



Aspect-oriented Model-driven Service-oriented architecture

TTT presentation
2 April 2009

Luís Ferreira Pires



Motivation

- ▶ I heard during a keynote presentation of a well-known person at ECOWS 2007 an enthusiastic argument for applying aspect-oriented model-driven techniques to service-oriented architectures (actually web services)
- ▶ but he didn't know exactly how this could be done...
- ▶ The importance of this combination of topics was confirmed by yet another ECOWS keynote

20 January 2010 TTT presentation 2



Motivation

Groups has expertise in the areas of

- ▶ Aspect-oriented modelling and programming
- ▶ Model-driven architecture (engineering or development)
- ▶ Service-oriented architecture

Opportunity for a project involving many people from the SE group?

20 January 2010 TTT presentation 3



Objective

- ▶ Investigate whether it is meaningful to apply aspect-oriented modelling and model-driven architecture techniques to service-oriented architectures
- ▶ Investigate what has been done on this combination of topics (not comprehensive yet)
- ▶ Draw some conclusions (together?)

20 January 2010 TTT presentation 4



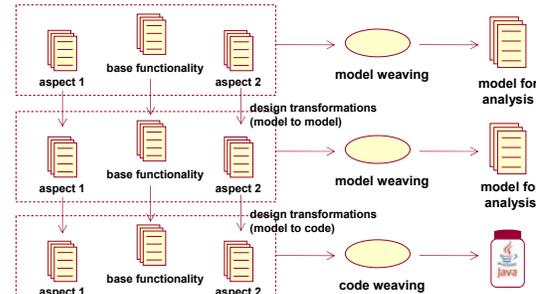
'I have a dream...'

- ▶ Define base functionality ('business logic') and aspects in different models at a high abstraction level
- ▶ Define how aspects influence base functionality
- ▶ Perform 'model weaving' for analysis whenever necessary
- ▶ Propagate these separate models via automated transformations through abstraction levels until the code level
- ▶ Perform code weaving after models are translated to code

20 January 2010 TTT presentation 5



'I have a dream...'



20 January 2010 TTT presentation 6

University of Twente
The Netherlands

Problem statement

- ▶ How to make this 'dream' come true for web services?
- ▶ What are the models and code in this case?
- ▶ Is this 'dream' reachable?
- ▶ What has already been done in this respect?

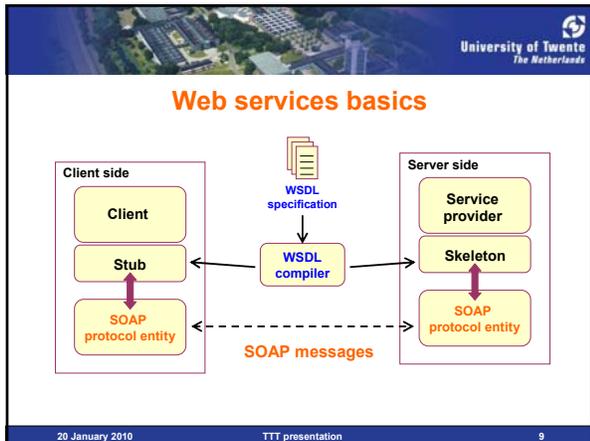
20 January 2010 TTT presentation 7

University of Twente
The Netherlands

Web services and aspects

- ▶ Web services is the most important technology available nowadays to implement Service-oriented architectures
- ▶ Web services architecture is 'aspects-ready'!
 - ▶ Automated generation of stubs and skeletons (excellent points for interception)
 - ▶ Separate standards to define non-functional properties (typical crosscutting concerns)
 - ▶ Intermediate nodes that can operate on SOAP headers (also potential points for interception)

20 January 2010 TTT presentation 8



University of Twente
The Netherlands

WSDL

Service developer provides a **description of the service** for the (potential) clients

- ▶ Which **messages** are related to each operation supported by the service?
- ▶ How are these **messages related**? e.g., **operation input and output**
- ▶ How are **SOAP messages exchanged**?

20 January 2010 TTT presentation 10

University of Twente
The Netherlands

WSDL

- ▶ **Port type** → logical collection of operations
- ▶ **Operation** → simple message exchange
- ▶ **Message** → unit of communication
- ▶ **XML schemas** are used as a **type system** for common understanding of data

20 January 2010 TTT presentation 11

University of Twente
The Netherlands

WSDL example: PurchaseOrder

```

<?xml:definitions name="PurchaseOrderService" targetNamespace="http://supply.com/PurchaseService/wsd1" >
  <wsdl:types>
    <xsd:schema targetNamespace="http://supply.com/PurchaseService/wsd1">
      <xsd:complexType name="CustomerInfoType">
        <xsd:sequence>
          <xsd:element name="CusName" type="xsd:string" />
          <xsd:element name="CusAddress" type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="POType">
        <xsd:sequence>
          <xsd:element name="PONumber" type="xsd:integer" />
          <xsd:element name="PODate" type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="InvoiceType">
        <xsd:all>
          <xsd:element name="InvPrice" type="xsd:float" />
          <xsd:element name="InvDate" type="xsd:string" />
        </xsd:all>
      </xsd:complexType>
    </xsd:schema>
  </wsdl:types>
  </?xml:definitions>
  
```

20 January 2010 TTT presentation 12

University of Twente
The Netherlands

WSDL example: PurchaseOrder

```

<wsdl:message name="POMessage">
  <wsdl:part name="PurchaseOrder" type="tns:POType" />
  <wsdl:part name="CustomerInfo" type="tns:CustomerInfoType" />
</wsdl:message>
<wsdl:message name="InvMessage">
  <wsdl:part name="Invoice" type="tns:InvoiceType" />
</wsdl:message>

<wsdl:portType name="PurchaseOrderPortType">
  <wsdl:operation name="SendPurchase">
    <wsdl:input message="tns:POMessage" />
    <wsdl:output message="tns:InvMessage" />
  </wsdl:operation>
</wsdl:portType>

```

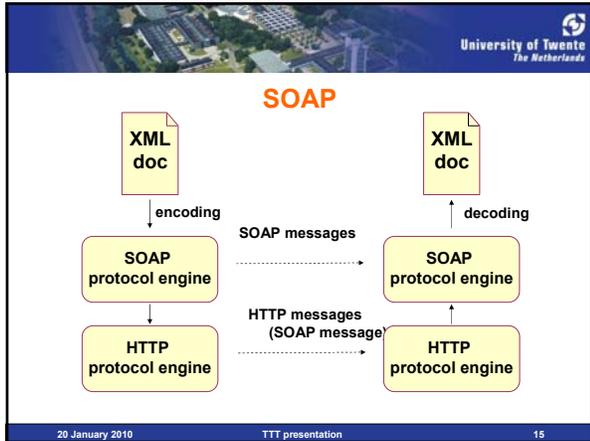
20 January 2010 TTT presentation 13

University of Twente
The Netherlands

WSDL and aspects

- ▶ WSDL only supports the description of the message syntax and (basic) exchange patterns to interact with a web service (and the bindings, which we can ignore for now ☺)
- ▶ Some 'traditional aspects' (transactional behaviour, security, QoS, etc.) are defined in separate documents, using other standards as WS-Policy, WS-Security, etc.

20 January 2010 TTT presentation 14

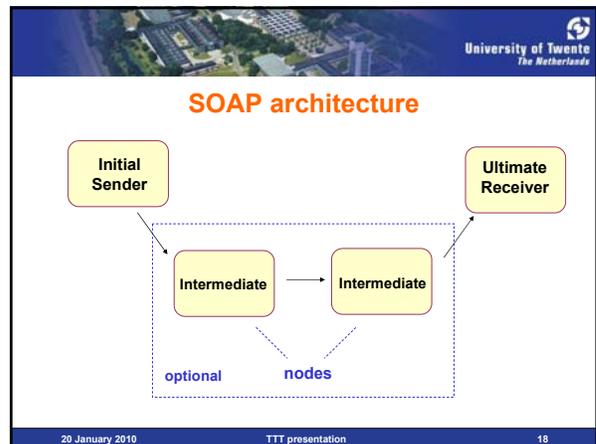
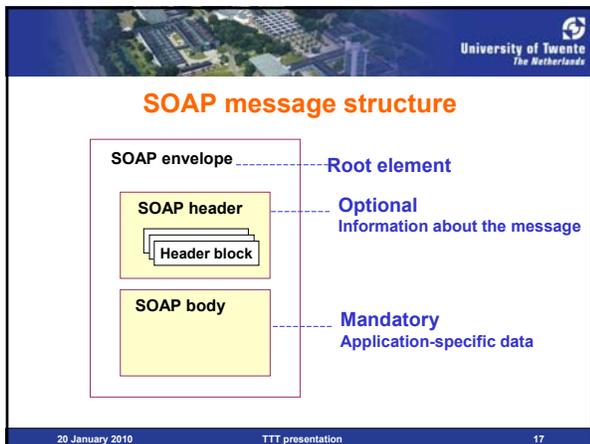


University of Twente
The Netherlands

SOAP messages

- ▶ SOAP message is a SOAP XML document instance
- ▶ SOAP message types are defined in an XML schema (<http://schemas.xmlsoap.org/soap/envelope/>)
- ▶ SOAP messages are carried as payload (user data) by other protocols
- ▶ SOAP messages are typically transported over HTTP (binding approved by WS-I BP 1.0)
 - ▶ As opposed to HTML documents, SOAP messages are not meant to be viewed by end users

20 January 2010 TTT presentation 16



University of Twente
The Netherlands

SOAP header

- ▶ Contains **additional information** primarily meant for intermediaries
- ▶ **Examples**
 - ▶ Transactional interactions (transaction ID)
 - ▶ Security (credentials)
 - ▶ Routing (instructions)
 - ▶ Debugging (e.g., logging)
- ▶ Intermediaries should **process and manipulate only SOAP headers** (not the SOAP body!)
- ▶ Nodes **'mind their own business'** → don't verify if other nodes have processed a message

20 January 2010 TTT presentation 19

University of Twente
The Netherlands

Aspects and web services

- ▶ Exploiting the general structure of a web service implementation
- ▶ Exploring the web service implementation code
- ▶ Applicable to crosscutting concerns like validation, exception management, caching, logging, instrumentation, authentication and authorisation
- ▶ Not 'model-driven'

20 January 2010 TTT presentation 20

University of Twente
The Netherlands

Entangled code to invoke a web service

```

public class HelloClient {
    private String endpointAddress;
    public static void main (String[] args) {
        try {
            endpointAddress = args[0];
            Stub stub = createProxy();
            stub.setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, endpointAddress);
            HelloIF hello = (HelloIF)stub;

            stub.setProperty(Stub.USERNAME_PROPERTY, username);
            stub.setProperty(Stub.PASSWORD_PROPERTY, password);

            String result = hello.sayHello("Testing");

            log ("HelloWorld result: " + result);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}

```

endpointAddress = args[0]; Stub stub = createProxy(); stub.setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY, endpointAddress); HelloIF hello = (HelloIF)stub;	Redirection
stub.setProperty(Stub.USERNAME_PROPERTY, username); stub.setProperty(Stub.PASSWORD_PROPERTY, password);	Authentication
String result = hello.sayHello("Testing");	Invocation
log ("HelloWorld result: " + result);	Logging
catch (Exception ex) { ex.printStackTrace(); }	Exception handling

20 January 2010 TTT presentation 21

University of Twente
The Netherlands

Aspects and web services

Parameter validation

Exceptions management

20 January 2010 TTT presentation 22

University of Twente
The Netherlands

Aspects and web services

Authentication / authorisation

20 January 2010 TTT presentation 23

University of Twente
The Netherlands

Service composition

- ▶ The traditional approach to service composition in web services is based on WS-BPEL
- ▶ Static definition of a process (workflow) in which web services are coordinated → orchestration
- ▶ WS-BPEL is an executable XML-based language
- ▶ WS-BPEL process is itself also a web service (hierarchical composition)

20 January 2010 TTT presentation 24

University of Twente
The Netherlands

Hierarchical process structures

Top-level Process (orchestration)

Partner Processes

20 January 2010 TTT presentation 25

University of Twente
The Netherlands

WS-BPEL limitations

- ▶ Web service composition is defined as a process per supported operation
→ crosscutting concerns may appear between these operations
- Processes are static and in order to modify them the whole process has to be stopped

20 January 2010 TTT presentation 26

University of Twente
The Netherlands

AOBPEL

- ▶ Approach in which an aspect-oriented variant of BPEL is defined
- ▶ Activities (receive, invoke, pick, reply) are join points
- ▶ Define XML-based notation to define pointcuts (specific activities) and advices
- ▶ Enables the definition and composition of aspects in a BPEL process

20 January 2010 TTT presentation 27

University of Twente
The Netherlands

Aspect-oriented Model-driven approach to message routing in a service bus

- ▶ Model-driven routing in a service bus
- ▶ Based on the concept of 'rich service' to build service buses
- ▶ Each rich service is defined using an MSC
- ▶ Operators to inject aspect behaviour in MSCs
- ▶ Change the routing of messages in the service bus depending of aspects

20 January 2010 TTT presentation 28

University of Twente
The Netherlands

Conclusions

There are many possibilities for applying aspect-oriented techniques to service-oriented architectures (web services)

Web service technology in a sense is already aspect-oriented

There is a lot of potential for aspect-oriented model-driven design of service-oriented architectures (e.g., with web services technologies)

→ requires **behaviour modelling** in order to be meaningful!

20 January 2010 TTT presentation 29