Lesson 5 – Web Service Interface Definition (Part II)

Service Oriented Architectures Security

Module 1 - Basic technologies

Unit 3 – WSDL

Ernesto Damiani

Università di Milano

Controlling the style (1)

- The style of the SOAP message is specified in a WSDL document in the binding section

```
<wsdl:binding name="…" type="tns:port
type name"> <soap:binding
style="rpc|document"> … <soap:binding>
</wsdl:binding>
```

Controlling the style (2)

- The style is also reflected in the

<wsdl:message> element, which can have:

  - For REST style, at most 1 <wsdl:partelement="…">
  - For RPC style, any number of <wsdl:parttype="…"> elements
  - With RPC style you only need an <wsdl:types> element to define the complex types used by the parameters
    - Note: REST style is more general as it can implement the RPC style by using an appropriate XML Schema

Controlling encoding (1)

- The serialized data content of a SOAP message can also be encoded in different ways:
  - Literal (follow the XML schema definition of the WSDL)
  - SOAP encoded (follow Section 5 of the SOAP 1.1 spec)
- The encoding is specified in the WSDL binding section, for each message exchanged as part of each operation, as follows

```
<wsdl:input>
<soap:body use="literal"/>
<soap:body use="encoded"
   encodingStyle="http://schemas.xmlsoap.org/soap/
   encoding"/>
</wsdl:input>
```

Controlling encoding (2)

- Style and encoding are usually paired (REST/Literal and RPC/Encoded). However, they are orthogonal and can be selected independently when the service is deployed
  - These are also orthogonal with respect to the MEP (Message Exchange Pattern) used by the operation

WSDL example (1)

```
<?xml version="1.0"?>
    <definitions name="StockQuote" targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
  xmlns:xsd1="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="GetTradePriceInput">
    <part name="tickerSymbol" element="xsd:string"/>
    <part name="time" element="xsd:timeInstant"/>
  </message>

  <message name="GetTradePriceOutput">
    <part name="result" type="xsd:float"/>
  </message>

  <portType name="StockQuotePortType">
    <operation name="GetTradePrice">
      <input message="tns:GetTradePriceInput"/>
      <output message="tns:GetTradePriceOutput"/>
    </operation>
  </portType>
```

From Web Services Description Language (WSDL) 1.1 W3C Note 15 March 2001

WSDL example (2)

```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetTradePrice">
      <soap:operation soapAction="http://example.com/GetTradePrice"/>
      <input>
        <soap:body use="encoded" namespace="http://example.com/stockquote"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded" namespace="http://example.com/stockquote"
         encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>>
  </binding>

  <service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
      <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>
</definitions>
```

From Web Services Description Language (WSDL) 1.1 W3C Note 15 March 2001

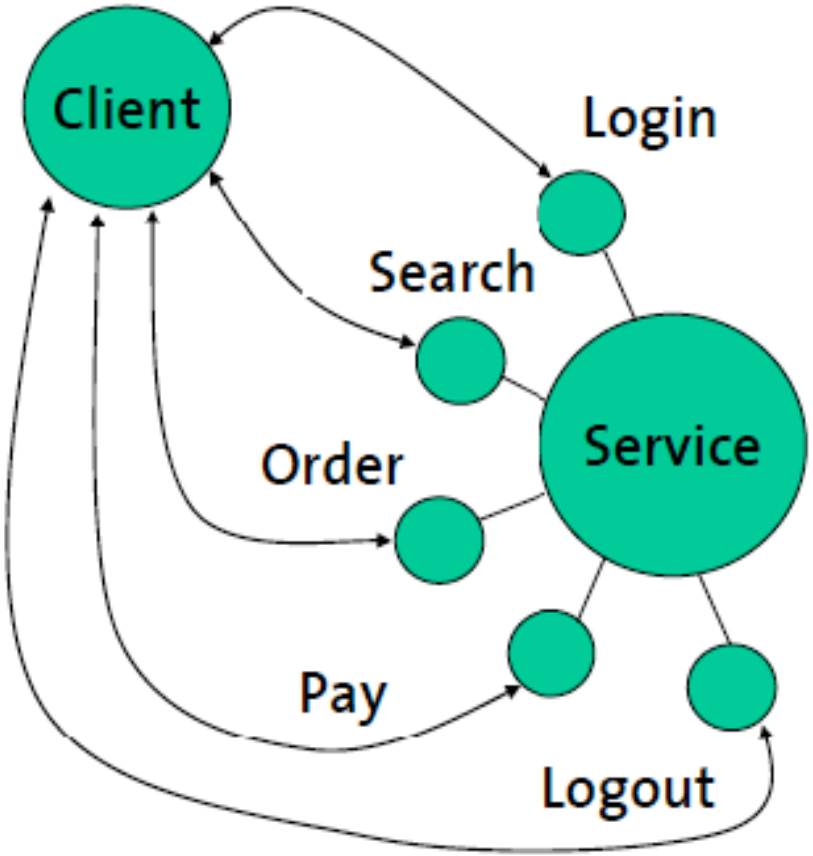
Conversations (1)

- As a first approximation, a conversation models the sequences of operations that a client may invoke as part of the interaction with a Web service
- In general, a conversation defines a complex interaction between multiple Web services involving the exchange of several messages and the invocation of different operations in a well defined order

Conversations (2)

- In this context, a coordination protocol specifies the set of correct conversations between the various services
  - the service interface description (WSDL) only lists the available operations but does not specify what is the correct order of invoking them

Conversations (3)



Conversations (4)

- WSDL defines the interface of a Web service in terms of what are the messages that are exchanged (received and produced by the service)
  - A WSDL document also structures the messages into pairs (that correspond to the operations provided by a service)
  - However, WSDL does not contain any further information specifying what is the correct order of invocation of the various operations. If an operation should not (yet) be invoked, a fault message is returned

Conversations (5)

- From the client's point of view, this makes it difficult to automatically ensure the correctness of the interaction
  - On the service side, an interaction across multiple operations may require to maintain session information. (statefulinteraction). This information is also used to enforce the correctness of the interaction. Whatever mechanism is employed, these constraints do not surface in the WSDL interface description
- The goal is to make the development as automatic as possible!

Modeling conversations (1)

• There are many different ways of modeling conversations. Extending the basic interface description of a service, the most important requirement is to describe what are all possible valid conversations with it. Thus, our description must define all acceptable sequences of operation invocations

- On the interface level, the goal is to make clients aware of what is the coordination protocol supported by the service so that they interact correctly with it
- Internally, the purpose of the conversation model is to support and facilitate the development of the implementation of the service relying on the coordination infrastructure to conduct the conversation

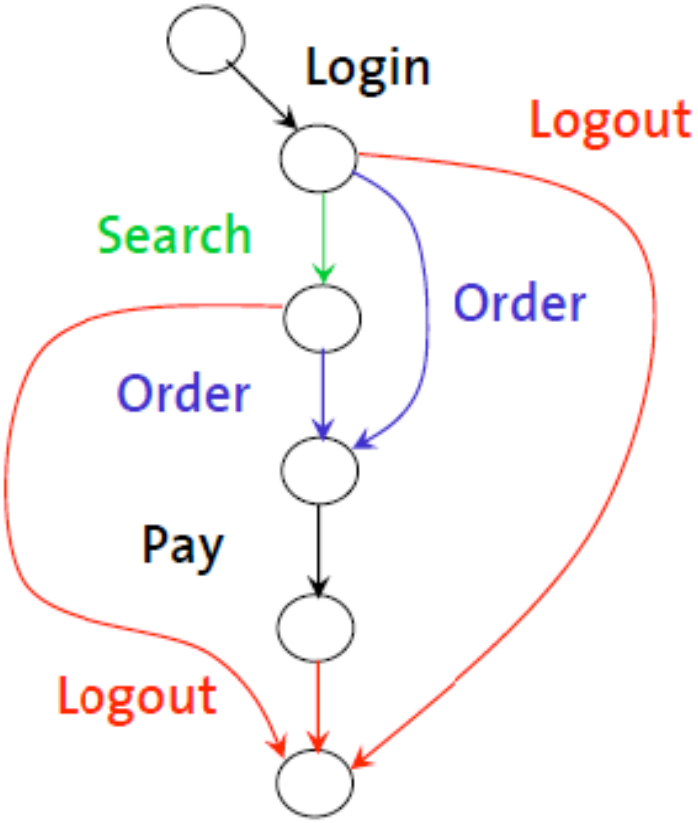
Modeling conversations (2)

- The peers to be coordinated are usually labeled with their role in the conversation
  - The coordination protocol specifies:
    - what are the possible messages (or operations) that make up the conversation
    - for each message, which role is the originator and who is the intended recipient
    - what are the ordering constraints in the sequence of message exchanges
    - under which conditions the conversation is considered to be completed
  - This information complements the WSDL port type (or interface) to be provided by each role, which only lists the accepted messages

Finite state machines (1)

- A simple way of describing a coordination protocol is to use a finite state machine
  - states model the stages followed by the conversation
  - transitions correspond to operations invoked by the client on the service (or in general, to messages which are received or sent by the service)
  - the set of valid operations (which can legally be invoked by the client) changes with the state of the conversation. Each time a message is exchanged, the state of the conversation must be updated
  - the valid operations are the ones associated with the transitions outgoing from the current state

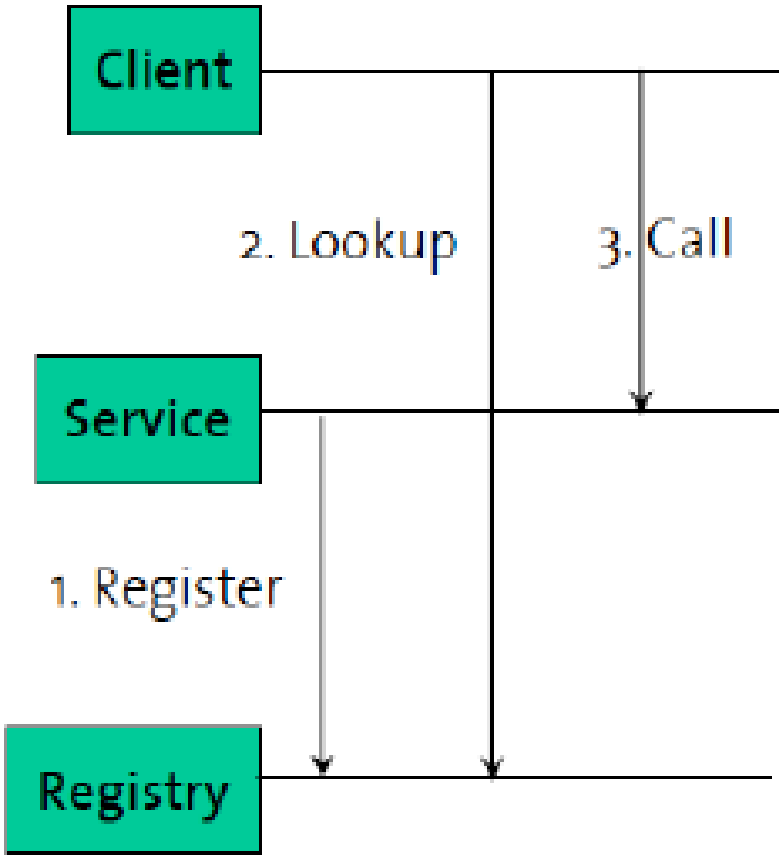
Finite state machines (2)



UML Sequence Diagrams (1)

- When multiple peers are involved, state machines can still be used with the addition of roles, identifying the sender and receiver of each message
- Alternative representations (with the same semantics) are also possible
  - UML Sequence Diagrams allow to visualize the evolution of the conversation over time, by showing each message as it is exchanged between two roles
  - Sequence diagrams lack branching support, where alternative diagrams must be used to show alternative paths in the conversation. Also, when many roles are involved, the readability of the diagram may suffer

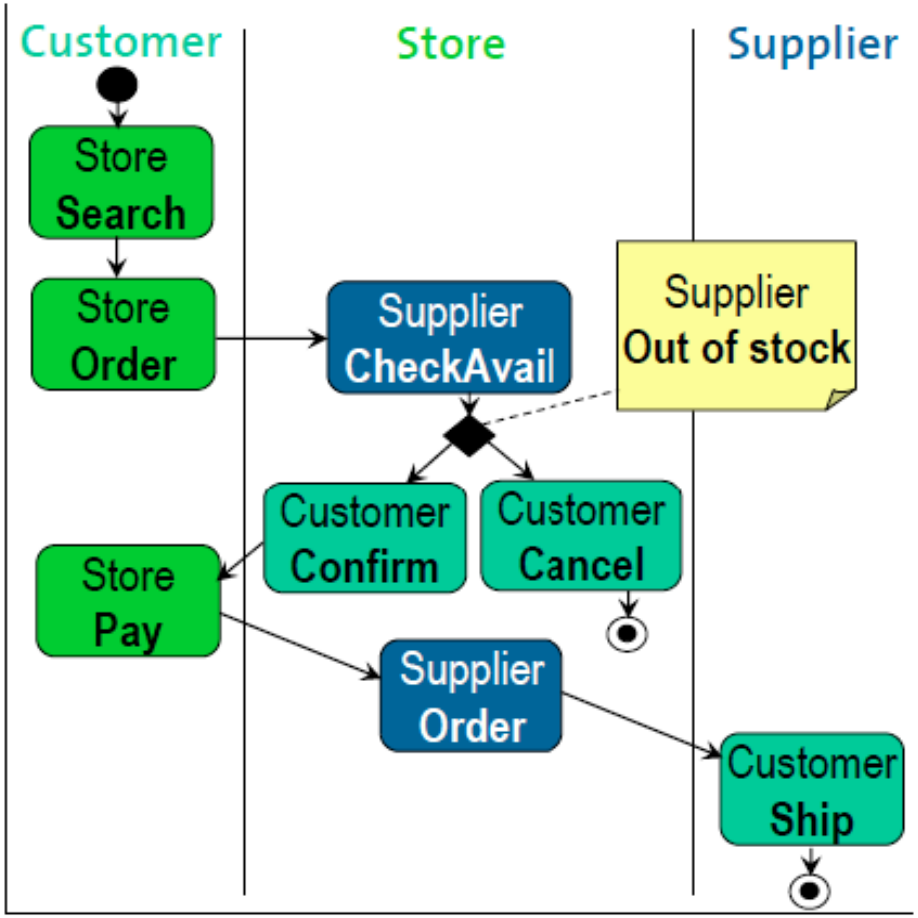
UML Sequence Diagrams (2)



UML activity diagrams (1)

- Activity Diagrams also support multiple roles through swimlanes. Unlike sequence diagrams, they also support branches (and parallelism) in the flow of the conversation
  - Activities model the synchronous or asynchronous invocation of an operation. The activity is positioned in the swimlane of the client role and is labeled with the target role
  - Final states are used to identify completed conversations

UML activity diagrams (2)



Syntax and semantics (1)

- One of the advantages of using self-describing XML for encoding SOAP messages is that it becomes really easy to develop the corresponding parsers (for reading messages) and emitters (for writing messages)
- There are however some disadvantages, not only related to the performance overhead (XML parsing and validation is expensive) but also to the limitations of XML as a data exchange format (SOAP Attachments for exchanging binary data)
- Another problem is that parseability does not guarantee interoperability

Syntax and semantics (2)

- The fact that all parties involved can parse SOAP messages, only solves the interoperability problem at the syntax level. Although progress has already been made by standardizing the syntax, there is still a lot to be done to agree on the semantics of the messages
  - At the SOAP-level, it may be necessary to apply transformations to the messages that are exchanged (Data mapping tools for EAI have not disappeared, they have just become XML/XSLT based)
  - At the WSDL-level, it should be possible to describe the semantics in addition to the syntax of the service interfaces

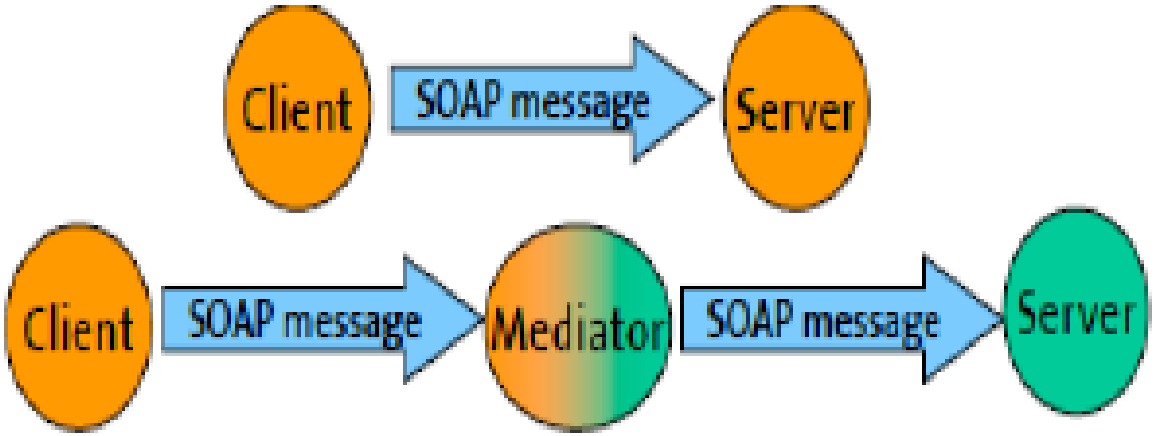
Interface syntax

- WSDL defines a service interface (or port type) as a set of operations, grouping together pairs of messages, which are defined in terms of parts (with name and data type, defined in an XML schema)
- From a WSDL description it is possible to automatically infer (and validate) the structure of the corresponding SOAP messages

Interface syntax: example

```
<message name="getRateRequest"><part name="country1" type="xsd:string"
   /> <part name="country2" type="xsd:string" />
</message><soap:Body
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><n:ge
   tRate> <country1 xsi:type="xsd:string">USD</country1>
<country2 xsi:type="xsd:string">CHF</country2> </n:getRate>
   </soap:Body>
<soap:Body
   soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<n:getRate> <country1 xsi:type="xsd:string">usa</country1>
<country2 xsi:type="xsd:string">italy</country2> </n:getRate>
   </soap:Body>
```

Data transformation (1)



Data transformations (2)

- In this example, both client and server use WSDL to describe their interface and SOAP to exchange a message. Even if we assume that the two parties are somehow compatible, this standardization does not guarantee interoperability, unless both services use the same XML Schema and (abstracted from the interface description), they agree on the semantics of the message

Data transformation (3)

- If it is possible to address this mismatch, the message cannot be sent directly, but should be transformed between the two schemas while preserving its semantics
  - This transformation can occur at the client-side (the client knows how to adapt to a given server), at the server-side (the server supports different data models) or –in a true integration scenario –in the middle (using a mediator service, or an ESB)

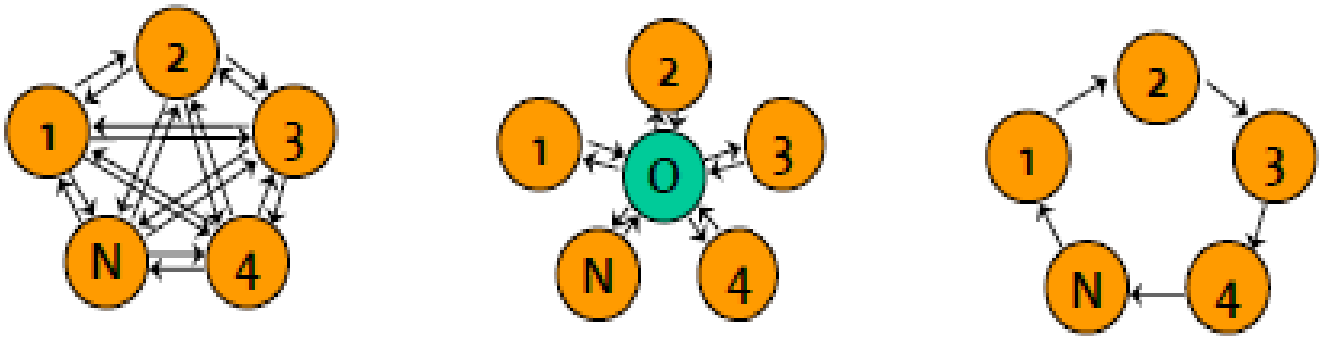
Scaling transformations (1)

- In general, given N parties to be integrated, up to = N (N-1) transformations have to be defined (and maintained)
  - This assumes that it should be possible to directly transform between the schemas used by each of the parties
  - If we allow to compose multiple transformations and introduce an intermediate representation, the number of transformations is reduced to 2N

Scaling transformations (2)

• If the various schemas are fully overlapping (i.e., the transformations do not loose information when they are applied to the messages), then it is also possible to avoid a single intermediate representation and apply a sequence of transformations in order to get the desired format. In this case, we only need N transformations

Scaling transformations (3)



Interface semantics (1)

- Each syntactical element of a service interface (message, data structure or operation) has a precise semantic meaning associated with it
- This meaning should be taken into account by clients invoking the service, so that they can understand what functionality is offered by the service

Interface semantics (2)

- Semantics can be modeled:
  - using standard documents
  - using constraints (e.g., in case of domains having enumerable elements)
  - using ontologies (which formally define a vocabulary of terms and relationships)
  - using contracts (pre-conditions and post-conditions)

Interface semantics (3)

- In an integration scenario, the middleware infrastructure should preserve the semantics of the applications to be integrated as well as provide support for mediation (the transformation of messages between different representation by mapping concepts that are shared between all applications)
  - If services are described with WSDL, there is very little semantics associated with them.
- Thus, there are many extensions to WSDL that can be used to model semantics, e.g., using the RDF (Resource Description Framework) and OWL (Web Ontology Language)