Web Services Description Language

(Version 2.0)

References

- http://www.w3.org/TR/wsdl20/
- Numerous presentations found on the web

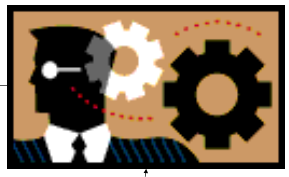
The need and the opportunity

- Standardization of communication protocols and message formats in the web community
- Describing the communications in a structured way becomes important and possible

➔WSDL:

- ❑XML grammar for describing network services as collections of communication endpoints capable of exchanging messages
- ❑ Provide a model and an XML format for describing Web services.

Intuition



What can it do? Features
How to use it? Operations
How to meet the requirements?

Result

Web Service

Abstract operations
Bindings
Faults

How can the service be executed?
What is the input/output?
What to do when unexpected event happen?

What is WSDL?

- Used for describing Web Services
- Models the interactions between nodes
  - Interaction: exchange of messages
- Focus on the "on-wire" representation
  - NOT on the programming language binding
- Independent of specific endpoint technology
  - Can be Java, C#, Perl, SQL, …
- Extensible
- Protocol independent
  - Built-in support for SOAP and HTTP/MIME
- Advantage: reusable (interfaces, binding, messages, etc.)

WSDL 2.0

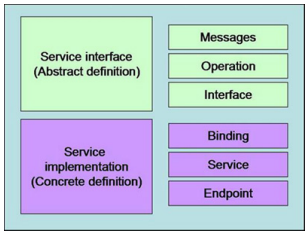
- 3 specifications
  - Part 1: Core
    - Abstract interfaces, independent of protocol and encoding
  - Part 2: Message Exchange Patterns
    - Predefined types of interactions
  - Part 3: Bindings
    - SOAP and HTTP/MIME binding

WSDL document

- Defines **services** as collection of network endpoints
- Separates the abstract definition of endpoints and messages from the concrete network deployment or data format bindings (“what” or the functionality of the services vs. “where” and “how” or the location of the service and how the service is executed)

WSDL Document

- 2 parts
  - Abstract part
    - Interfaces (from Part 1) and MEPs (from Part 2)
  - Concrete part
    - Binding, with concrete protocol and encoding (from Part 3)



Conceptual Model of WSDL

(From: http://www.xml.com/lpt/a/ws/2004/05/19/wsdl2.html)

In short

- A WSDL document is an XML document describing a Web service.
- Specification of what the service can do and where the service can be located.

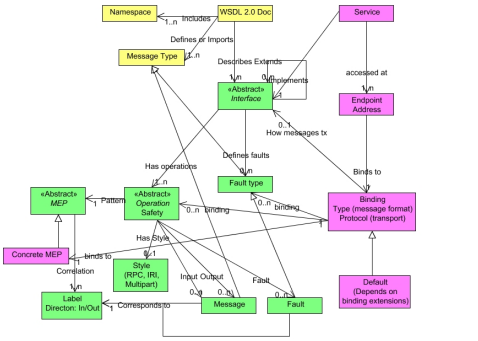
Component model

- WSDL elements are treated as components
- Components are typed collections of properties that correspond to different aspects of Web services

Description Component

- Container for two categories:
  - WSDL 2.0 components
    - Interfaces
    - Bindings
    - Services
  - Type system components: constraints on message’s contents
    - Information set

Full Model (approx.)



From a presentation of Paul Brebner, CSIRO ICT Centre, Canberra

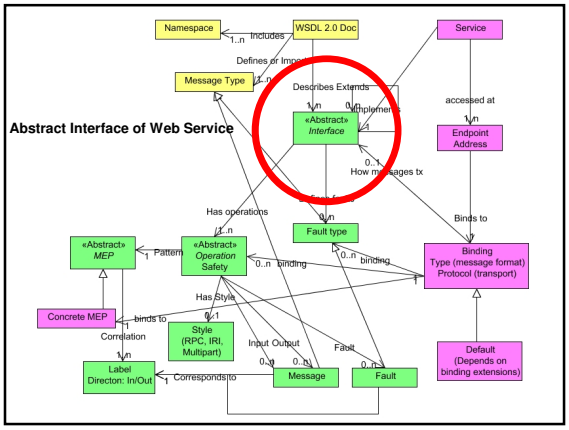
XML Representation of Description Component

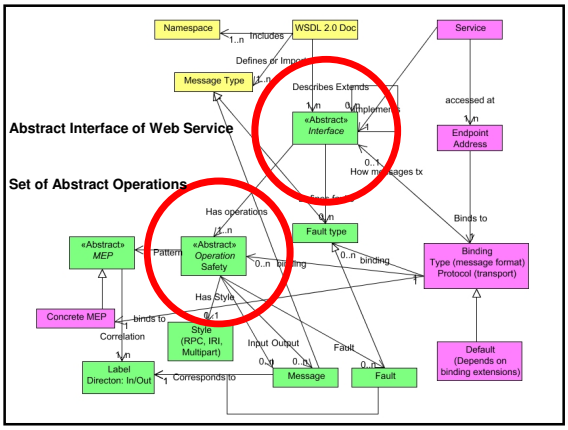
<description REQUIRED
targetNamespace="xs:anyURI" >
<documentation />*
[<import /> | <include />]* (in picture: yellow)
<types />?
[<interface /> | <binding /> | <service />]*
</description>
(*: one or more, []: optional)

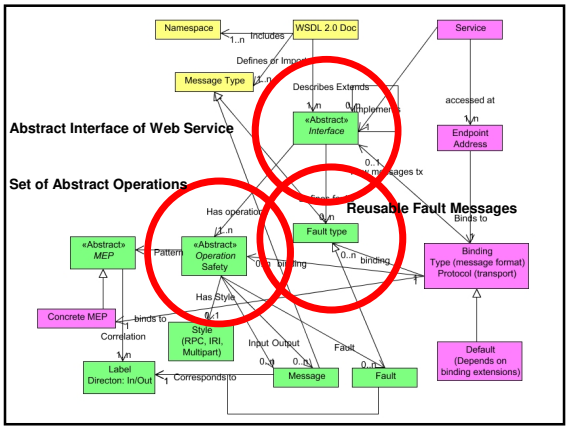
Allowing: description to be divided into multiple documents (use of import/include)

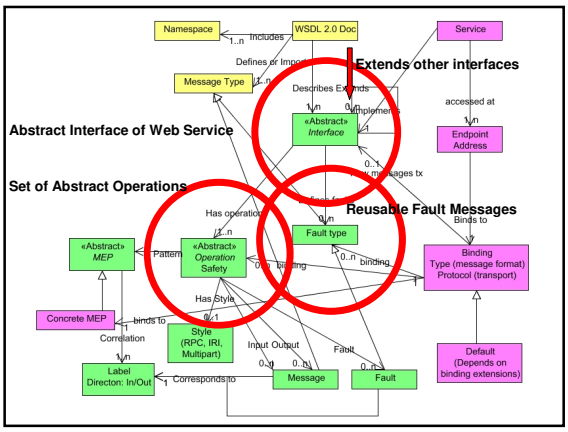
Interfaces

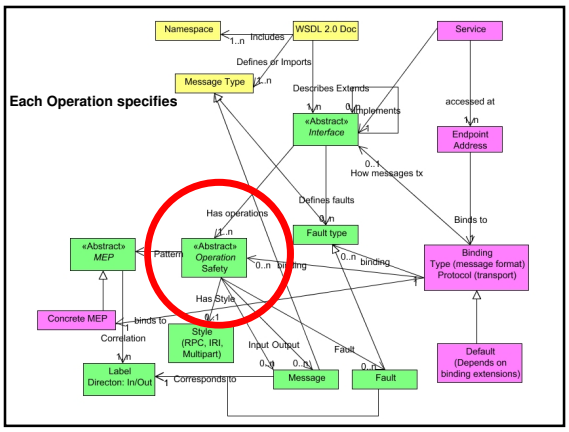
- A WSDL 2.0 interface
  - Defines the abstract interface of a Web service as a set of abstract *operations* and reusable fault messages
  - Can extend other interfaces
- Each operation
  - represents a simple interaction between the client and the service.
  - specifies the types of messages that the service can send or receive as part of that operation (including faults)
  - specifies a message exchange *pattern* that indicates the sequence in which the associated messages are to be transmitted between the parties.
  - Has a style and safety

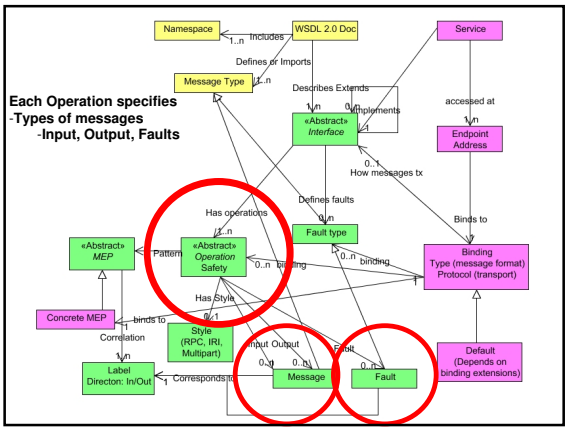


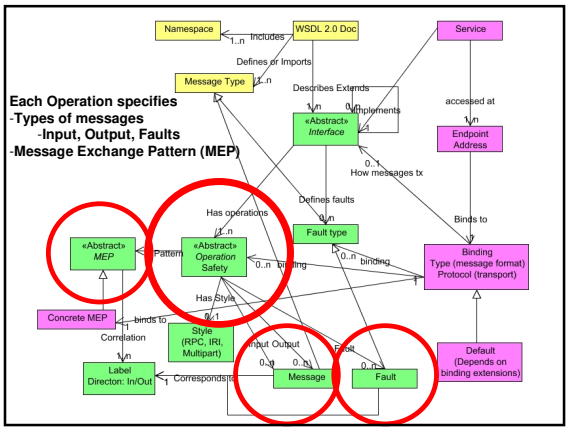


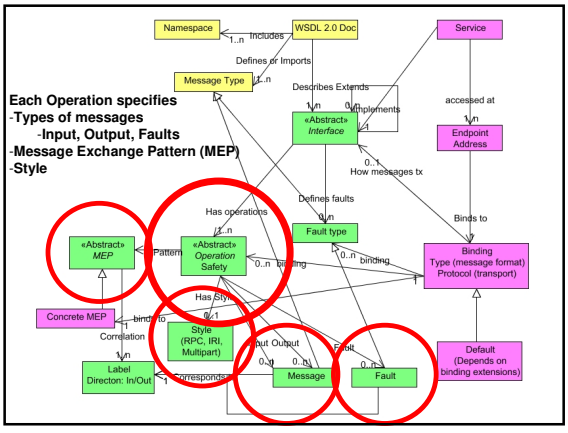


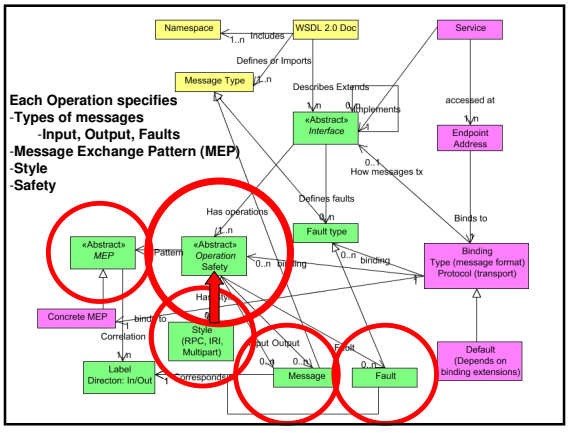

Extends other interfaces
Abstract Interface of Web Service
Set of Abstract Operations
Reusable Fault Messages
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
Implements
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Has Style
Concrete MEP
binds to
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message


Each Operation specifies
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
Implements
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Defines faults
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Has Style
Concrete MEP
binds to
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message


Each Operation specifies
-Types of messages
-Input, Output, Faults
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
Implements
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Defines faults
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Has Style
Concrete MEP
binds to
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message

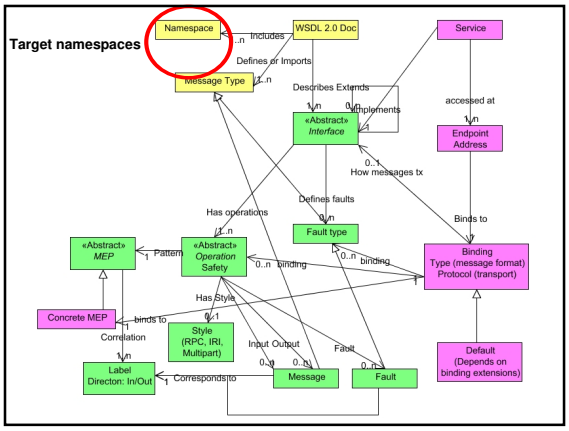

Each Operation specifies
-Types of messages
-Input, Output, Faults
-Message Exchange Pattern (MEP)
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
Implements
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Defines faults
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Has Style
Concrete MEP
binds to
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message

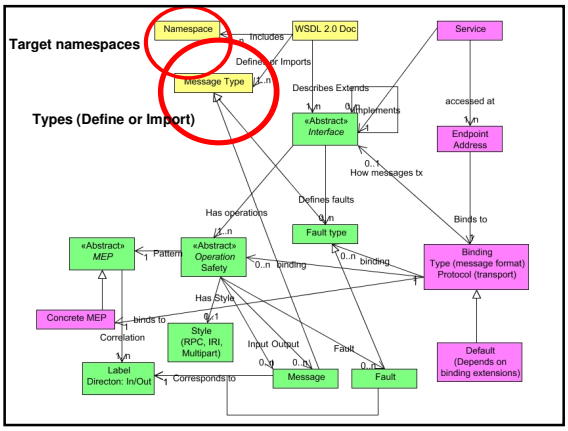

Each Operation specifies
-Types of messages
-Input, Output, Faults
-Message Exchange Pattern (MEP)
-Style
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
Implements
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Defines faults
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Has Style
Concrete MEP
binds to
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message


Each Operation specifies
-Types of messages
-Input, Output, Faults
-Message Exchange Pattern (MEP)
-Style
-Safety
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
Implements
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Defines faults
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Has Style
Concrete MEP
binds to
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message

Message Types

- Top-level WSDL document **description** describes
  - The target namespaces
  - Documentation (comments)
  - Types
    - Define the type of every message
      - Including faults
    - Multiple solutions
      - Minimally, XML Schema supported
      - Define message types
        - » In WSDL document, or
        - » Import them





Interface

- describes sequences of messages that a service sends and/or receives

<**interface** name="*xs:NCName*"
extends="*list of xs:QName*"?
styleDefault="*list of xs:anyURI*"? >
<documentation />*
[<fault /> | <operation /> | <feature /> | <property />]*
</**interface**>

NCName & targetNamespace of the WSDL document forms the QName of the interface. MUST be unique!

Interface

- Only the name is required
- Inheritance: derives from other interfaces (extends attribute) – careful design is needed to avoid conflicting inheritances

Interface Fault

- describes a fault that MAY occur during invocation of an operation of the interface
- fault: an event that occurs during the execution of a message exchange that disrupts the normal flow of messages

<interface>
<**fault** name="*xs:NCName*"
element="*xs:QName*"? >
<documentation />*
[<feature /> | <property />]*
</**fault**>
</interface>
(REQUIRED: name (NCName) and parent (as property))

Interface Operation

- Describes an operation that a given interface supports from the point of view of the service
  - Independently of the protocol and encoding
  - Between any number of participants
  - Following a given pattern/message exchange pattern
- An operation: set of messages exchanged between the services and the other parties

Interface Operation

```
<interface>
      <operation name="xs:NCName"
            pattern="xs:anyURI"
            style="list of xs:anyURI"? >
          <documentation />*
          [ <feature /> | <property /> | [ <input /> |
            <output /> | <infault /> | <outfault /> ]+ ]*
      </operation>
</interface>
```

Describing the pattern of messages for ordinary and faulty communication

Interface Feature

- Describes an abstract piece of functionality typically associated with the exchange of messages between communicating parties

```
<feature ref="xs:anyURI"
      required="xs:boolean"? >
  <documentation />*
</feature>
```

Use of the feature is mandatory or not

Example

```
<description targetNamespace="http://example.com/bank"
        xmlns=http://www.w3.org/2006/01/wsdl
        xmlns:ns1="http://example.com/bank">
  <interface name="ns1:Bank">
  <!-- All implementations of this interface must be secure -->
  <feature ref="http://example.com/secure-channel" required="true"/>
  <operation name="withdrawFunds">
        <!-- This operation must have ACID properties -->
        <feature ref="http://example.com/transaction" required="true"/>
        ...
  </operation>
  <operation name="depositFunds">
        <!-- This operation requires notarization -->
        <feature ref="http://example.com/notarization" required="true"/>
        ...
  </operation>
  </interface>
```

Example

```
<binding name="ns1:BankSOAPBinding">
      <!-- This particular binding requires ISO9001 compliance to be
                  verifiable -->
      <feature ref=http://example.com/ISO9001  required="true"/>
      <!-- This binding also requires notarization -->
      <feature ref=http://example.com/notarization  required="true"/>
</binding>
<service name="ns1:BankService" interface="tns:Bank">
      <endpoint binding="ns1:BankSOAPBinding">
            ...
      </endpoint>
</service>
</description>
```

Property

- Property component
  - Specifies value(s) supported or required by the property (for the corresponding feature)
  - Can be shared by several features
  - Not limited to SOAP 1.2

```
<property ref="xs:anyURI" >
  <documentation />*
  [ <value /> | <constraint /> ]?
</property>
```

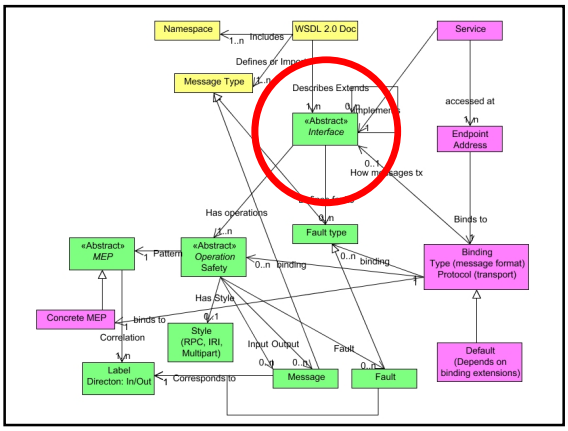
Binding

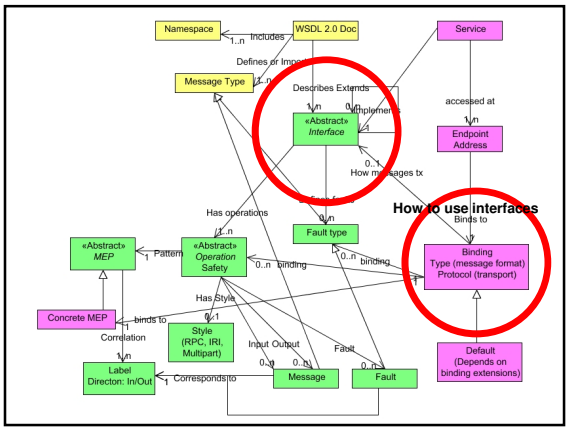
- Defines the implementation details necessary to access the service

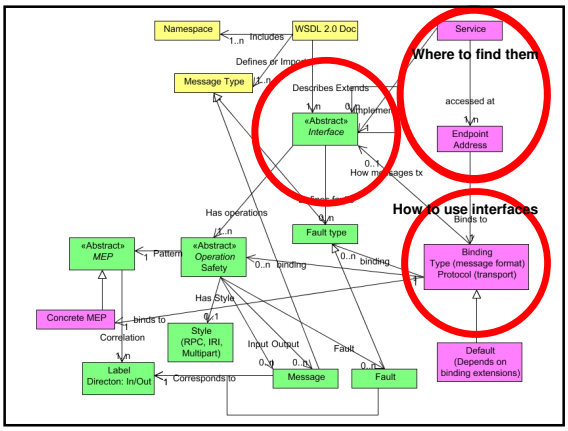
<**binding** name="*xs:NCName*“
interface="*xs:QName*"?
type="*xs:anyURI*" >
<documentation />*
[<fault />|<operation /> | <feature /> | <property />]*
</**binding**>

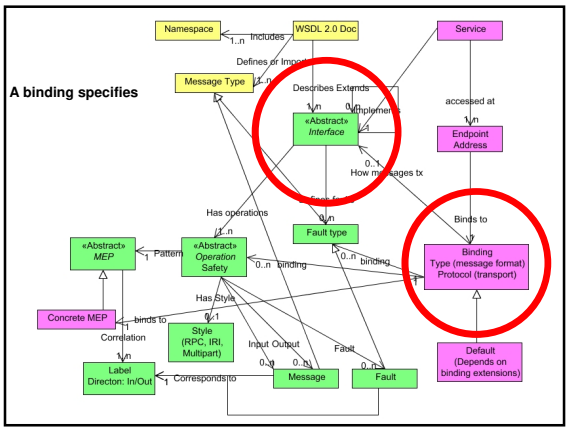
Binding

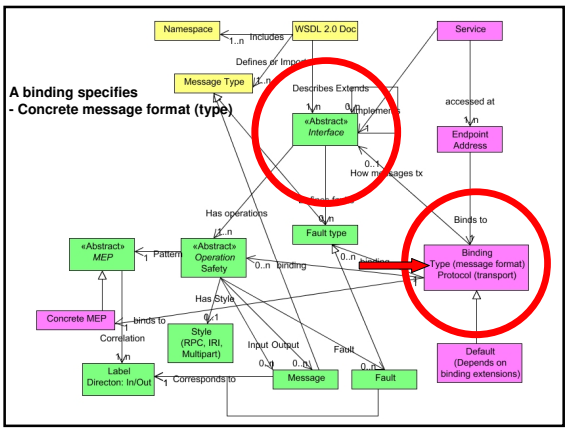
- A binding specifies
  - concrete message format (E.g. SOAP 1.1, 1.2) and
  - transmission protocol details (E.g. HTTP, HTTPS)
  - Concrete MEP
- Every operation and fault in the interface must have a binding.
- Default bindings
  - if the “binding extension” (see spec: WSDL 2.0 extensions) provides suitable defaulting rules, then
    - Bindings only needed at the interface level
    - will implicitly propagate bindings to operations of the interface.

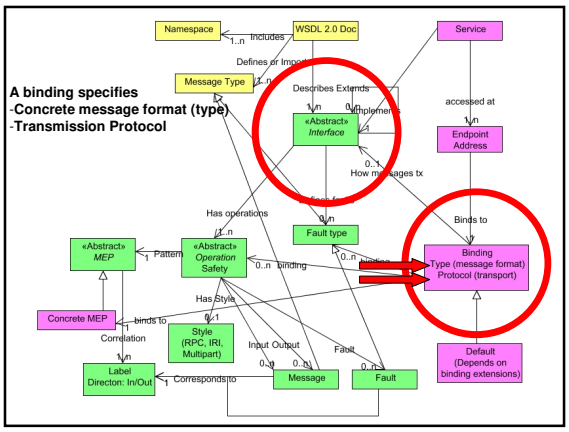


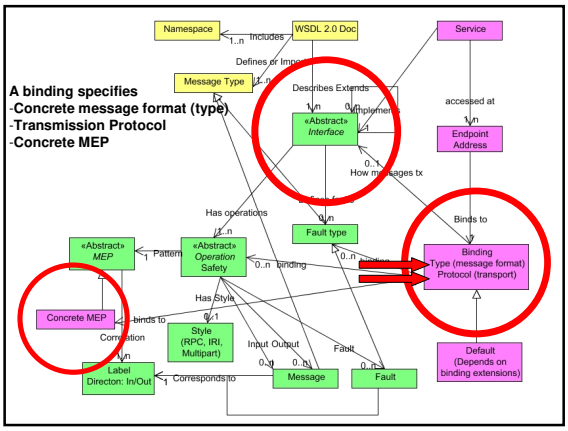


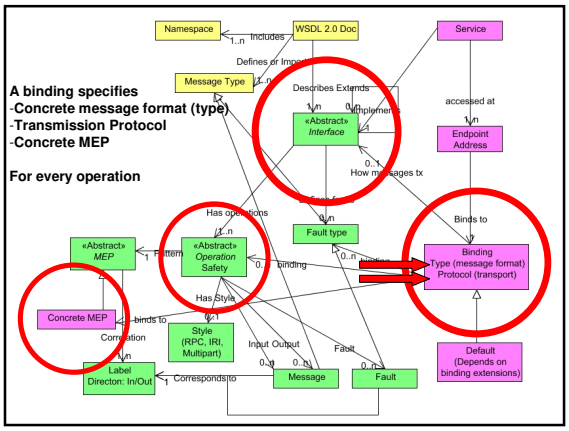


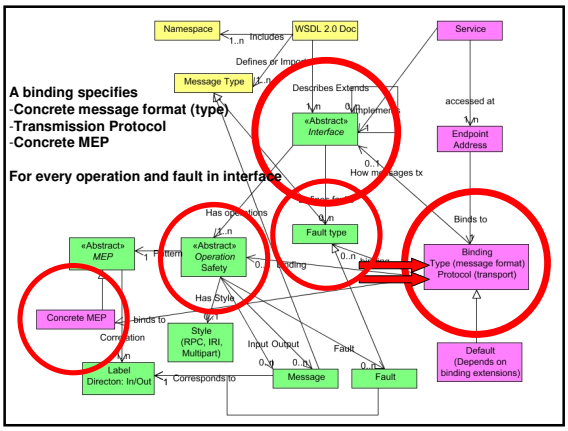


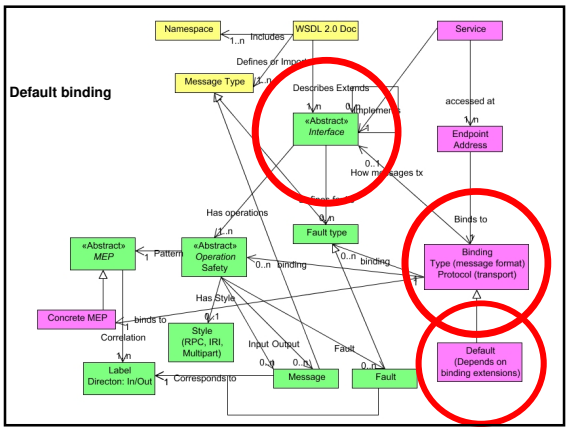

A binding specifies
- Concrete message format (type)
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Concrete MEP
binds to
Has Style
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message
Fault

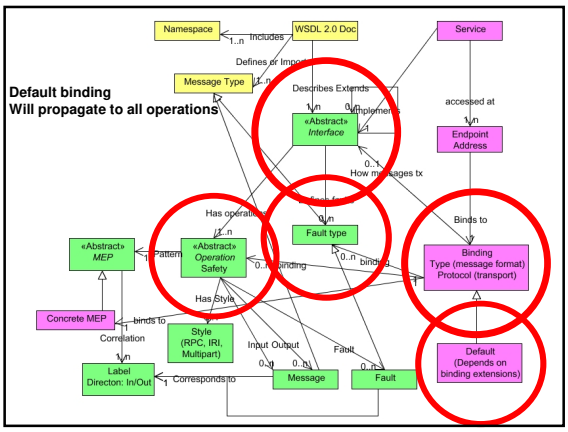

A binding specifies
-Concrete message format (type)
-Transmission Protocol
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Concrete MEP
binds to
Has Style
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message
Fault

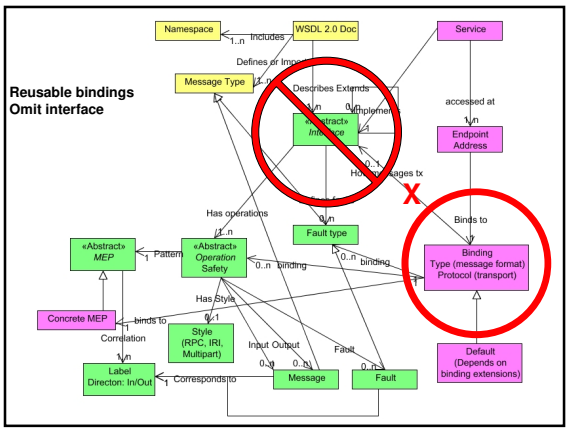

A binding specifies
-Concrete message format (type)
-Transmission Protocol
-Concrete MEP
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Concrete MEP
binds to
Has Style
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message
Fault

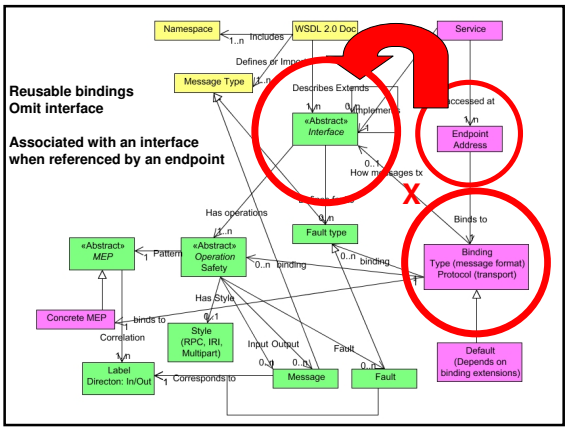

A binding specifies
-Concrete message format (type)
-Transmission Protocol
-Concrete MEP
For every operation
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Concrete MEP
binds to
Has Style
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message
Fault

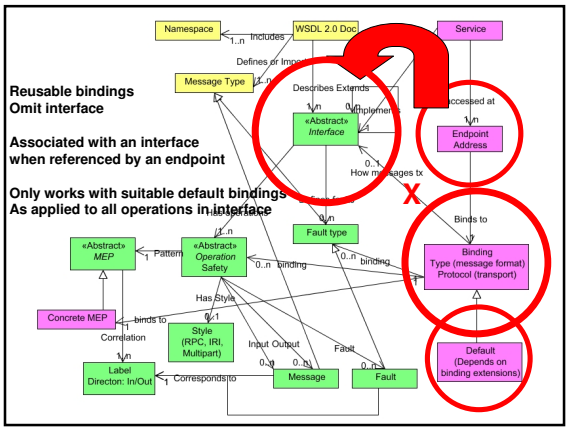

A binding specifies
-Concrete message format (type)
-Transmission Protocol
-Concrete MEP
For every operation and fault in interface
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Concrete MEP
binds to
Has Style
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message
Fault

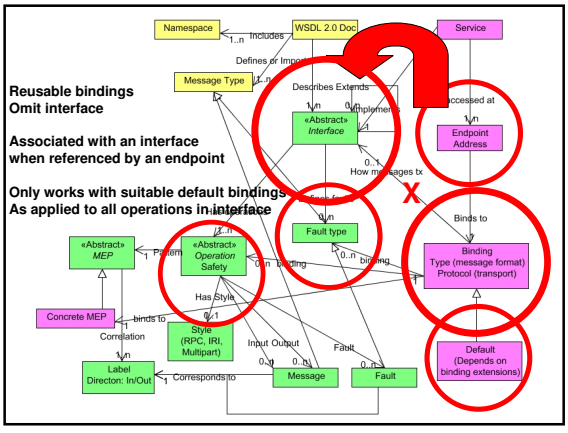

Default binding
Namespace
WSDL 2.0 Doc
Service
Includes
Defines or Imports
Message Type
Describes Extends
«Abstract» Interface
accessed at
Endpoint Address
How messages tx
Has operations
Fault type
Binds to
«Abstract» MEP
Pattern
«Abstract» Operation Safety
binding
Binding Type (message format) Protocol (transport)
Concrete MEP
binds to
Has Style
Correlation
Style (RPC, IRI, Multipart)
Input Output
Fault
Default (Depends on binding extensions)
Label Directon: In/Out
Corresponds to
Message
Fault











Service

- Describes a set of endpoints at which a particular deployed implementation of the service is provided.

<**service** name="*xs:NCName*" ← Name of service
interface="*xs:QName*" > ← Name of interface which the service is an instance of
<documentation />*
<endpoint />+ ← Location of the service
[<feature /> | <property />]*
</**service**>

Endpoint

- Defines the particulars of a specific endpoint at which a given service is available.

<**endpoint** name="*xs:NCName*"
binding="*xs:QName*"
address="*xs:anyURI*"? >
<documentation />*
[<feature /> | <property />]*
</**endpoint**>

Types

- Use to constraint the content of messages and faults

<**types**>
<documentation />*
[<xs:import namespace="*xs:anyURI*"
schemaLocation="*xs:anyURI*"? /> |
<xs:schema
targetNamespace="*xs:anyURI*" /> |
other extension elements]*
</**types**>

Generic WSDL Document Format (1)

<description targetNamespace="*xs:anyURI*" >
<documentation />?
<import namespace="*xs:anyURI*" location="*xs:anyURI*"? >
<documentation />*
</import>*
<include location="*xs:anyURI*" >
<documentation />*
</include>*
<types>
<documentation />*
[<xs:import namespace="*xs:anyURI*"
schemaLocation="*xs:anyURI*"? /> | <xs:schema
targetNamespace="*xs:anyURI*" /> | *other extension elements*]*
</types>

Generic WSDL Document Format (2)

<interface name="*xs:NCName*" extends="*list of xs:QName*"? styleDefault="*list of xs:anyURI*"? > <documentation />*
<fault name="*xs:NCName*" element="*xs:QName*"? > <documentation />*
<feature ... />* <property ... />*
</fault>*

<operation name="*xs:NCName*" pattern="*xs:anyURI*" style="*list of xs:anyURI*"? > <documentation />*
<input messageLabel="*xs:NCName*"? element="*union of xs:QName, xs:token*"? > <documentation />*
<feature ... />* <property ... />*
</input>*
<output messageLabel="*xs:NCName*"? element="*union of xs:QName, xs:token*"? > <documentation />*
<feature ... />* <property ... />*
</output>*
<infault ref="*xs:QName*" messageLabel="*xs:NCName*"? > <documentation />*
<feature ... />* <property ... />*
</infault>*
<outfault ref="*xs:QName*" messageLabel="*xs:NCName*"? > <documentation />*
<feature ... />* <property ... />*
</outfault>*
<feature ... />* <property ... />*
</operation>*

<feature ref="*xs:anyURI*" required="*xs:boolean*"? > <documentation />*
</feature>*
<property ref="*xs:anyURI*" > <documentation />*
<value> *xs:anyType* </value>?
<constraint> *xs:QName* </constraint>?
</property>*
</interface>*

Generic WSDL Document Format (3)

<binding name="*xs:NCName*" interface="*xs:QName*"? type="*xs:anyURI*" > <documentation />*
<fault ref="*xs:QName*" > <documentation />*
<feature ... />* <property ... />*
</fault>*
<operation ref="*xs:QName*" > <documentation />*
<input messageLabel="*xs:NCName*"? > <documentation />*
<feature ... />* <property ... />*
</input>*
<output messageLabel="*xs:NCName*"? > <documentation />*
<feature ... />* <property ... />*
</output>*
<infault ref="*xs:QName*" messageLabel="*xs:NCName*"? > <documentation />*
<feature ... />* <property ... />*
</infault>*
<outfault ref="*xs:QName*" messageLabel="*xs:NCName*"? > <documentation />*
<feature ... />* <property ... />*
</outfault>*
<feature ... />* <property ... />*
</operation>*
<feature ... />* <property ... />*
</binding>*

Generic WSDL Document Format (4)

<service name="*xs:NCName*" interface="*xs:QName*" >
<documentation />*
<endpoint name="*xs:NCName*“
binding="*xs:QName*“ address="*xs:anyURI*"? >
<documentation />*
<feature ... />* <property ... />*
</endpoint>*
<feature ... />* <property ... />*
</service>*
</description>

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description targetNamespace="http://example.org/TicketAgent.wsdl20"
        xmlns:xsTicketAgent=http://example.org/TicketAgent.xsd
        xmlns:wsdl=http://www.w3.org/2006/01/wsdl
        xmlns:xs=http://www.w3.org/2001/XMLSchema
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xsi:schemaLocation="http://www.w3.org/2006/01/wsdl
        http://www.w3.org/2006/01/wsdl/wsdl20.xsd">
    <wsdl:types>
        <xs:import schemaLocation="TicketAgent.xsd"
        namespace="http://example.org/TicketAgent.xsd" />
    </wsdl:types>
    <wsdl:interface name="TicketAgent">
      <feature ref="http://example.com/secure-channel" required="true"/>
      <wsdl:operation name="listFlights"
        pattern="http://www.w3.org/2006/01/wsdl/in-out">
        <wsdl:input element="xsTicketAgent:listFlightsRequest"/>
        <wsdl:output element="xsTicketAgent:listFlightsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="reserveFlight"
        pattern="http://www.w3.org/2006/01/wsdl/in-out">
        <wsdl:input element="xsTicketAgent:reserveFlightRequest"/>
        <wsdl:output element="xsTicketAgent:reserveFlightResponse"/>
      </wsdl:operation>
    </wsdl:interface>
</wsdl:description>
```

IRI (Internationalized Resource Identifiers)

TicketAgent.wsdl

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:description targetNamespace="http://example.org/TicketAgent.wsdl"
    xmlns:xsTicketAgent="http://example.org/TicketAgent.xsd"
    xmlns:wsdl="http://www.w3.org/2006/01/wsdl" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <wsdl:types>
        <xs:import schemaLocation="TicketAgent.xsd"
            namespace="http://example.org/TicketAgent.xsd" />
    </wsdl:types>
    <wsdl:interface name="TicketAgent">
        <wsdl:operation name="listFlights" pattern="http://www.w3.org/2006/01/wsdl/in-out">
            <wsdl:input element="xsTicketAgent:listFlightsRequest" />
            <wsdl:output element="xsTicketAgent:listFlightsResponse" />
        </wsdl:operation>
        <wsdl:operation name="reserveFlight" pattern="http://www.w3.org/2006/01/wsdl/in-out">
            <wsdl:input element="xsTicketAgent:reserveFlightRequest" />
            <wsdl:output element="xsTicketAgent:reserveFlightResponse" />
        </wsdl:operation>
    </wsdl:interface>
</wsdl:description>
```

TicketAgent.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://example.org/TicketAgent.xsd">
    <element name="listFlightsRequest">
        <complexType> <sequence>
            <element name="depart" type="dateTime" />
            <element name="origin" type="string" />
            <element name="destination" type="string" />
        </sequence>  </complexType>
    </element>
    <element name="listFlightsResponse">
        <complexType> <sequence>
            <element name="flight" type="string" minOccurs="0" maxOccurs="unbounded" />
        </sequence> </complexType>
    </element>
    <element name="reserveFlightRequest">
        <complexType> <sequence>
            <element name="depart" type="dateTime" />
            <element name="origin" type="string" />
            <element name="destination" type="string" />
            <element name="flight" type="string" />
        </sequence>  </complexType>
    </element>
    <element name="reserveFlightResponse">
        <complexType> <sequence>
            <element name="result" type="string" />
        </sequence> </complexType>
    </element>
</schema>
```