Web Services Description Language – WSDL 1.1

Richard Hopkins
National e-Science Centre, Edinburgh
February 23 / 24 2005

Information Society

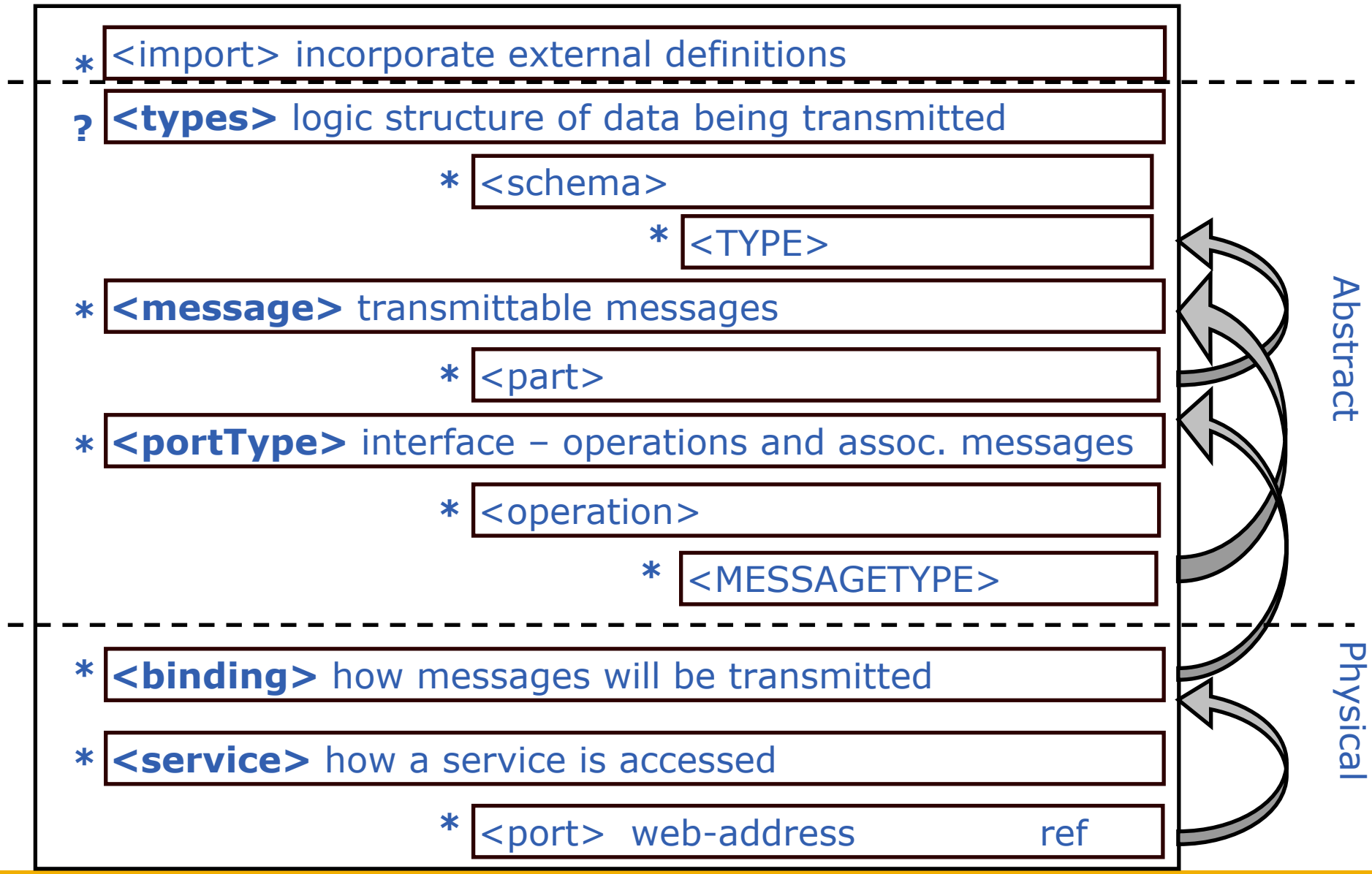
OUTLINE

- **Goals**
  - To be able to construct and read an WSDL services definition
- **Outline**
  - General
  - Core
  - Extensions

What is WSDL

- **An XML format**
- **For describing network services**
  - Which Operate either on
    - Documents
    - Procedure calls
  - Describes the **exposed** interface –
  - what the consumer sees of the service
  - Constitutes a contract with the client
    - Provides a specification of what is offered by the service provider which can be relied on by the service consumer
- **Which supports Separation of concerns**
    - logical structure of messages
    - binding to a specific underlying protocol
    - definition of a particular deployed service
  - To allow common definition and re-combination
- **Here using WSDL 1.1 – a W3C submission (March 2001)**
  - 2.0 – is a last call working draft (Aug 2004) – many differences

Structure

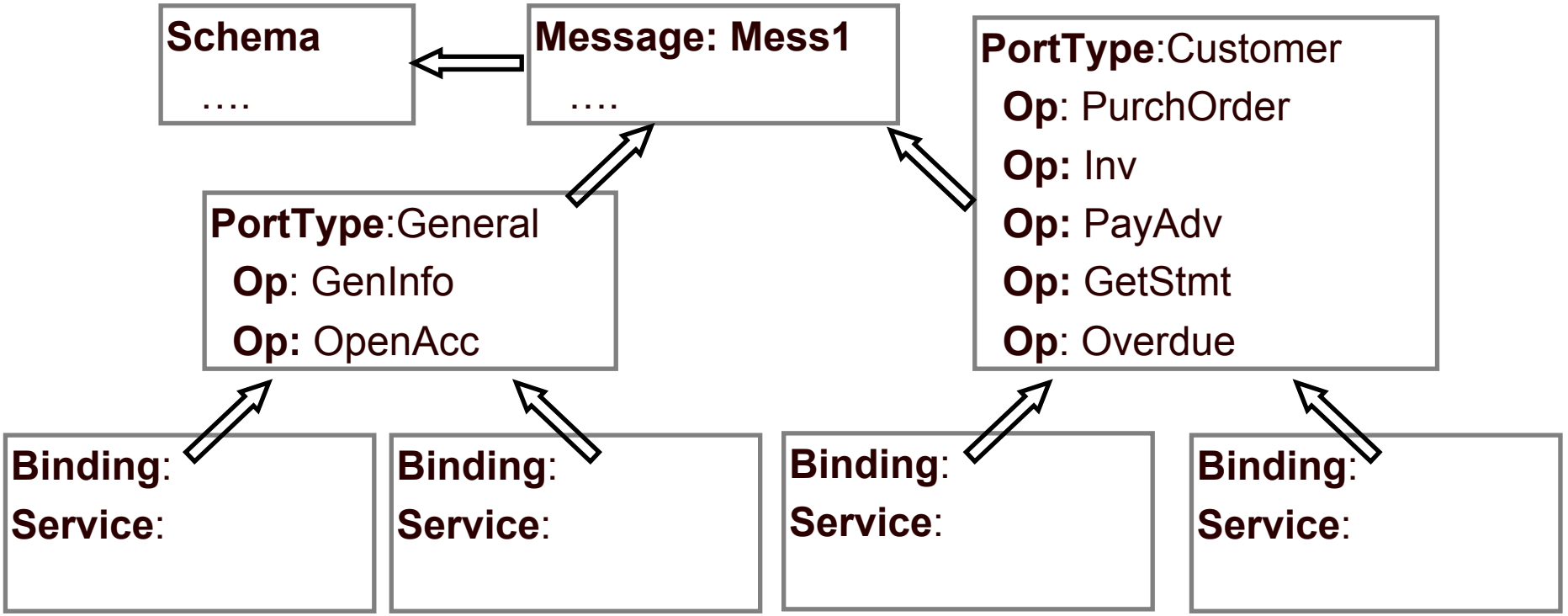

* <import> incorporate external definitions
? <types> logic structure of data being transmitted
* <schema>
* <TYPE>
* <message> transmittable messages
* <part>
* <portType> interface – operations and assoc. messages
* <operation>
* <MESSAGETYPE>
* <binding> how messages will be transmitted
* <service> how a service is accessed
* <port> web-address ref
Abstract
Physical

Services Structure Example

- **Example**
- **Company Provides two types of Service (PortTypes)**
  - General Service
    - Get general information
    - Open an Account
  - Customers Service (being a “Customer” = having an account)
    - Purchase Order
    - Invoice
    - Payment Advice
    - Get Statement
- **Both over two kinds of binding**
  - RPC
  - Email

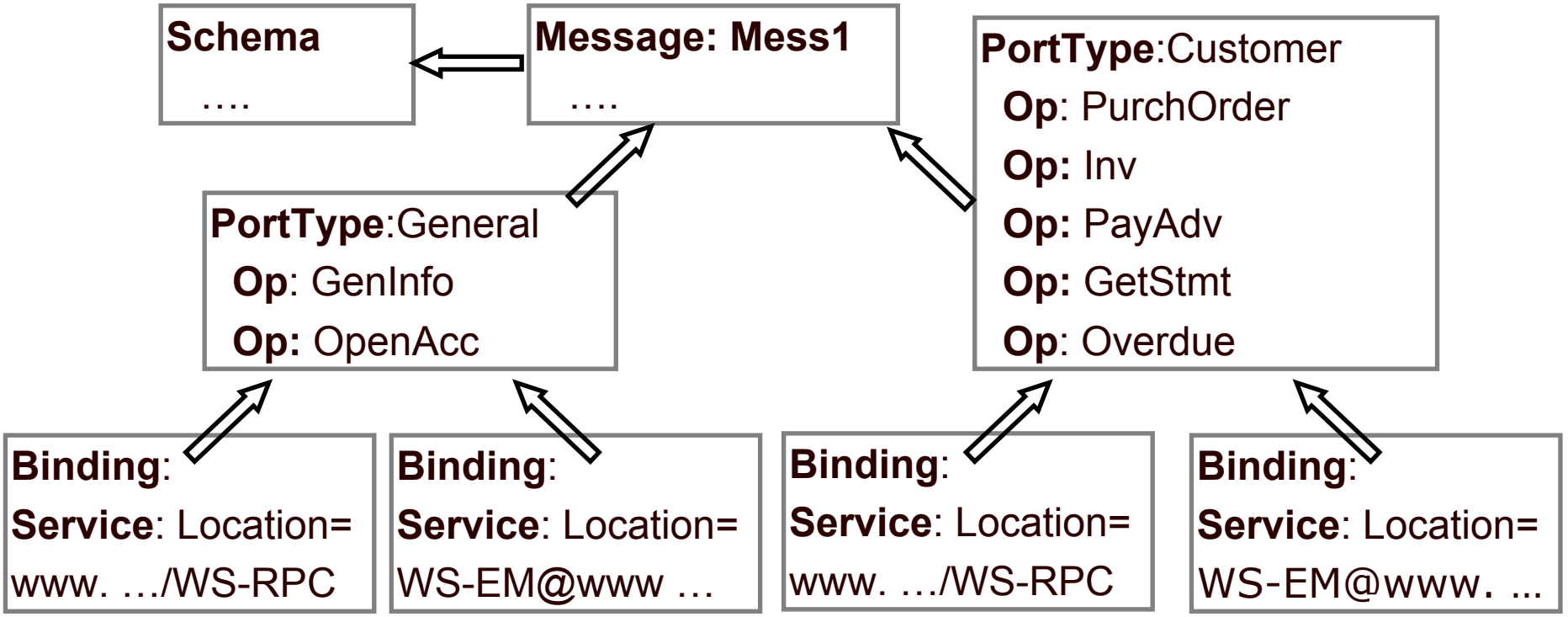
Services Structure Example

- Structure this into several WSDL/Schema files
  - (WSDL is like a specialised form of Schema)
  - For consistent changing of message definitions



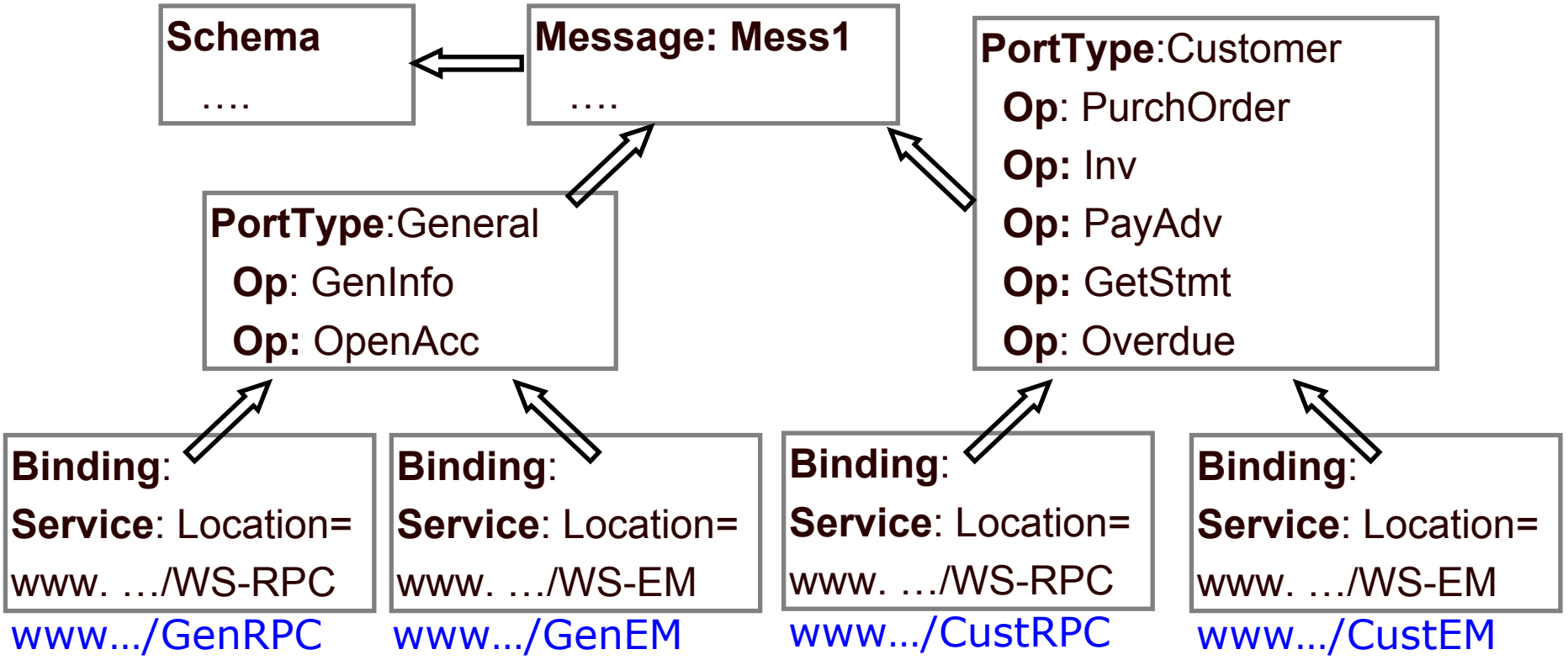
Services Structure Example

- One service location for all RPC services
  - http://www.company.org/WebServices/ws-RPC
- One service location for all e-mail services
  - mailto:WS-EM@www.company.org



Services Structure Example

- **Published WSDLs for the four service views,**
  - http://www.company.org/WSDLS/GenRPC.wsdl
  - http://www.company.org/WSDLS/GenEM.wsdl
  - http://www.company.org/WSDLS/CustRPC.wsdl
  - http://www.company.org/WSDLS/CustEM.wsdl

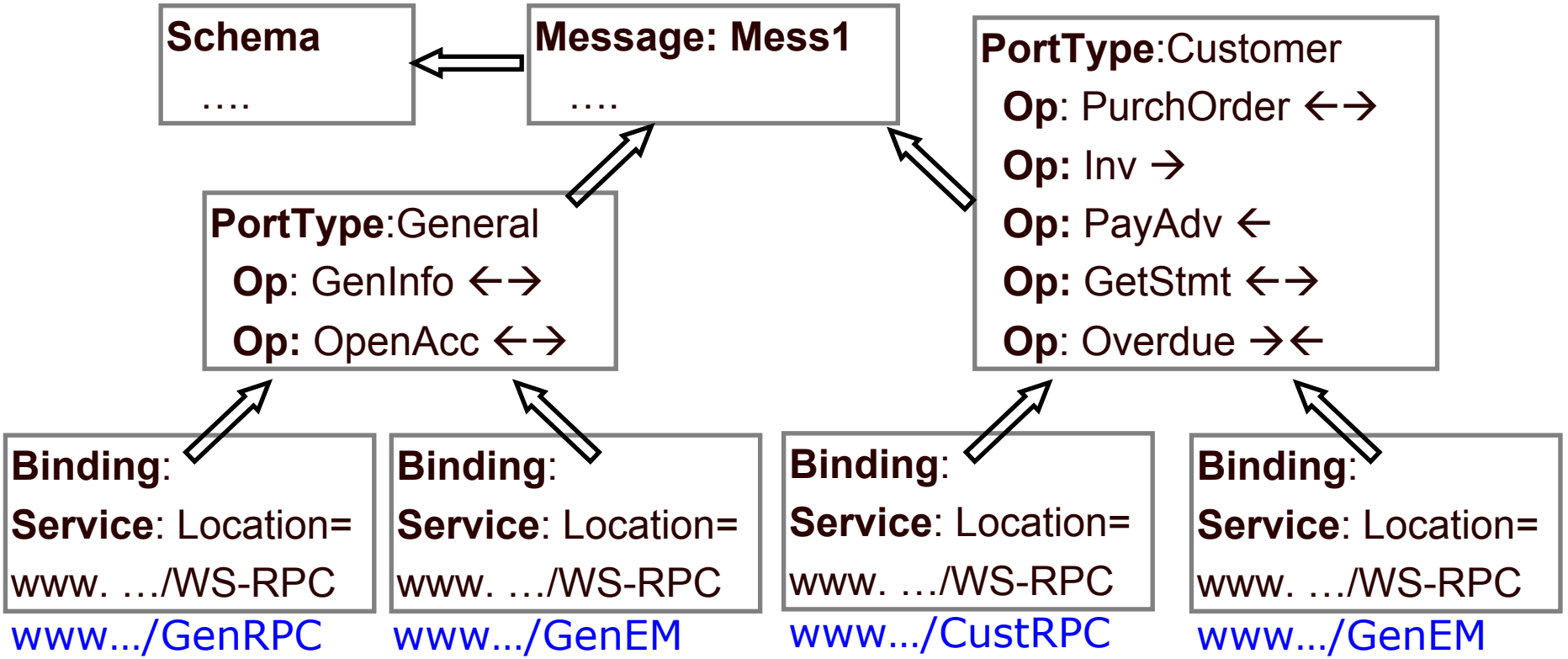


Services Structure Example

Four message patterns

← IN	One-way
→ OUT	Notify *
←→ IN then OUT	Request/Response
→← OUT then IN	Solicit/Response *

* Reversed roles
Provider
proactive = client
Consumer
reactive = server



CORE

- **Goals**
  - To be able to construct and read an WSDL services definition
- **Outline**
  - General
  - Core
  - Extensions

Organisational

```
<?xml version="1.0"?>
<wsdl:definitions
    name="ServiceAmasterWSDL"
    wsdl:targetNamespace = "X"
    xmlns:x="X"
    xmlns:wsdl="…xmlsoap.org/wsdl/"
    xmlns:soap ="…xmlsoap.org/wsdl/soap/"
    xmlns:y1="Y1"
    xmlns:y2="Y2"
    xmlns:z1="Z1"
    xmlns:z1="Z1" >
  <import namespace="Y1" location="Y1loc">
  <import namespace="Y2" location="Y2loc">
  <types>
    <wsdl:schema targetNameSpace="Z1"> … </>
    <wsdl:schema targetNameSpace="Z2"> … </></>
  <wsdl:message> …. </>…. </>
```

Name – optional, lightweight documentation

targetNameSpace – as for a schema

Use as default namespace

Use definitions for wsdl and its SOAP extension

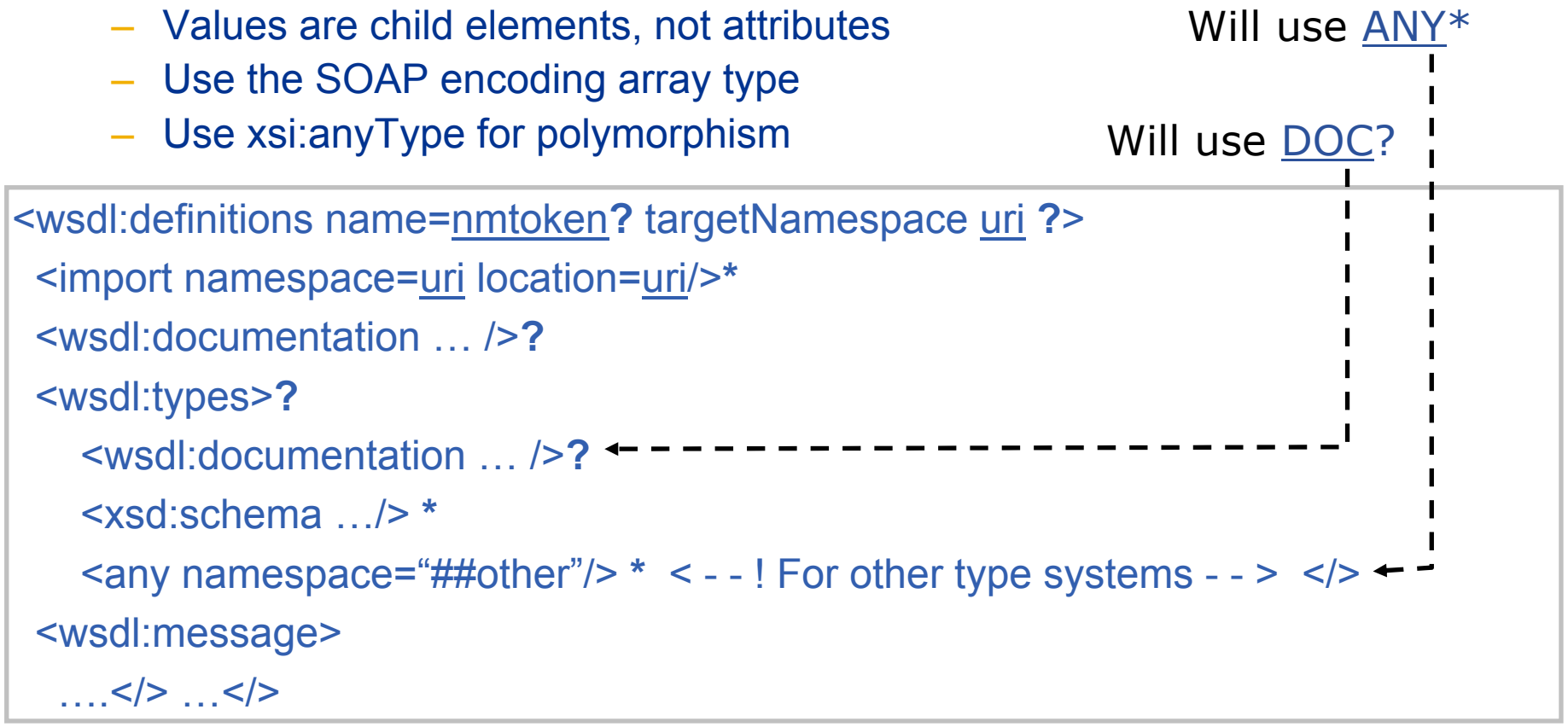
Namespaces prefixes for imported schemas/wsdls

Namespaces prefixes for in-line schemas –

Need distinct namespaces?

Type Definitions

- **Whether imported or in-line – no real difference**
- **Can be a schema or some other type definitions framework**
- **Schema is the standard, even if wire format is not XML**
  - Recommended to follow soap encoding style in formulating the schema definition
  - Values are child elements, not attributes
  - Use the SOAP encoding array type
  - Use xsi:anyType for polymorphism



Messages

- **Has a name – so message can be ref'd by a portType definition**
- **Consists of parts, each part**
  - Is a logical unit, e.g. a parameter
  - Has a name so that it can be referenced by a portType definition
    - E.g to put one part in a header and the other part in the body (non-SOAP)
  - Has a type – a Schema type definition or a Schema element definition
- **Normally use 1-part messages (no parts in WSDL2.0 ??)**

```
<wsdl:message name=POinM>
   <part name=accInfo  type=m:accInfoT>
   <part name=order  element=m:accInfoT>
</>
```

Input message for the purchase order operation

Two logical parts

```
<wsdl:message name=nmtoken>*                DOC?
   <part name=nmtoken element=qname? type=qname?>*
</>
```

PortTypes

- **PortType defines an interface**
- **A number of operations, named to be ref'd by Binding**
- **Each operation defines a number of messages which can be communicated as the interface to the operation**
  - Refer to a message definition
  - Provide a name for that message in this context, to be ref'd by Binding
- **Message Exchange Patterns**
  - ← One-way (Request) – **input**
  - ←→ Request-Response – **input output fault***
  - →← Solicit-Response – **output input fault***
  - → Notify – **output**

```
<wsdl:portType name=nmtoken>*                                   DOC?
   <wsdl:operation name=nmtoken>*                               DOC?
         <wsdl:input name=nmtoken? message=qname>?              DOC?</>
          <wsdl:output name=nmtoken? message=qname>?            DOC?</>
          <wsdl:fault name=nmtoken? message=qname>*             DOC?</>
</></>
```

Request - Response

- **Most Common Pattern**
  - Message to service provider; reply to service consumer
- **A logical patttern, Binding might be either**
  - An HTTP request/response
  - Two HTTP requests

```
<wsdl:portType name=CustomerPort>
   <wsdl:operation name=PurchOrder>
      <wsdl:input name=PurchOrderRequest message=POinM>
      <wsdl:output name=PurchOrderResponse message=DeliveryScheduleM>
      <wsdl:fault name=PurchOrderDuffAccFM message=DuffAccFM>
   ….. </>
  <wsdl:operation …> </> …</>
```

```
<wsdl:message name=POinM>
   <part name=accInfo  type=m:accInfoT>
   <part name=order  element=m:accInfoT>
</>
```

Default message name –
operation + request/response

Solicit - Response

- **Backwards two-way Pattern**
  - Message from service provider to consumer; reply from consumer to provider
- **Example – "overdue payment"**
  - Company sends this notification to the cusomer and expects a response

```
<wsdl:portType name=CustomerPort>
   …
   <wsdl:operation name=Overdue>
     <wsdl:output name=OverdueSolicit message=OverdueOutM>
     <wsdl:input name=OverdueResponse message=ExcuseInM>
     <wsdl:fault name=OverdueThreatOutFM message=ThreatOutFM>
   ….. </>
  <wsdl:operation …> </> …</>
```

Wrong order!

Default message name – operation + solicit/response

Single Message

- **Notify**
  - Message from service provider to consumer, with no reply
  - Example – "Invoice"
    - Send an invoice
- **One-way -- Request with no reply**
  - Message from service consummer to provider
  - Example – "payment advice"
    - Company gets notification from customer that a payment has been made

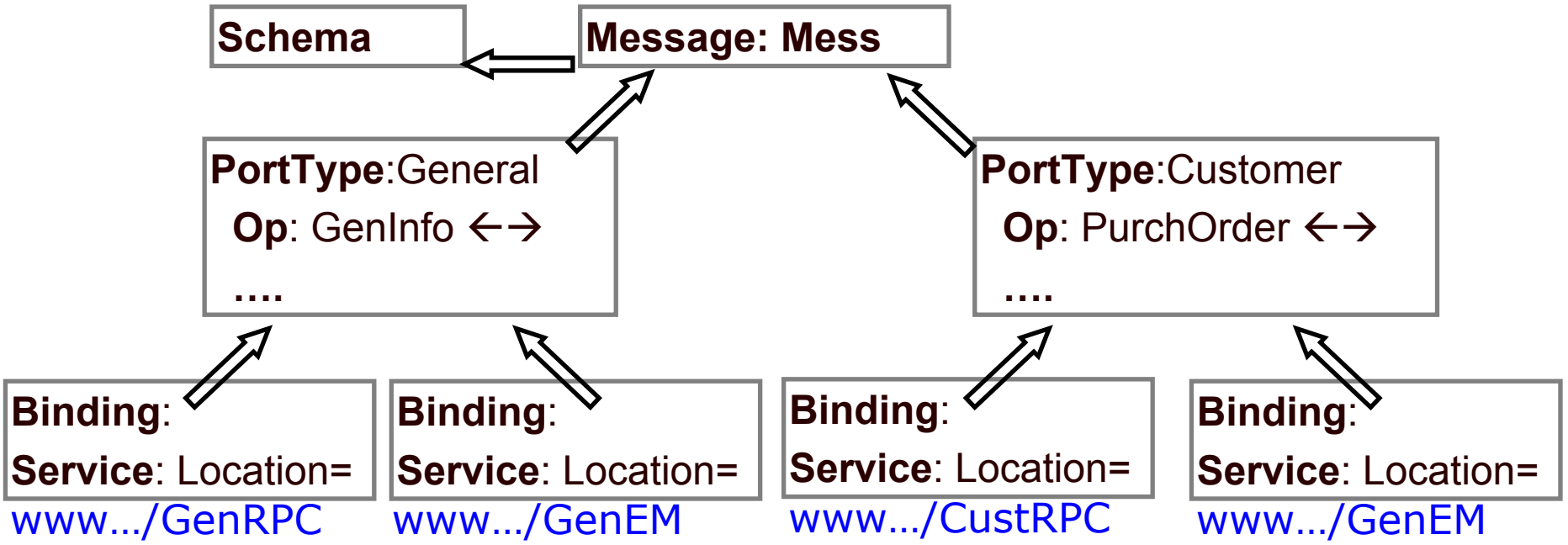
```
<wsdl:portType name=CustomerPort>
   …..
   <wsdl:operation name=Inv>
     <wsdl:output name=Inv message=InvoiceOutM> </>
   <wsdl:operation name=PayAdv>
     <wsdl:input name=PayAdv message=PaymentAdviceInM> </>
 </>
```

Can't have fault message

Default message name – operation

Binding - General

- **A Binding defines**
  - For a particular PortType – named as its “type”
  - Particular message format and communication protocol details
    - By extensiblity point
    - A standard extension is SOAP binding
- **Can have multiple bindings for one PortType**
  - Different modes in which it can be accessed



Binding General - Example

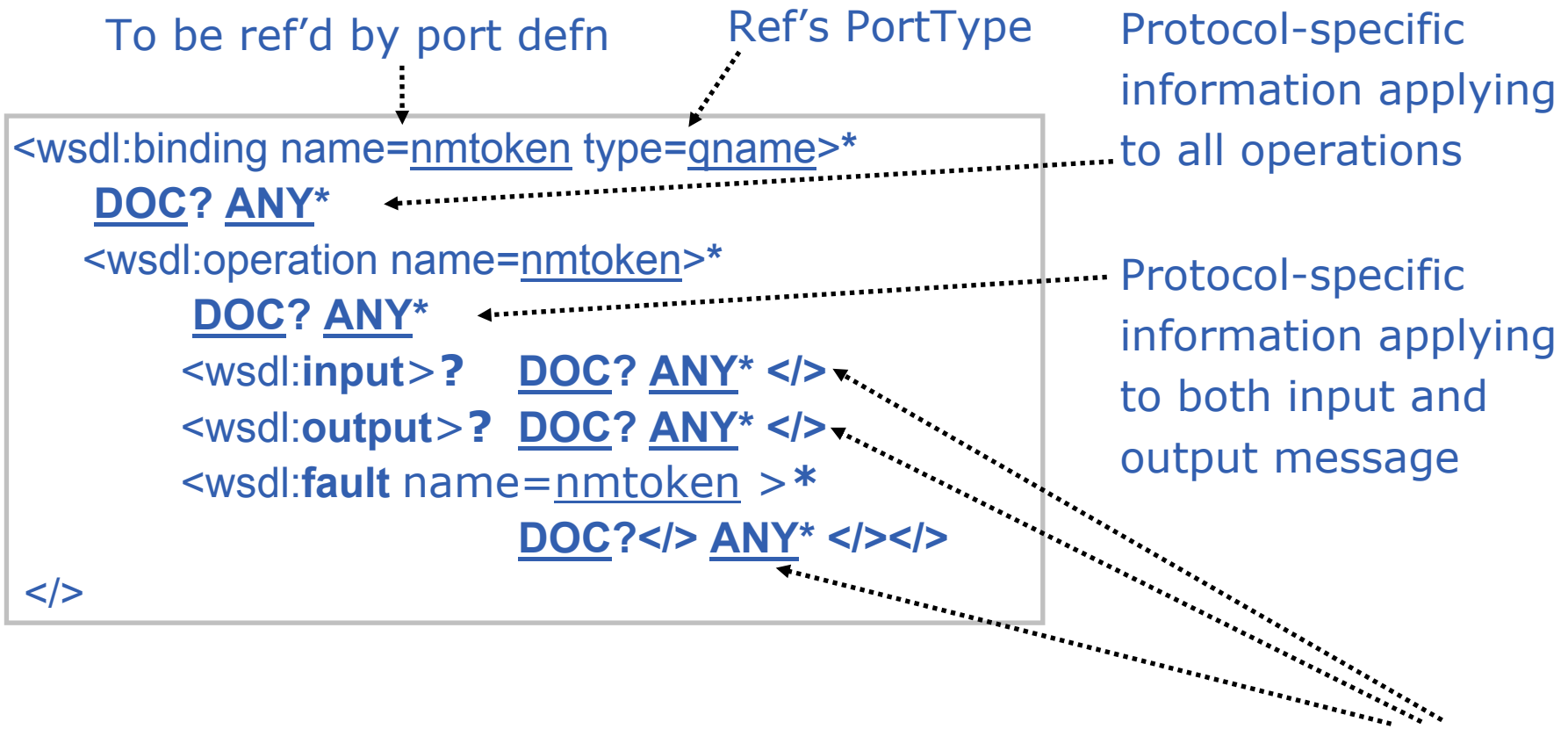
- **For binding of the Customer PortType**

```
<wsdl:binding name=CustPortRpcB type=CustomerPort>
                                                    … some binding info …
   <wsdl:operation name=PurchOrder>
     <wsdl:input>                                   … some binding info … </>
     <wsdl:output>                                  … some binding info … </>
     <wsdl:fault name=OverdueThreatOutFM>           … some binding info … </>
          … other faults …
    </>
    <wsdl:operation name=Inv>
     <wsdl:output>                                  … some binding info … </>
          …. Other operations ….

</>
```

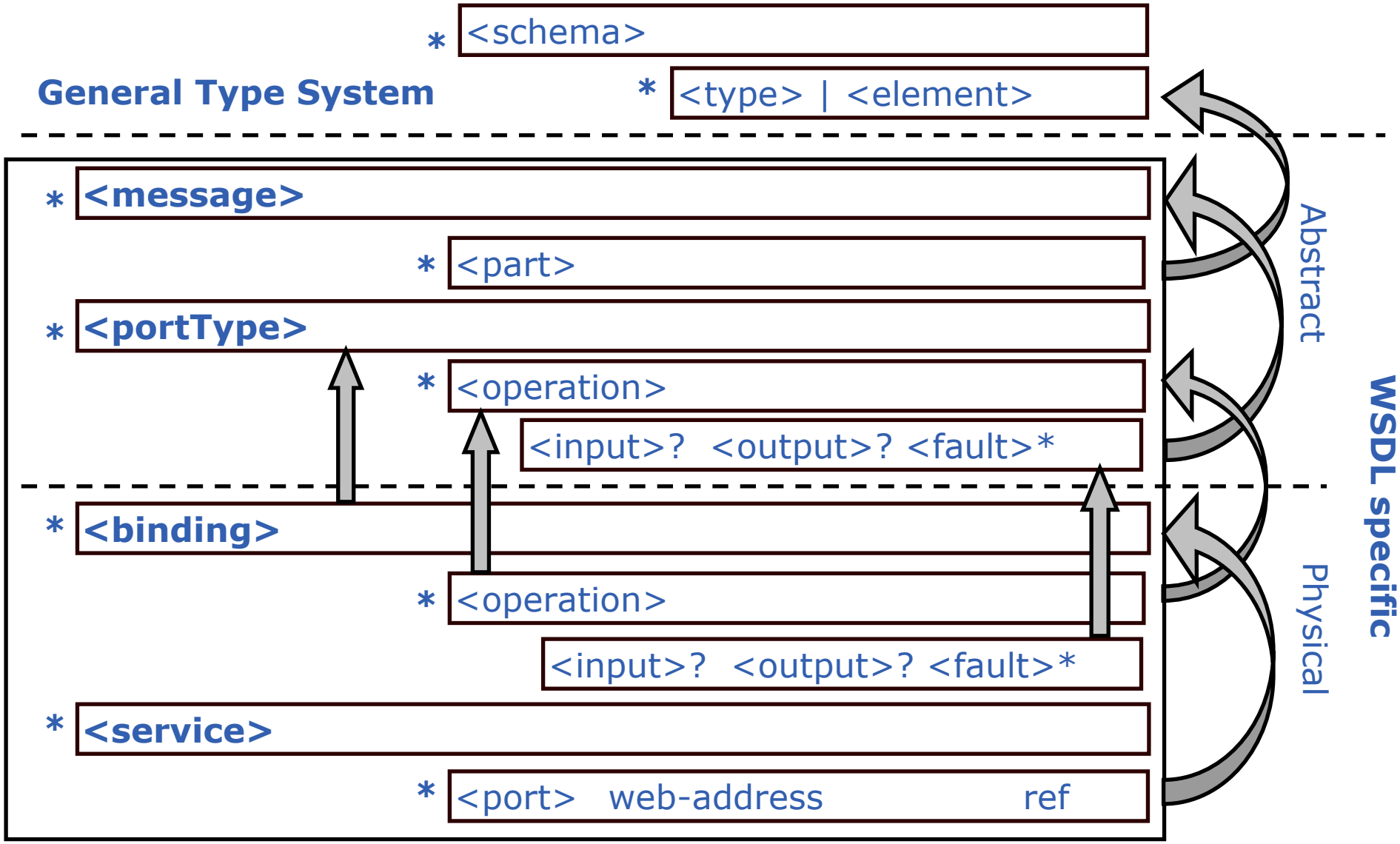
Binding - General Syntax

- **Binding mirrors structure of associated PortType**
  - Operations available and input / output / faults for each



1. Protocol specific information applying to particular message
2. How message parts map into protocol and data formats

Structure Review



Service Definition

- Can have multiple services in one WSDL definition document
- Each Service can have multiple ports
  - Each port is an individual endpoint - URL
- For WSDL 2.0 – all ports of a service must have the same portType
  - Can have different portTypes in WSDL 1.1 –
    consumer may need all functionalities for the service to be useful
- Two ports having the same portType means same semantics

```
<wsdl:binding name=CustPortRpcB type=CustomerPort> …  </>
```

```
<wsdl:service name=CustRpcS>
    <wsdl:port name=CustRpcSP binding=CustPortRpcB>
        <soap:address location= "http://www. ... .org/WebServices/ws-RPC">
```

```
<wsdl:service name=nmtoken>*                    DOC?
    <wsdl:port name=nmtoken  binding=qname >*
        DOC? ANY* </>
    ANY* </>
ANY* </wsdl:definitions>
```

An extension for SOAP binding
To give the port's address

EXTENSIONS

- **Goals**
  - To be able to construct and read an WSDL services definition
- **Outline**
  - General
  - Core
  - Extensions

Binding Extensions

- **There are a number of defined bindings**
  - **SOAP – identify the SOAP standards**
    - Transport
      - *Over **HTTP***
      - *….*
    - Style
      - ***RPC***
      - ***Document***
    - Use
      - ***Literal***
      - *Encoded*
  - HTTP
  - MIME
- **SOAP over HTTP, Literal is most commonly used**
  - all we will deal with here

The Soap Binding Extension

```
<wsdl:binding name=nmtoken type=qname>*
    <soap:binding   style= “rpc”|“document”?   transport=uri? />
   <wsdl:operation name=nmtoken>*
      <soap:operation   soapAction=uri?   style=“rpc”|“document”? >?
        <wsdl:input>?   ... </>
        <wsdl:output>? ... </>
        <wsdl:fault name=nmtoken >* ...  </></></></>
```

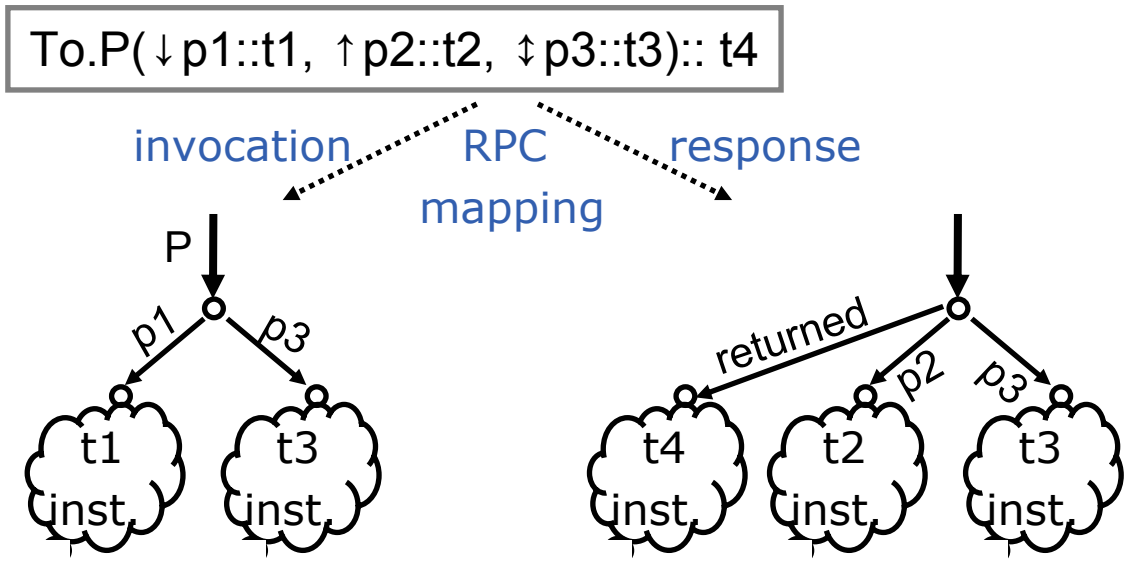
For ANY in general definition

- <soap:binding ..> says SOAP structure – envelope, header body
- style – how message parts are mapped into the body
  - Can: specify it for each operation, provide a default for all operations
  - If not specified at all, default is “document”
- transport – a SOAP transport binding, e.g. http://schemas.xmlsoap.org/soap/http
- soapAction – only for SOAP HTTP binding and there mandatory

  – specifies the value for the SOAPAction header (at HTTP level) e.g. http://www.company.org/soapActions/PO

  for document style this is what says what “operation”

RPC vs Document



- **RPC**
  - Hint that this is best dealt with as a procedure call (/return)
  - Message parts are parameters which are wrapped as one component of Body
  - As in the SOAP RPC standard
- **Document**
  - This is a document to be processed – message parts are directly in body
  - Wrapped convention – single message part – looks like RPC style

RPC vs Document

```
<wsdl:message name=POinM>
    <part name=accInfo  type=…>
    <part name=order  element=…>
</>
```

```
<wsdl:message name=POoutM>
    <part name=Result  type=…>
    <part name=delivSched  type=…>
</>
```

```
<wsdl:operation name=PurchOrder>
      <wsdl:input name=PurchOrderRequest message=POinM>
      <wsdl:output name=PurchOrderResponse message=POoutM> ….. </>
```

```
<env:Body>
  <m:PurchOrderRequest>
    <accInfo> … </>
    <order> … </></></></>
```

⟹

```
<env:Body>
  <m:PurchOrderResponse>
    <Result> … </>
    <delivSched> … </></></></>
```

RPC Actual messages

```
<env:Body>
  <accInfo> … </>
  <order> … </></></></>
```

⟹

```
<env:Body>
  <Result> … </>
  <delivSched> … </></></></>
```

Document Actual messages

SOAP - in/out bodies

```
<wsdl:binding> …
  <wsdl:operation> …
    <wsdl:input>?
        <soap:body parts="nmtokens"? use="literal"|"encoded"?
                   encodingStyle="uri-list"? namespace="uri"?>
        <soap:header … >*
            <soap:headerfault … />* </></>
        <wsdl:output>? ditto </>
        <wsdl:fault name=nmtoken >* …  </></></></>
```

- Parts - space-separated list of message parts to be included in soap body (default is all) – rest are placed elsewhere
- Use = "Encoded" : each message part ref's a type – an abstract type which is encoded by the sheme(s) identified in encodingStyle
- Use = "literal" : each message part ref's a type or element which gives the concrete format (encodingStyle may hint at the format used)
- Namespace: the namespace to be used for the outermost elements

SOAP - in/out headers

```
<wsdl:binding> …
 <wsdl:operation> …
  <wsdl:input>?
     <soap:body ..?>
     <soap:header message="qname" part="nmtoken"
           use="literal"|"encoded"?  encodingStyle="uri-list"?   namespace="uri"?>*
        <soap:headerfault message="qname" part="nmtoken"
           use="literal"|"encoded"?  encodingStyle="uri-list"?   namespace="uri"?>*
</>
        <wsdl:output>? ditto </>
        <wsdl:fault name=nmtoken >* …  </></></></>
```

Header –

- Message and part jointly identify a message part that defines the header type
- Use, encodingStyle, namespace – as for body; Style taken as document.

Headerfault -

As header – format of error messages relating to this header (such faults mut be reported in headers.

SOAP - Fault

```
<wsdl:binding name=nmtoken type=qname>*
  <soap:binding   style= “rpc”|“document”?   transport=uri? >
    <wsdl:operation name=nmtoken>*
      <soap:operation   …  >?
      <wsdl:input>?   … </>
      <wsdl:output>? … </>
      <wsdl:fault name=nmtoken >*
          <soap:fault use=“literal”|“encoded”?
                        encodingStyle=“uri-list”?   namespace=“uri”?></></></></>
```

- Gives the name of the fault within the operation for the PorrType – that identifies a message that must have only one part
- Use, encodingStyle, namespace – as for body; Style taken as document.

A simple Example

- **Allows a lot fo flexibility --> complexity.**
- **Mostly will be simple**

```
<wsdl:binding name="CustPortRpcB" type="CustomerPort">
   <soap:binding   style="rpc"   transport="http://schemas.xmlsoap.org/soap/http" >
   <wsdl:operation name="PurchOrder">
        <soap:operation   soapAction="http://company.org/PO">
         <wsdl:input> <soap:body use="literal"/></>
         <wsdl:output> <soap:body use="literal"/></>
          <wsdl:fault name=OverdueThreatOutFM>
                <soap:fault use="literal"/></>
          … other faults …
```

Explore it

- A site where you can obtain WSDL definitions of services and see what SOAPmessages are produced
- http://xmethods.com/

The End

THE END