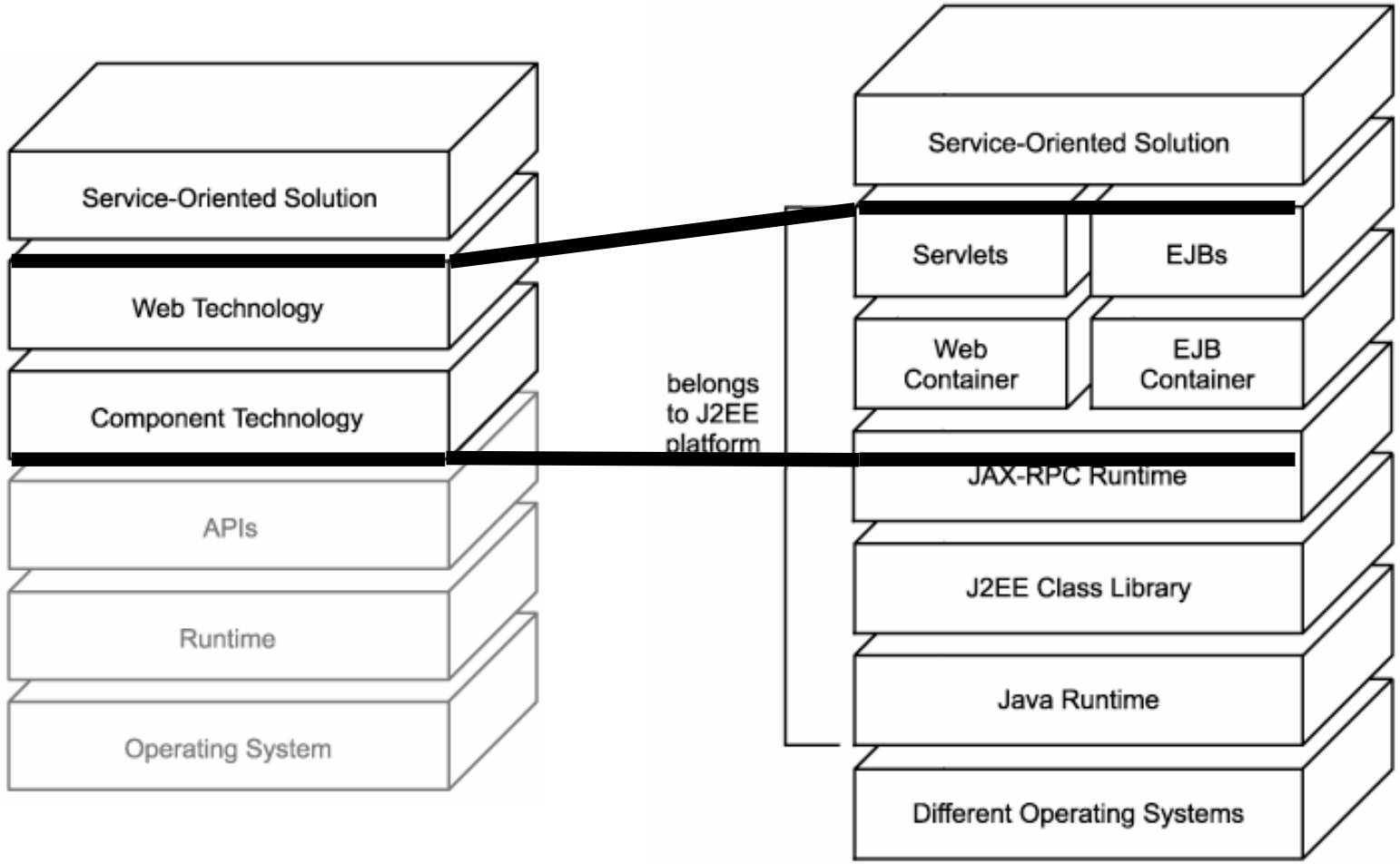


Service Processing Task

- Service requester

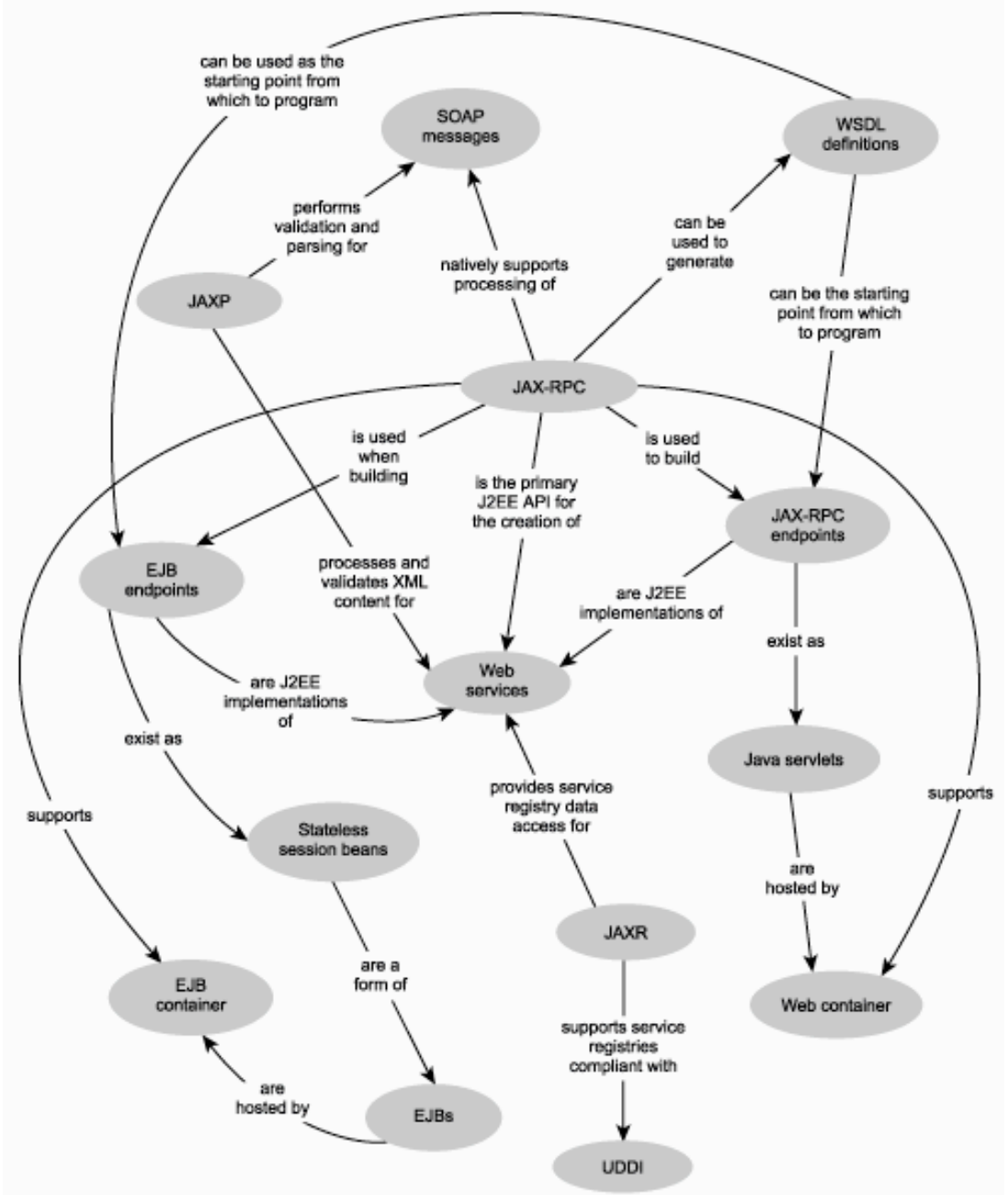


SOA SUPPORT IN J2EE

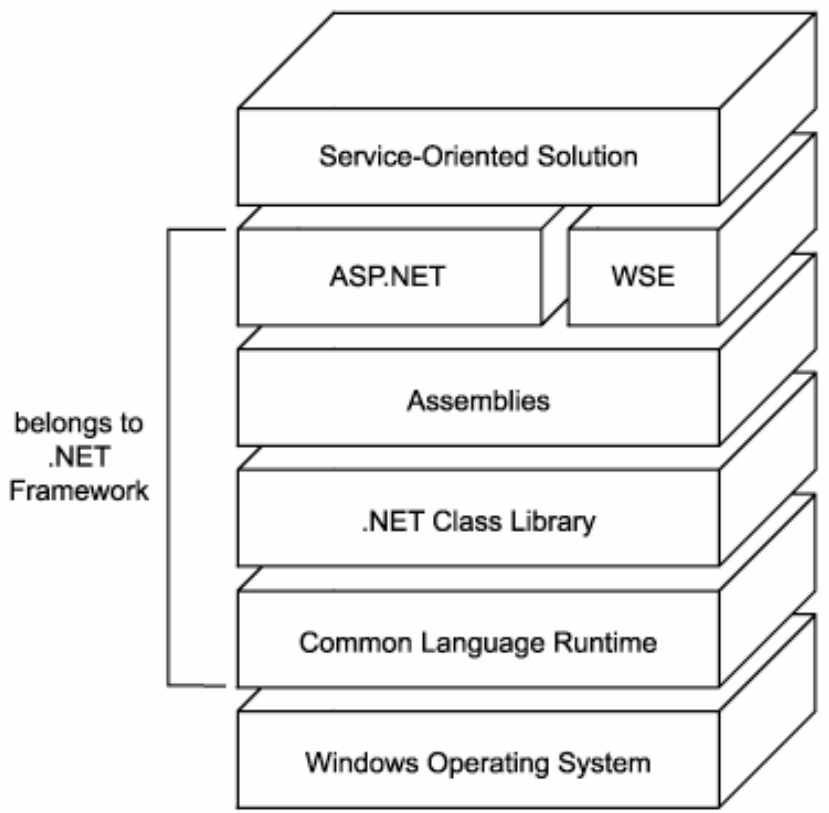


SOA SUPPORT IN J2EE

- **Java API for XML Processing (JAXP)** This API is used to process XML document content using a number of available parsers. Both Document Object Model (DOM) and Simple API for XML (SAX) compliant models are supported, as well as the ability to transform and validate XML documents using XSLT stylesheets and XSD schemas.
- **Java API for XML-based RPC (JAX-RPC)** The most established and popular SOAP processing API, supporting both RPC-literal and document-literal request-response exchanges and one-way transmissions. Example packages that support this API include:
- **Java API for XML Registries (JAXR)** An API that offers a standard interface for accessing business and service registries. Originally developed for ebXML directories, JAXR now includes support for UDDI.
- **Java API for XML Messaging (JAXM)** An asynchronous, document-style SOAP messaging API that can be used for one-way and broadcast message transmissions (but can still facilitate synchronous exchanges as well).
- **SOAP with Attachments API for Java (SAAJ)** Provides an API specifically for managing SOAP messages requiring attachments. The SAAJ API is an implementation of the SOAP with Attachments (SwA) specification.
- **Java Architecture for XML Binding API (JAXB)** This API provides a means of generating Java classes from XSD schemas and further abstracting XML-level development.
- **Java Message Service API (JMS)** A Java-centric messaging protocol used for traditional messaging middleware solutions and providing reliable delivery features not found in typical HTTP communication.


can be used as the starting point from which to program
SOAP messages
WSDL definitions
performs validation and parsing for
natively supports processing of
can be used to generate
JAXP
can be the starting point from which to program
JAX-RPC
is used when building
is the primary J2EE API for the creation of
is used to build
JAX-RPC endpoints
EJB endpoints
processes and validates XML content for
are J2EE implementations of
exist as
are J2EE implementations of
Web services
exist as
Java servlets
provides service registry data access for
supports
supports
Stateless session beans
are hosted by
JAXR
are a form of
EJB container
Web container
supports service registries compliant with
are hosted by
EJBs
UDDI

SOA Platform



Bibliography

- Web Service Architecture W3C working group
  - http://www.w3.org/TR/ws-arch
- Service-Oriented Architecture Concept, Technology, and Design
  - Thomas Erl - Prentice Hall PTR
- Some article from
  - http://www-128.ibm.com/developerworks/webservices