HELSINKI UNIVERSITY OF TECHNOLOGY
Media Technology

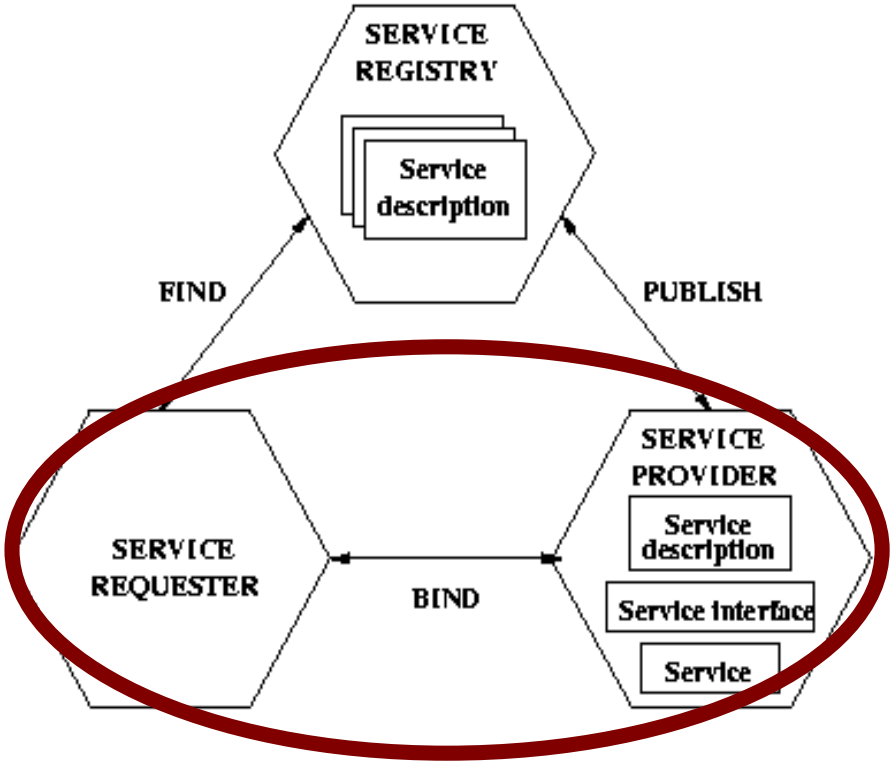
AS-75-3600
WSDL – Web Service Definition Language

Tuukka Ruotsalo
Semantic Computing Research Group (SeCo)
Helsinki University of Technology (TKK),
Laboratory of Media Technology
and
University of Helsinki, Department of Computer Science
Slides partially based on (Alonso & Pautasso, 2004)

http://www.seco.hut.fi/

UNIVERSITY OF HELSINKI

SeCo
semantic computing


SERVICE
REGISTRY
Service
description
FIND
PUBLISH
SERVICE
REQUESTER
BIND
SERVICE
PROVIDER
Service
description
Service interface
Service

Interoperability of distributed systems

- Information system consists of two main components
  - Data model
    - » Conceptual model of the universe of discourse
    - » i.e. data that the system handles
  - Process model
    - » Process model of the possible event occurences in the system
    - » i.e. processes that may affect the data-model
- Distributed systems
  - Map data and process models
- Web services
  - How to send data and process information over the web
  - What kind of communication models this requires

In practice

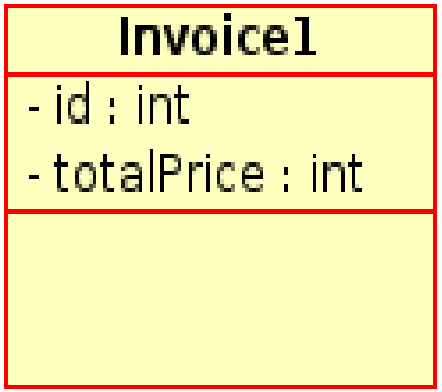
- Description of the data and data structures
  - Types
- Description of the interfaces receiving and sending the messages
  - Events / operations
- Description of the bindings
  - Connections to the actual protocols

Example


```
<xs:complexType name="Invoice1Type">

   <xs:element name="id" type="xs:nonNegativeInteger"/>
   <xs:element name="totalPrice" type="xs:nonNegativeInteger"/>

</xs:complexType>
```

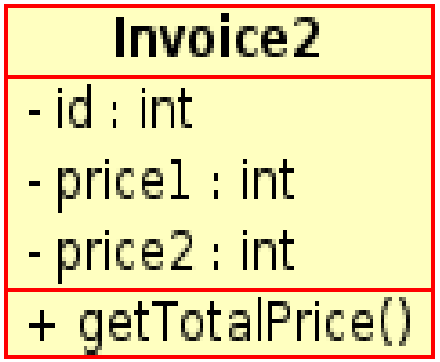


Example


```
<xs:complexType name="Invoice2Type">

   <xs:element name="id" type="xs:nonNegativeInteger"/>
   <xs:element name="price1" type="xs:nonNegativeInteger"/>
   <xs:element name="price2" type="xs:nonNegativeInteger"/>

</xs:complexType>
```



- Methods (Events/Operations) can not be serialized to XML Schema

WSDL

- Abstract description
  - the type system used to describe the messages (usually XML Schema)
  - the individual operations exchanging types
  - features of the operations
  - an interface that groups the operations that constitute an abstract service
- Concrete description
  - binding the interface to a transport protocol
  - the endpoint or network address of the binding
  - a service as a collection of all bindings of the same interface

WSDL 2.0 definition

- Types– a container for data type definitions
  - using some type system (such as XML Schema)
- Interfaces
  - possesses operations
- Operation
  - name
  - message exchange pattern
  - message references: the messages involved (input, output). To XML Schema types
  - fault references: the faults involved (infault, outfault)
  - style (optional): RPC, set-attribute or get-attribute
  - features and properties
- Style
  - RPC = implies interactions mirroring the behavior of RPC
  - set- and get- attribute = implies interactions of the type commonly found in object oriented languages
- Features and properties
  - are used to specified characteristics of the message exchange implied by an operation. Examples include reliability, security, routing, etc

Types: XML Schema definition for structures and data-types

- The types in WSDL are used to specify the contents of the messages (normal messages and fault messages)
- The type system is typically based on XML Schema (structures and data types)
- An extensibility element can be used to define a schema definition language other than XML Schema

```
<types>
 <xs:schema
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     targetNamespace="http://greath.example.com/2004/schemas/resSvc"
     xmlns="http://greath.example.com/2004/schemas/resSvc">

   <xs:element name="checkAvailability" type="tCheckAvailability"/>
   <xs:complexType name="tCheckAvailability">
     <xs:sequence>
       <xs:element  name="checkInDate" type="xs:date"/>
       <xs:element  name="checkOutDate" type="xs:date"/>
       <xs:element  name="roomType" type="xs:string"/>
     </xs:sequence>
   </xs:complexType>

   <xs:element name="checkAvailabilityResponse" type="xs:double"/>

   <xs:element name="invalidDataError" type="xs:string"/>

 </xs:schema>
</types>
```

Operations

- An operation is a set of messages and faults. The sequencing and number of messages in the operation is determined by the message exchange pattern

- An operation has:
  - name
  - message exchange pattern
  - message references: the messages involved (input, output)
  - fault references
  - style (optional): RPC, set-attribute or get-attribute
  - features and properties

```
<interface  name = "reservationInterface" >

  <fault name = "invalidDataFault"
         element = "ghns:invalidDataError"/>

  <operation name="opCheckAvailability"
         pattern="http://www.w3.org/2005/08/wsdl/in-out"
         style="http://www.w3.org/2005/08/wsdl/style/iri"
         wsdlx:safe = "true">
     <input messageLabel="In"
           element="ghns:checkAvailability" />
     <output messageLabel="Out"
           element="ghns:checkAvailabilityResponse" />
     <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>

</interface>
```

Operations

- Style
  - RPC = implies interactions mirroring the behaviour of RPC
  - set- and get- attribute = implies interactions of the type commonly found in object oriented languages
- Features and properties:
  - are used to specified characteristics of the message exchange implied by an operation. Examples include reliability, security, routing, etc

Fault messages

- Called the “fault reference component”, it contains:
  - a name
  - message reference: the message to which the fault refers to
  - direction: whether the fault is inbound or outbound
  - message: the actual contents

```
<interface  name = "reservationInterface" >

  <fault name = "invalidDataFault"
         element = "ghns:invalidDataError"/>

  <operation name="opCheckAvailability"
         pattern="http://www.w3.org/2005/08/wsdl/in-out"
         style="http://www.w3.org/2005/08/wsdl/style/iri"
         wsdlx:safe = "true">
      <input messageLabel="In"
           element="ghns:checkAvailability" />
      <output messageLabel="Out"
           element="ghns:checkAvailabilityResponse" />
      <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>

</interface>
```

Binding

- Bindings anchor the interface to a implementation
  - Connection to the communication protocol
- A binding defines message formats and protocol details for the operations and messages of a given interface
- An interface corresponds to the abstract description of the Web service, it does not contain any information about where the service resides or what protocols are used to invoke the Web service
- Endpoints define the actual address of the service

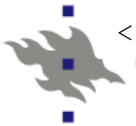
```
<binding name="reservationSOAPBinding"
         interface="tns:reservationInterface"
         type="http://www.w3.org/2005/08/wsdl/soap"
         wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">

   <fault ref="tns:invalidDataFault"
     wsoap:code="soap:Sender"/>
   <operation ref="tns:opCheckAvailability"
     wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
 </binding>

 <service name="reservationService"
      interface="tns:reservationInterface">

    <endpoint name="reservationEndpoint"
              binding="tns:reservationSOAPBinding"
              address ="http://greath.example.com/2004/reservation"/>

 </service>
```



WSDL 2.0 Message exchange patterns
(Alonso & Pautasso, 2004)

- IN-ONLY
  - a single incoming message (A) with no faults
- ROBUST IN-ONLY
  - an inbound message (A) that might trigger a fault message
- IN-OUT
  - An incoming message (A) received from node N
  - An outgoing message (B) sent to node N
  - Faults, if any, replace message B
- IN-MULTI-OUT
  - Like IN-OUT but with zero or more outbound messages and “fault replaces message” behavior
- OUT-ONLY
  - An outbound message (A) that expects no faults
- ROBUST OUT-ONLY
  - An outbound message (A) that might trigger a fault
- OUT-IN
  - An outbound message (A) to node N
  - An inbound message (B) from node N
  - Faults, if any, replace message B
- ASYNCHRONOUS OUT-IN
  - Like OUT-IN but with trigger behaviour for messages
- OUT-MULTI-IN
  - reverse of IN-MULTI-OUT

WSDL 2.0 Example (www.w3c.org)


```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/2005/08/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/2005/08/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdlx= "http://www.w3.org/2005/08/wsdl-extensions">
  <documentation>
    This document describes the GreatH Web service.  Additional
    application-level requirements for use of this service --
    beyond what WSDL 2.0 is able to describe -- are available
    at http://greath.example.com/2004/reservation-documentation.html
  </documentation>
  <types>
    <xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://greath.example.com/2004/schemas/resSvc"
        xmlns="http://greath.example.com/2004/schemas/resSvc">
      <xs:element name="checkAvailability" type="tCheckAvailability"/>
      <xs:complexType name="tCheckAvailability">
        <xs:sequence>
          <xs:element  name="checkInDate" type="xs:date"/>
          <xs:element  name="checkOutDate" type="xs:date"/>
          <xs:element  name="roomType" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>

      <xs:element name="checkAvailabilityResponse" type="xs:double"/>

      <xs:element name="invalidDataError" type="xs:string"/>
    </xs:schema>
  </types>
```

WSDL 2.0 Example (www.w3c.org)


```
  <interface  name = "reservationInterface" >
    <fault name = "invalidDataFault"
           element = "ghns:invalidDataError"/>

    <operation name="opCheckAvailability"
           pattern="http://www.w3.org/2005/08/wsdl/in-out"
           style="http://www.w3.org/2005/08/wsdl/style/iri"
           wsdlx:safe = "true">
       <input messageLabel="In"
              element="ghns:checkAvailability" />
       <output messageLabel="Out"
              element="ghns:checkAvailabilityResponse" />
       <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>
  </interface>
  <binding name="reservationSOAPBinding"
         interface="tns:reservationInterface"
         type="http://www.w3.org/2005/08/wsdl/soap"
         wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>
    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
  </binding>
  <service name="reservationService"
       interface="tns:reservationInterface">
     <endpoint name="reservationEndpoint"
              binding="tns:reservationSOAPBinding"
              address ="http://greath.example.com/2004/reservation"/>

  </service>
</description>
```

Short introduction to WS-Addressing & WS-Security

- Addressing in http is easy = URI based
- In WS world bindings may be done through several protocols
  - Need for a way to discover the service endpoint protocol independently
  - WS-Addressing
    - Web Services Addressing 1.0 - Core (WS-Addressing) defines two constructs, message addressing properties and endpoint references, that normalize the information typically provided by transport protocols and messaging systems in a way that is independent of any particular transport or messaging system.
- Identification of users and reliable messaging required
  - Need for more general way to authenticate on message level (implementation independent) and provide support for message level security
  - WS-Security
    - The goal of WS-Security is to enable applications to construct secure SOAP message exchanges
    - WS-Security is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of security models and encryption technologies. Implementing WS-Security does not mean that an application cannot be attacked or that the security cannot be compromised.

WSDL – still need for Business Protocol

HELSINKI UNIVERSITY OF TECHNOLOGY
Media Technology

- WSDL is in its current version an extension of the IDL model to support interaction through the Internet:
  - XML as syntax and type system
  - possibility of grouping operations into a service
  - different options for accessing the service (addresses and protocols)
- This is its great advantage …
  - it is straightforward to adapt existing middleware platforms to use or support WSDL
  - automatic translation from existing IDLs to WSDL is trivial
- … but also the disadvantage
  - electronic commerce and B2B interactions are not single service calls
  - WSDL does not reflect the structure of the procedures to follow to correctly interact with a service (conversations)
    - business protocol = set of valid conversations
- Without a business protocol, most of the development work is still manual
- Business Protocol can be defined with orchestration and common understanding of contents changed between services (semantics)

UNIVERSITY OF HELSINKI

SeCo
semantic computing