Web Services, WS-*, SOAP, WSDL, etc

Oxford University

Software Engineering Programme

Dec 2012

Contents

- Understanding WS-*
- SOAP
- SOAP examples

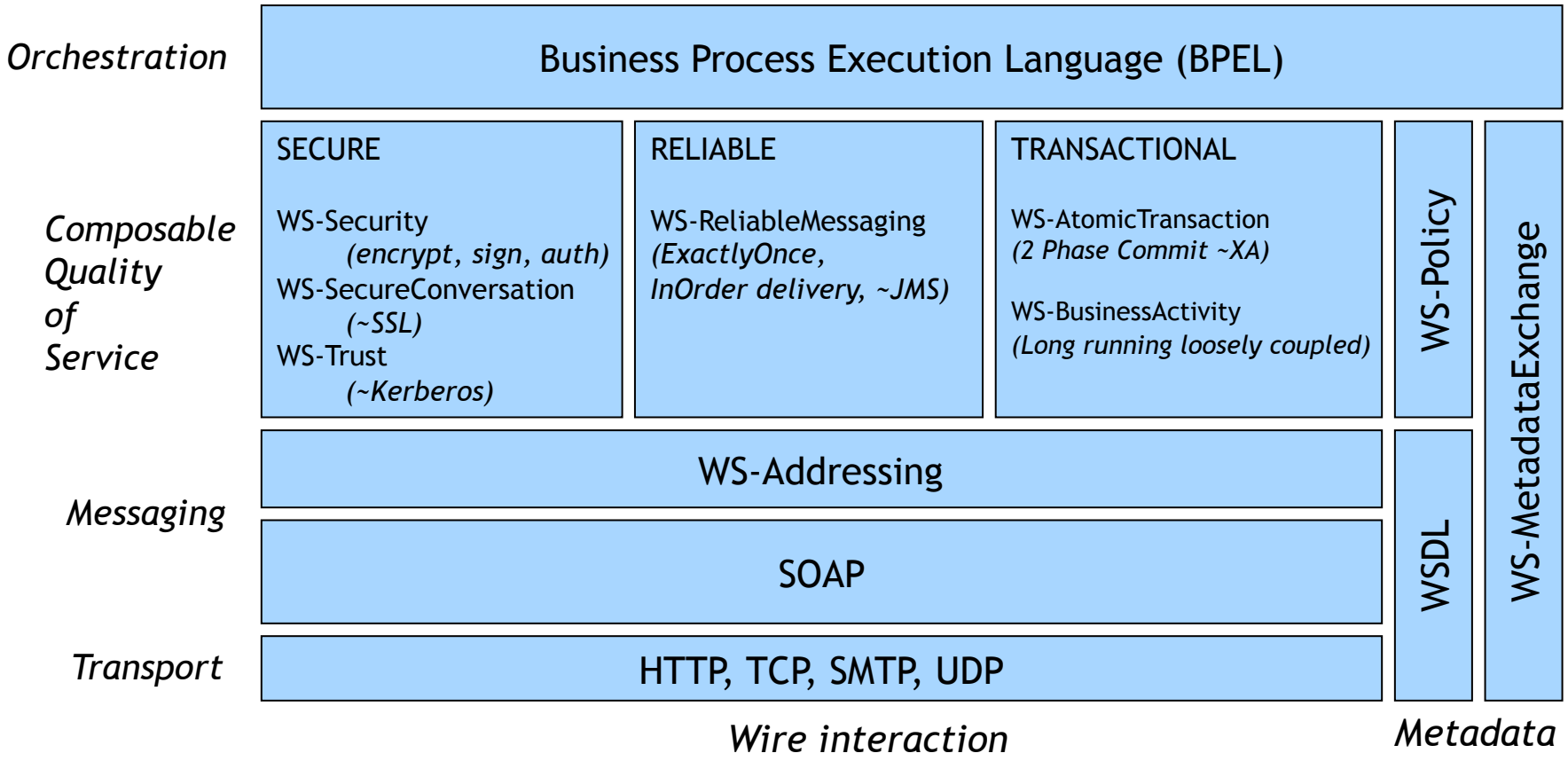
WS-* Standards

- A set of *extensible* and *composable* standards that work together
- Providing a common, standard, interoperable base to implement SOA

Composability

- The ability to use the various Web services standards together or apart
- Some examples from other spaces:
  - The Spring Framework allows you to choose whether or not to use transactions without rewriting your code
  - Java Security can be applied to existing code libraries by setting *policies*
- More about composability later

Key Web Services Standards



The Web services platform forms a complete framework for open standards enterprise middleware

Standards Bodies

Pulling it together:
Interoperability
Composability
Profiling

Qualities of Service:
Security, Transactions, Reliability
Resource management

Base standards:
XML, HTTP, SOAP,
WS-Addressing, Policy

SOAP

- SOAP was originally proposed by Microsoft and Developmentor
- IBM joined up shortly afterwards
- SOAP *used* to stand for:
  - Simple Object Access Protocol
  - BUT they soon realised it isn't that simple and isn't an object access protocol!
  - So now its just SOAP

A Sample SOAP Message

```
<soap:Envelope xmlns:soap="http://
  schemas.xmlsoap.org/soap/envelope/">
 <soap:Header/>
 <soap:Body>
  <getProductDetails       xmlns="http://
  warehouse.example.com/ws">
   <productID>827635</productID> </
  getProductDetails>
  </soap:Body>
</soap:Envelope>
```

A Sample SOAP Message (cont)

```
<soap:Envelope xmlns:soap="http://
  schemas.xmlsoap.org/soap/envelope/">
 <soap:Header/>
 <soap:Body>
  <getProductDetails       xmlns="http://
  warehouse.example.com/ws">
   <productID>827635</productID> </
  getProductDetails>
  </soap:Body>
</soap:Envelope>
```

The SOAP header provides a space for arbitrary headers to be added to the message`

A Sample SOAP Message (cont)

```
<soap:Envelope xmlns:soap="http://
  schemas.xmlsoap.org/soap/envelope/">
 <soap:Header/>
 <soap:Body>
  <getProductDetails       xmlns="http://
  warehouse.example.com/ws">
   <productID>827635</productID> </
  getProductDetails>
  </soap:Body>
</soap:Envelope>
```

The contents of the SOAP body element can be any valid XML that the parties wish to interchange

Main Features of SOAP

- <SOAP:Envelope>
  - This is a simple XML wrapper that holds the message and indicates this is a SOAP message
- <SOAP:Header>
  - An optional element that carries headers
  - This is how security, reliability, etc are composed
- <SOAP:Body>
  - The business payload of the message
  - An XML element

Further Aspects of SOAP

- Two versions:
  - SOAP 1.1
    - The mainly deployed version
    - Submitted as the proposed spec to the W3C
  - SOAP 1.2
    - The standardized version from the W3C

Further Aspects of SOAP Cont…

- Major differences
  - SOAP 1.2 is defined in terms of XML Infoset
    - This is the LOGICAL structure of an XML document and NOT the actual <> encoding
  - The namespace changed
  - Actor → Role (see next page)
  - Addition of none and UltimateReceiver roles
  - Different faults and fault model
  - Abstract binding framework

Roles

- A SOAP message may be passed from agent to agent
  - Known as SOAP intermediaries
- Each header can be targeted against a specific role (e.g. a security manager)
- A little used aspect of the specification
- Specific roles:
  - None – shouldn't be processed
  - Next – must be processed by whoever receives it
  - UltimateReceiver – the final consumer of the message

mustUnderstand Attribute

- Headers can be marked with an attribute:
  - <wsrm:Sequence mustUnderstand='true'/>
- This means that the processing agent must either be able to process this header, or send a fault

SOAP Encoding

- Is dead!
- SOAP encoding is a model that was initially presented
  - Allows a graph structure instead of a tree structure, and supports arrays
  - Pointers within the XML like object references
- WS-I Basic Profile bans it
- Pure XML is the cleaner approach

SOAP- A Simple Enveloping Model

- So why use SOAP over POX?
- Firstly, you don't have to
  - We will see how Axis2 supports both SOAP and POX without any change to your code
  - POX is a great model for simple lightweight communications
- But
  - SOAP has the "space" in the message to add security, reliability etc when you need it
  - Almost every enterprise integration model needs extensible headers
  - SOAP is much less reliant on the transport for headers, routing, etc

Using SOAP Headers

- Some simple examples
  - Add a signature to ensure the message isn't modified
  - Add a process identifier to track this message as part of a wider process
  - Add a userid so that end-to-end security can be guaranteed
  - Add a message number so messages can be resent if lost

What is a service definition?

What is a service definition?

- What does it do?
- Where is it?
- Who owns and runs it?
- Is it going to be up on Monday?
- What do I have to do to use it?
- How much does it cost?

What is a service definition?

- What does it do?
- **Where is it?**
- Who owns and runs it?
- Is it going to be up on Monday?
- **What do I have to do to use it?**
- How much does it cost?

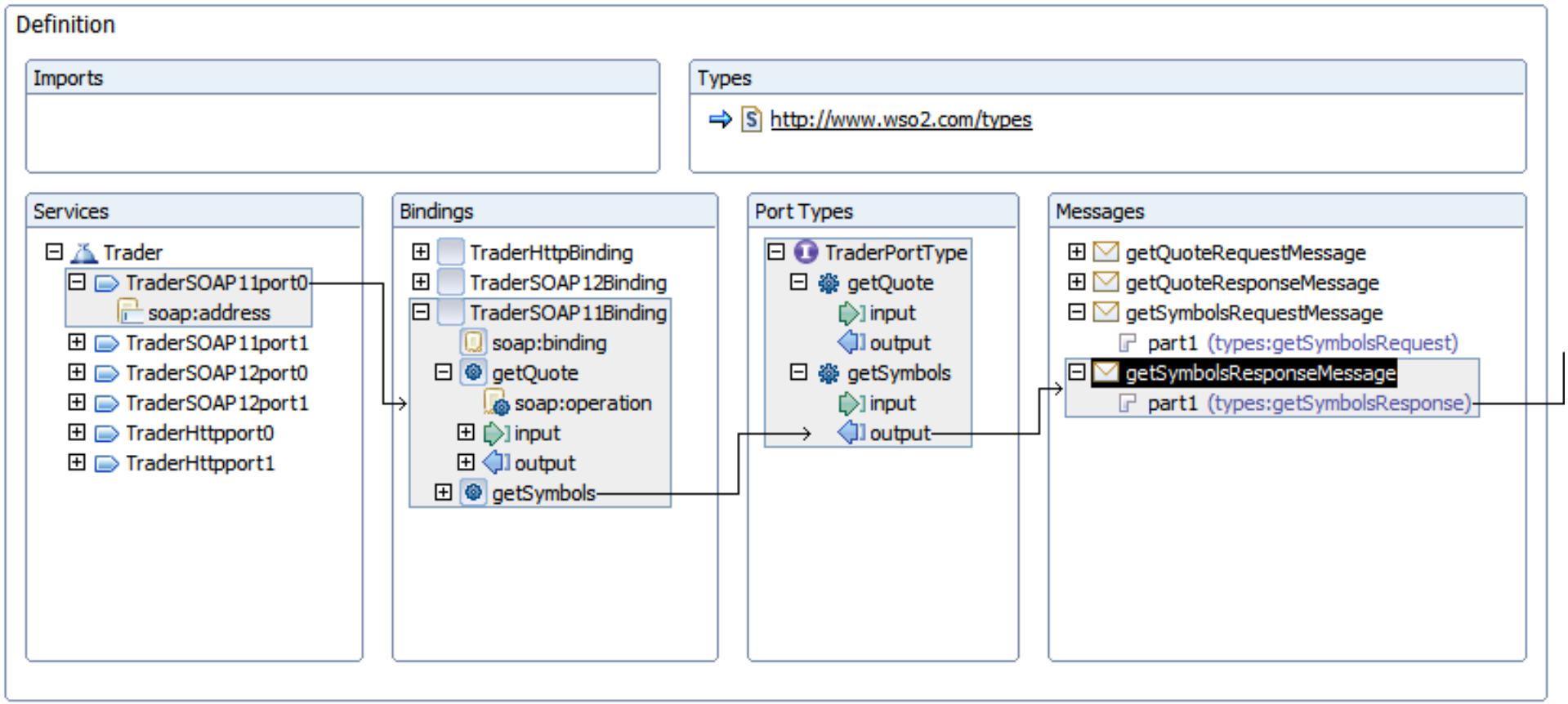
Web Services Description Language

- WSDL
  - Currently used version 1.1
  - Recently 2.0 made available
- Focuses on:
  - What the messages are
    - Schema
  - How they flow (in, in-out, etc)
    - Message Exchange Pattern
  - Where they are
    - Endpoint URLs

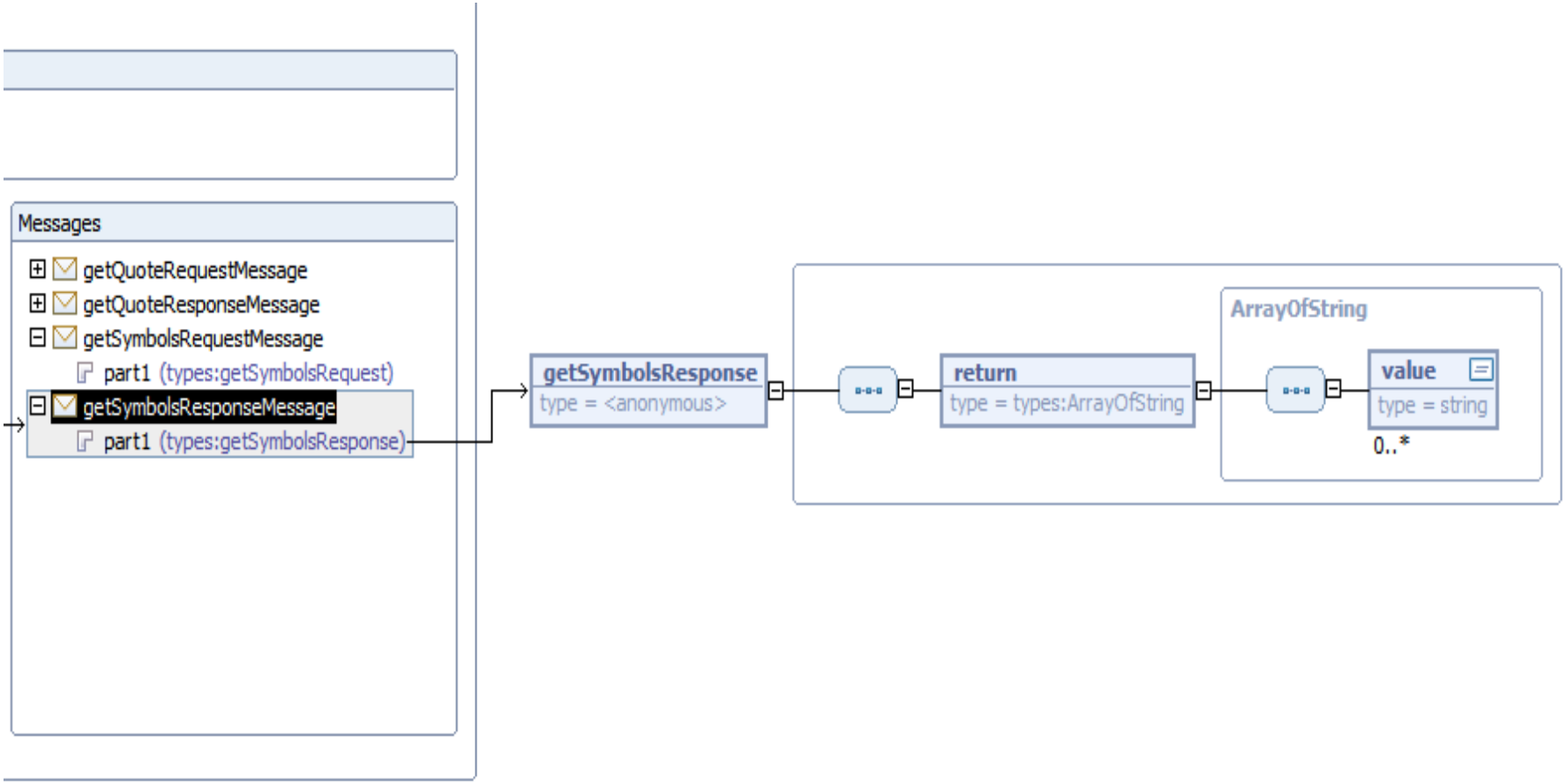
Abstraction

- WSDL splits into:
  - Interface / PortType
    - The abstract interface
  - The Binding
    - The mapping into SOAP or XML/HTTP (or +++)
  - The port
    - The actual endpoint or location

Graphical view of WSDL



WSDL link to Schema



WSDL type definitions

```
<wsdl:types>
  <schema>
   <element name="getQuoteRequest">
       …
   </element>
  </schema>
</wsdl:types>
```

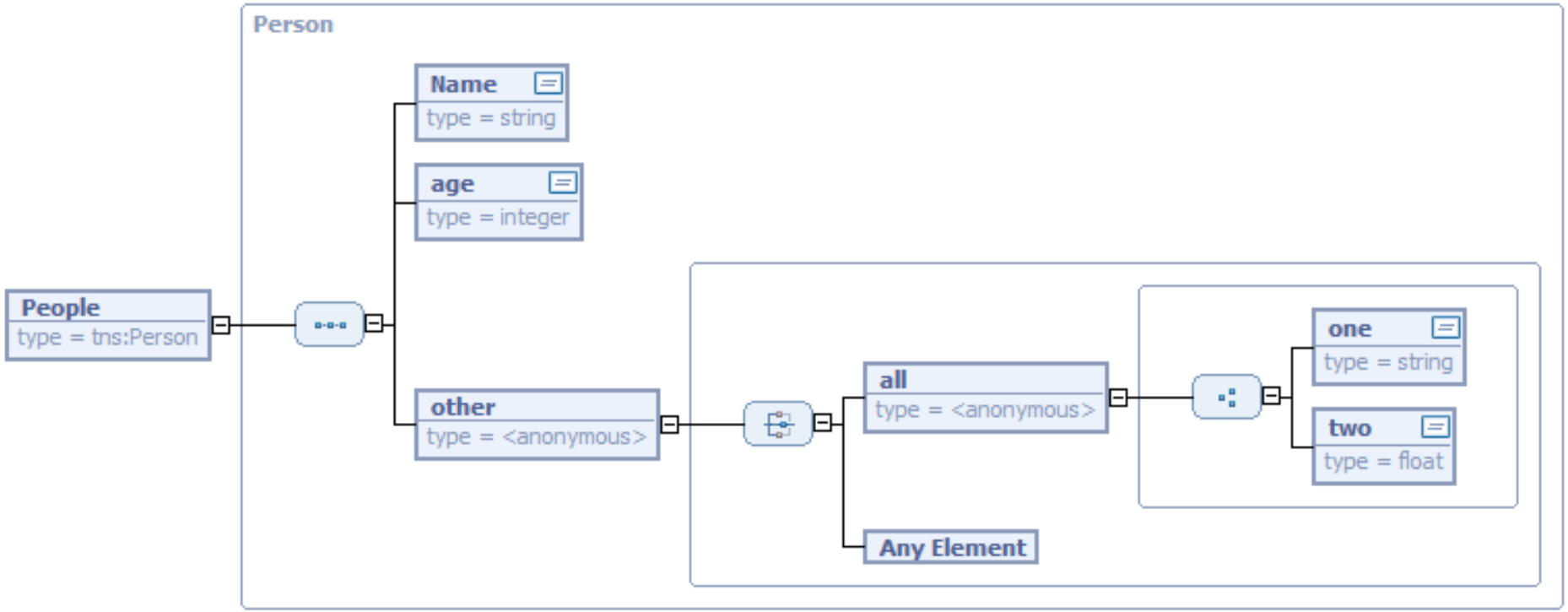
A simple schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema>
  <complexType name="Person">
    <sequence>
      <element name="Name" type="string“/>
      <element name="Company" type="string”/>
    </sequence>
  </complexType>

  <element name="People" type="tns:Person”/>

</schema>
```

Schema

- Simple types
  - e.g: integer, decimal, string, short, time, unsignedLong, date, any, hexBinary
- ComplexTypes
  - Named or inline
  - sequence, choice, all
- Multiplicity
  - 0..1,1..1,etc

Graphically



Messages

```
<wsdl:message
name="getQuoteRequestMessage">
    <wsdl:part
    element="types:getQuoteRequest"
    name="part1" />
</wsdl:message>
```

PortType

```
<wsdl:portType name="TraderPortType">
  <wsdl:operation name="getQuote">
    <wsdl:input
message="types:getQuoteRequestMessage" />
    <wsdl:output
message="types:getQuoteResponseMessage" />
  </wsdl:operation>
</wsdl:portType>
```

Bindings

```
<wsdl:binding type="types:TraderPortType“
    name="TraderSOAP11Binding">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http“
    />
    <wsdl:operation name="getQuote">
        …
    </wsdl:operation>
</wsdl:binding>
```

Service and Ports

```
<wsdl:service name="Trader">
   <wsdl:port
      binding="types:TraderSOAP11Binding"
      name="TraderSOAP11port0">
            <soap:address
               location=
    "https://localhost:9443/axis2/services/Trader“
            />
   </wsdl:port>
</wsdl:service>
```

WSDL styles of SOAP binding

- The WSDL portType is a theoretical definition
  - May have defined the message Parts in terms of Schema types or elements
- The SOAP binding says how this relates to the actual SOAP message
  - Elements => use=literal
  - Types => use can be literal or encoded, but almost always literal
- Literal means that the elements in the SOAP body are examples of the elements defined
- Encoded + encodingStyle means that the parts are defined as Types, and a particular concrete encoding is used to make up the SOAP body

WSDL styles continued

- Also the body can be defined as document or RPC
- In document style, the message parts appear directly in the SOAP body
- In RPC style, the first element in the body is a wrapper element, named after the operation

Variations

- doc/lit
  - Usually a single schema element defines the whole SOAP body. The element is not “encoded” in any way
- rpc/encoded
  - The message parts are parameters, defined using schema types
  - There is a wrapper element named as the op
  - Each object is mapped into XML using SOAP encoding (possibly pointers)
- rpc/literal
  - There are multiple parts defined as elements
  - Still a wrapper element
- doc/encoded
  - Never seen this, though probably someone used it once somewhere ☺

Wrapped doc/lit

- The “wrapped” style is useful way of mapping an object method to a SOAP operation
- doc/lit wrapped emulates this but hides it in the schema
- A single element, named the same as the op
- The first level of children are the parameters

WS-I Basic Profile

- Tried to clarify this mess:

R2705 A wsdl:binding in a DESCRIPTION MUST use either be a rpc-literal binding or a document-literal binding.

Granularity

- Fine-grained
- Are you exposing services or the internals of your application?
- Often the result of taking existing APIs and "service-enabling" them
- Coarse grained
- Generally considered better
- But can be too big
  - Require too much data passed in every request
  - Need to be useful in your enterprise

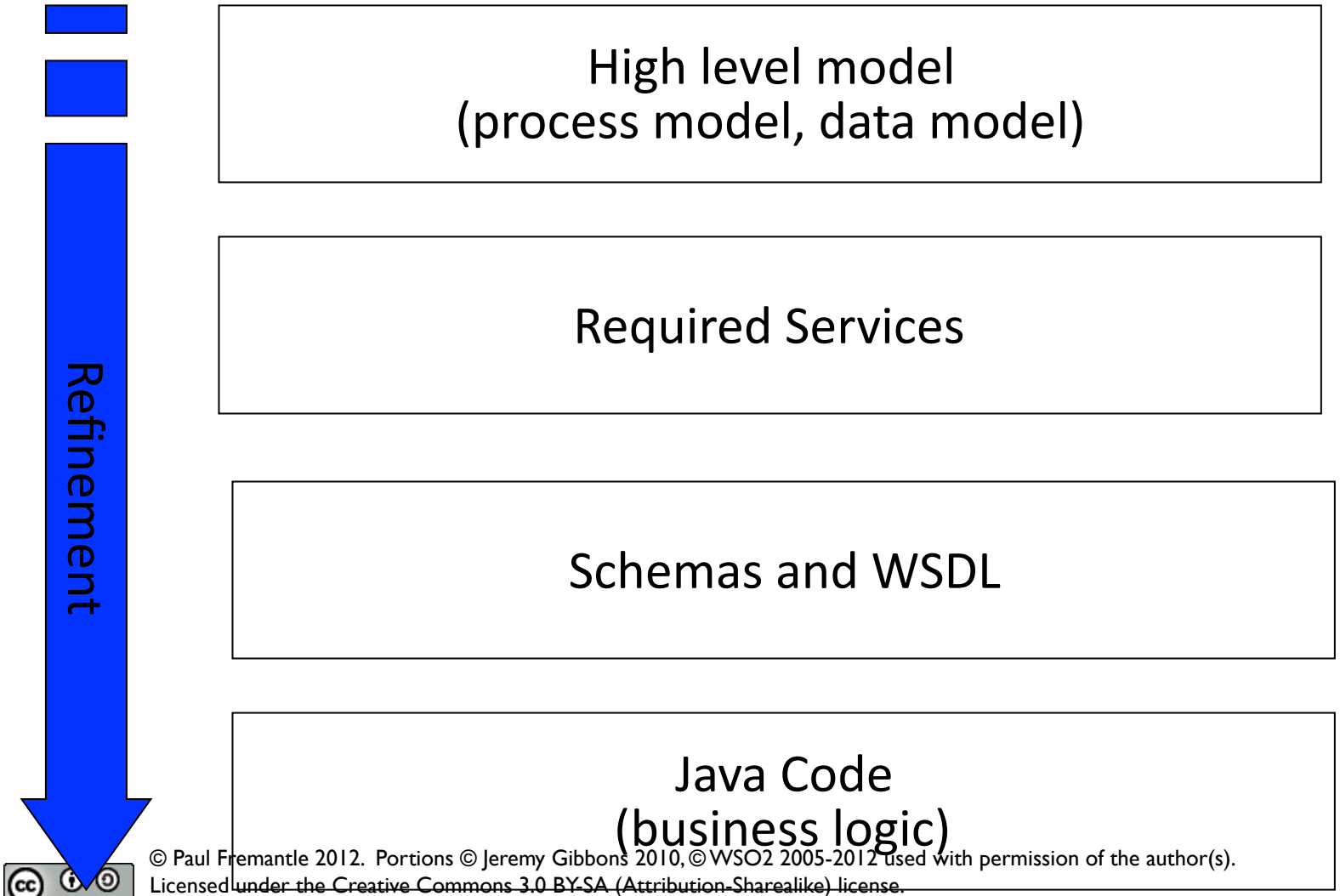
Bottom-up modelling

- Take existing code and expose as services
- Unlikely to expose *re-usable* services
  - Because the existing code was designed to be used within the application
- Quick way to get started

Top-down modelling

- A major undertaking
- Requires a good understanding of the business and business processes
- Various methodologies exist:
  - IBM's SOMA – Service Oriented Modeling Architecture
    - Based on a very high level business analysis
    - Refined down to processes and services
  - A simpler approach is BPEL process modeling and evolve the service definitions from the processes
- If this is a long process it may be counter-productive

Top down design



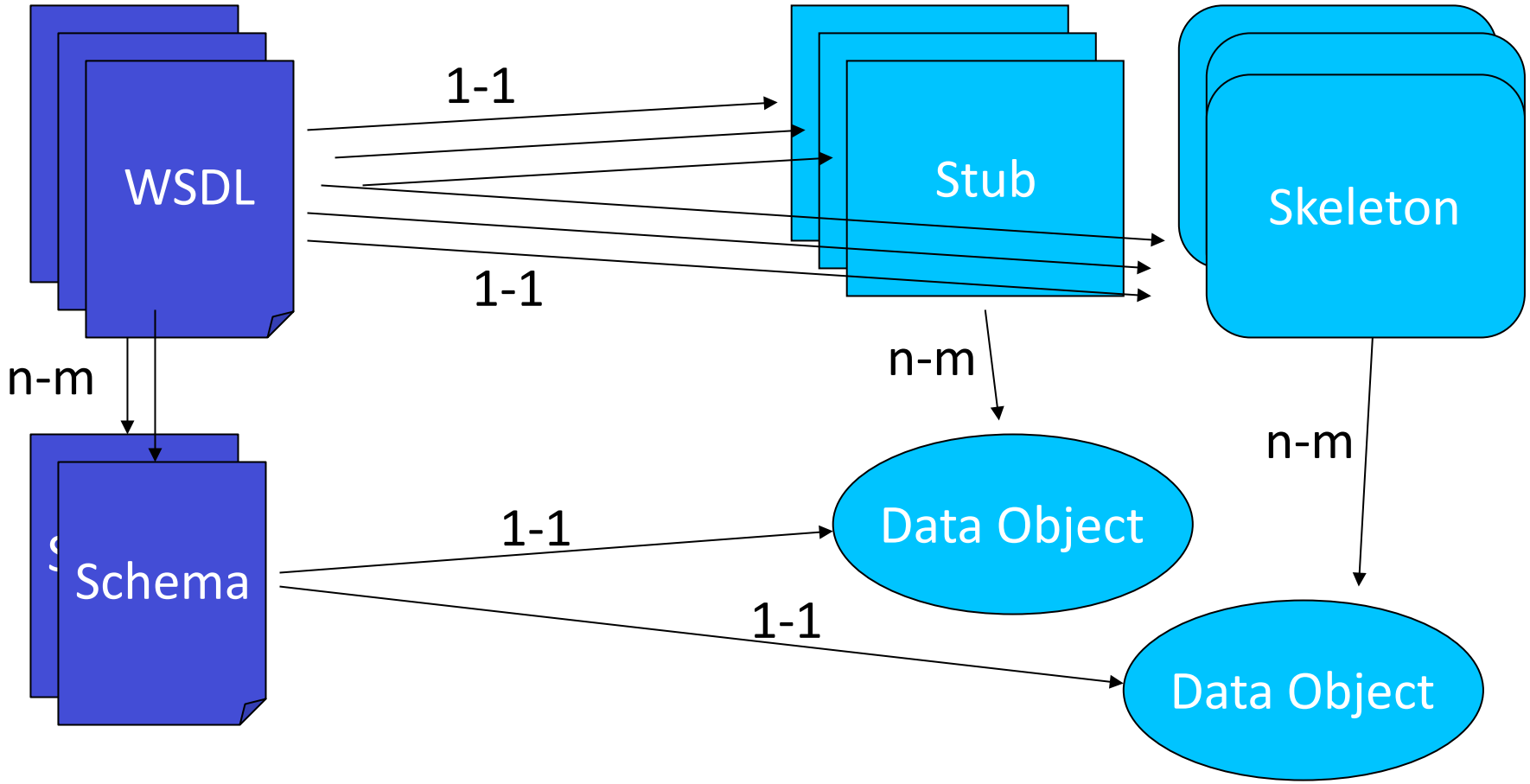
Why Contract First?

- *Advantages*
  - Agree the external interface
    - Good design principle
    - How Service Oriented Architecture is meant to be
    - Focus the mind on what is most important
  - Improves interoperability
    - WSDL first design leads to much more interop
- *Disadvantages*
  - Need to know WSDL and Schema syntax!

Contract First Development

1. Construct the WSDL
   - Numerous tools available to make it easier
2. Generate an empty service class from WSDL
   - Usually called a "skeleton"
   - Contains all the framework specific code except the business logic
3. Fill in the business logic
4. Deploy the service
   - Updates the WSDL
5. Build a client

Contract first development



Understanding the model

- Each WSDL may reference multiple data types from schema(s)
- Ideally the data types will be re-used across the set of services
- Each data type will be mapped to a Java object
- Each service will be mapped to a stub and/or a skeleton
- Those stubs and skeletons will use the Java data objects

Complexity of this approach

- WSDL was never designed to be written by mere mortals
  - Complex structure
  - Lots of pointers that have to be correct
    - E.g. Service -> port -> binding -> portType
- Schema is worse!
  - Highly complex spec
  - Element -> ComplexType links
- Only solution
  - Use a tool

Tools to Construct WSDL

- Construct WSDL
  - Eclipse WTP (Web Tools project)
    - Open Source tool – completely free
  - XMLSpy
    - Free version available
    - Licensed version has a very good WSDL editor
  - Stylus Studio
    - Proprietary tool

Eclipse Web Tools Platform

- http://www.eclipse.org/webtools/

Another approach

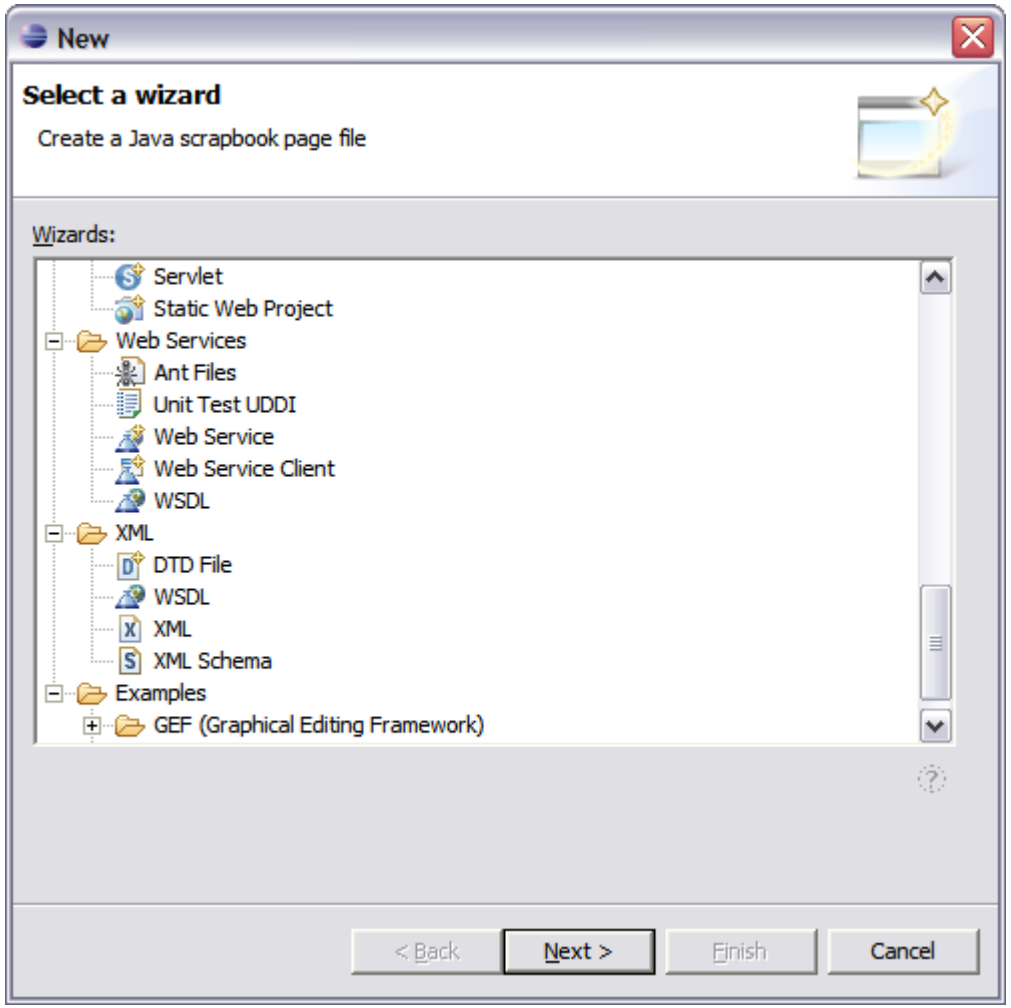
- Cheat!!!!!

- Write Java interfaces matching your “model”
- Run Java2WSDL
- Now “throw away” the Java
  - Your WSDL is now the authoritative model

Iteration

- Especially valuable when starting on SOA
  - Allows quick and simple first steps
  - Typically update service definitions as new users come on board
- Requires the right approach and attitude!
  - As well as the right infrastructure to support versioning and routing

Web Tools-WSDL and Schema



Where do you find WSDLs?

- ?wsdl
- Email, Web page, etc
- xmethods.net
- wsdlicio.us
- Registry

Summary

- WSDL is a key technology for *Interoperability*
- A standard XML based interface language
- Flexible binding to Schema
- VERY WELL TOOLED!