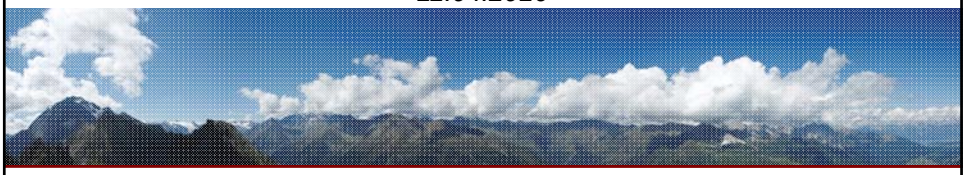
Semantic Web Services
SS 2016

Web services

Anna Fensel

11.04.2016



Where are we?

STI · INNSBRUCK

#	Title
1	Introduction
2	Web Science + TourPack project (separate slideset)
3	Service Science
4	**Web services**
5	Web2.0 services
6	Semantic Web
7	Semantic Web Service Stack (WSMO, WSML, WSMX)
8	OWL-S and the others
9	Semantic Services as a Part of the Future Internet and Big Data Technology
10	Lightweight Annotations
11	Linked Services
12	Applications
13	Mobile Services

Outline

STI · INNSBRUCK

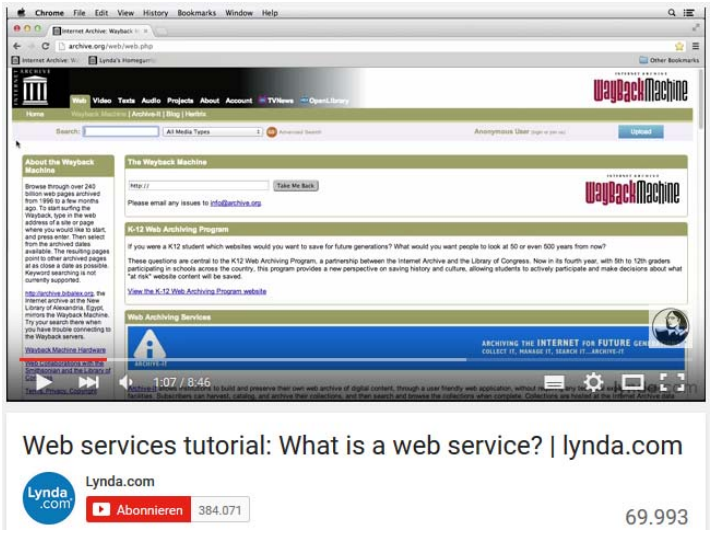
- Motivation
- Technical solution
  - SOAP protocol and related specifications
  - WSDL interface definition language
  - Additional related specifications
- Illustration by a larger example
- Possible Extensions
- Summary
- References

STI · INNSBRUCK

MOTIVATION

Motivation: “What is a Web Service?”

STI · INNSBRUCK



https://www.youtube.com/watch?v=u80uPzhEYvc

5

Motivation
SOA

STI · INNSBRUCK

- Service-Oriented Architecture
- For large-scale distributed computing
- For interoperability within and among enterprises
- Business-oriented services
  - Coarse-grained service interfaces
  - Improves evolvability, manageability
- Needs a suitable set of technologies

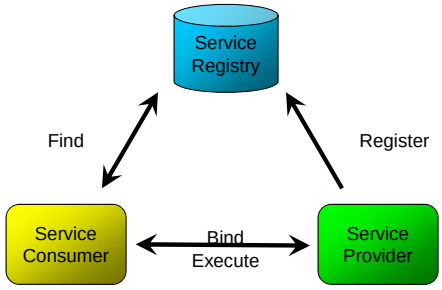
6

Motivation
Solutions needed to implement SOA lifecycle

STI · INNSBRUCK

- SOA is a programming style
- An SOA application is a composition of services
- A 'service' is a building block/unit of an SOA
- Service encapsulates a business process
- A Service Provider registers services
- Service consumption involves Find-Bind-Execute pattern

- Questions
  - How to specify data involved in the process?
  - How to specify the way to access a service functionality?
  - How to communicate to services?
  - Can we use some existing solutions?



SOA Elements

7

Motivation
Web Services

STI · INNSBRUCK

- Technology for SOA
- Client-server messaging approach
  - With predefined *Message exchange patterns*
  - Independent of network protocols
- Application-specific interfaces
  - Similar to RPC, CORBA
- Most data in XML
  - Descriptions in XML Schema

8

Motivation
RPC vs. Web Services

STI · INNSBRUCK

- Remote Procedure Calls
- Aim to hide the network
  - Distributed system looks like a local system
  - Some calls happen to go over the network
- But network is important
  - Latency, bandwidth
  - Reliability – connection, nodes
- RPC is harmful

9

Motivation
Why is RPC considered harmful?

STI · INNSBRUCK

- Introduces semantics of a synchronous, blocking invocation
  - SOA needs a more flexible implementation technology
- Usage of statically typed interfaces
  - Late binding is one of the premises of SOA
- Restrictive, inflexible and does not provide the required level of abstraction
  - Implicit introduction of the behavior which characterizes unified memory space solutions
- Hinders resolution of the interoperability issues
  - Harder to adapt
- An appropriate SOA underlying technology needs to shift from the single-machine-centric vision towards the network-centric vision

10

Motivation
CORBA vs. Web Services

STI · INNSBRUCK

- CORBA = Distributed Objects
  - Encourages fine-grained design
- Much more advanced than RPC

- High barrier to entry
  - Caused by binary protocols, high core complexity
- Initial big interoperability problems
  - Due to the above, and incomplete specifications

11

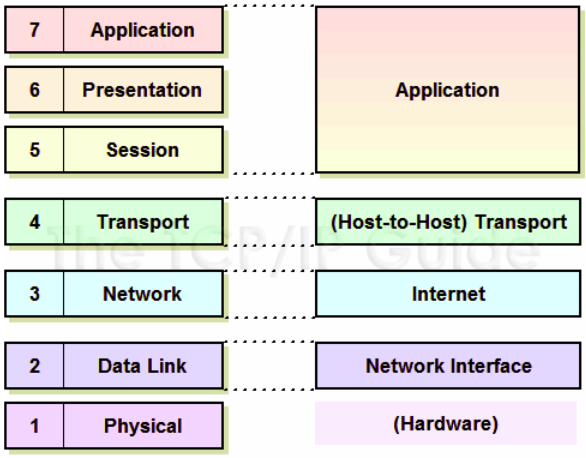
STI · INNSBRUCK

TECHNICAL SOLUTION
WEB SERVICE SPECIFICATION AND PROTOCOL STACK

12

Position in the Communication Architecture: TCP/IP vs. OSI Model

STI · INNSBRUCK

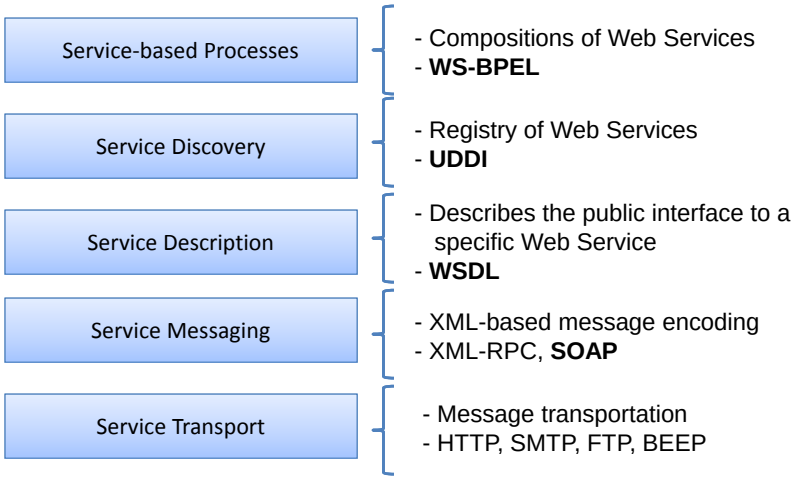


See: http://www.tcpipguide.com/free/t_TCPIPArchitectureandtheTCPIPModel-2.htm

13

Web Service Stack
Overview

STI · INNSBRUCK



Web Service Specification and Protocol Stack

14

SOAP
Overview

STI · INNSBRUCK

- née Simple Object Access Protocol
  - In fact, simple messaging protocol
- Current version: 1.2 @ W3C Recommendation 2007

- Message structure
- Processing model
- Protocol bindings

15

SOAP
Messages

STI · INNSBRUCK

- XML Envelope
- Body
  - Application payload
  - Faults
- Headers
  - Metadata, processing instructions
  - May be marked as mandatory
  - Possibly targeted at *intermediaries*

16

SOAP
Message Example

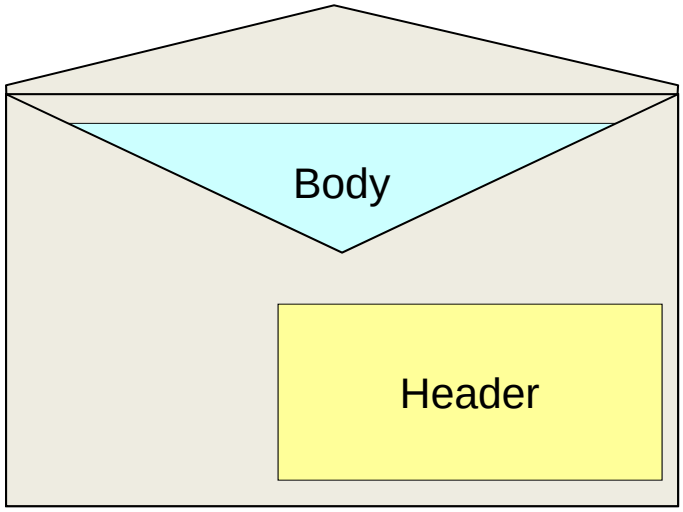
STI · INNSBRUCK

```
<soap:Envelope  namespaces…>
   <soap:Header>
      <wsa:MessageID> unique ID </wsa:MessageID>
      <wsa:ReplyTo> endpoint </wsa:ReplyTo>
   </soap:Header>
   <soap:Body>
      <bank:BalanceRequest>
         data
      </bank:BalanceRequest>
   </soap:Body>
</soap:Envelope>
```

17

SOAP
Message Envelope

STI · INNSBRUCK



18

SOAP
Processing Model

STI · INNSBRUCK

- Processing a message

1. Selecting headers targeted at *me*
   - The current intermediary or ultimate receiver
2. **Checking for understanding**
   - Do I understand all that is targeted at me and marked as *mandatory*?
3. Processing everything in *some* order

19

SOAP
Mandatory Headers

STI · INNSBRUCK

- **mustUnderstand="true"**
  - The recipient must *understand* them
  - Implies agreement to act in accordance to the spec
  - Non-mandatory headers can be ignored
- This mechanism enables gracious evolution
  - If a new feature can be ignored, its introduction won't harm older nodes
  - If a new feature must be understood, its introduction will be discovered early by older nodes, without unexpected behavior

20

SOAP
Protocol Bindings

STI · INNSBRUCK

- Transporting the message over a network
- Addressing
  - What an endpoint address looks like
- Serialization
  - How to put the XML message in on-the-wire bits and bytes
- Connection
  - How to send the bits to the endpoint

21

SOAP
HTTP Binding

STI · INNSBRUCK

- Addressing: URIs
- Serialization: HTTP message body
  - Media type application/soap+xml
- Connection: TCP
- Possibly Web-friendly
  - SOAP 1.1 only used HTTP POST

22

SOAP
MEPs

STI · INNSBRUCK

- Message Exchange Patterns

- Request-Response
  - Input message followed by output or fault
- SOAP-Response
  - Request without SOAP (e.g. HTTP GET)
  - SOAP output or fault

23

SOAP
Summary

STI · INNSBRUCK

- Still pretty simple
- Extremely extensible
- HTTP binding for easy communication

- The value of SOAP:

  XML and Processing Model

24

SOAP
Attachments

STI · INNSBRUCK

- XML is nice, but… not binary
- XOP: XML-binary Optimized Packaging
  - Binary data in XML logically in the tree
  - On serialization, it is outside, raw, efficient
    - MIME multipart message
- MTOM: XOP for SOAP
  - Message Transfer Optimization Mechanism
  - Extends SOAP HTTP binding

25

WS-Addressing

STI · INNSBRUCK

- Simple routing protocol
- Endpoint References for addressing
  - With parameters or metadata in addition to the service address
- Message headers for routing & correlation
  - Many headers use endpoint references

26

WSA Msg Info Headers

STI · INNSBRUCK

- To – let the middleware deliver the message
- From, ReplyTo, FaultTo – channels back
- MessageID, RelatesTo – simple correlation
- Action – semantics implied by the message

```
<S:Header>
    <wsa:MessageID> unique ID </wsa:MessageID>
    <wsa:ReplyTo> endpoint </wsa:ReplyTo>
    <wsa:To> address </wsa:To>
    <wsa:Action> URI </wsa:Action>
</S:Header>
```

27

WSDL
Overview

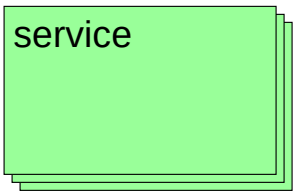
STI · INNSBRUCK

- Web Service Description Language
  - Interface Definition Language (IDL) for Web Services
- Current version: 2.0 @ W3C Recommendation 2007
  - Version 1.1 still in widespread use
- Interface – reusable, abstract
  - Operations with MEPs
- Binding – reusable, concrete
- Service implements an interface
  - Endpoints use bindings

28

WSDL
Component Structure

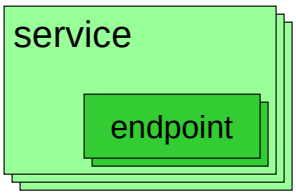
STI · INNSBRUCK



29

WSDL
Component Structure

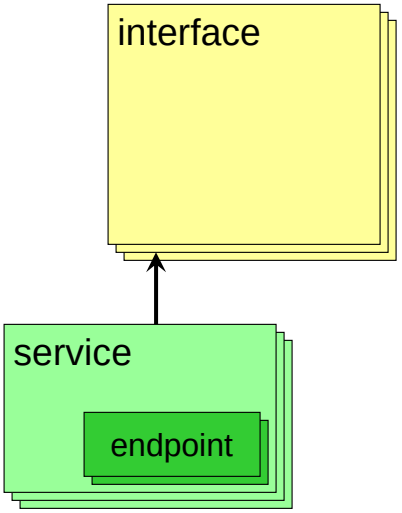
STI · INNSBRUCK



30

WSDL
Component Structure

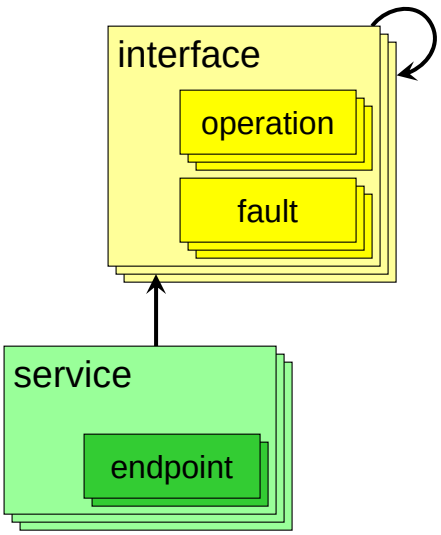
STI · INNSBRUCK



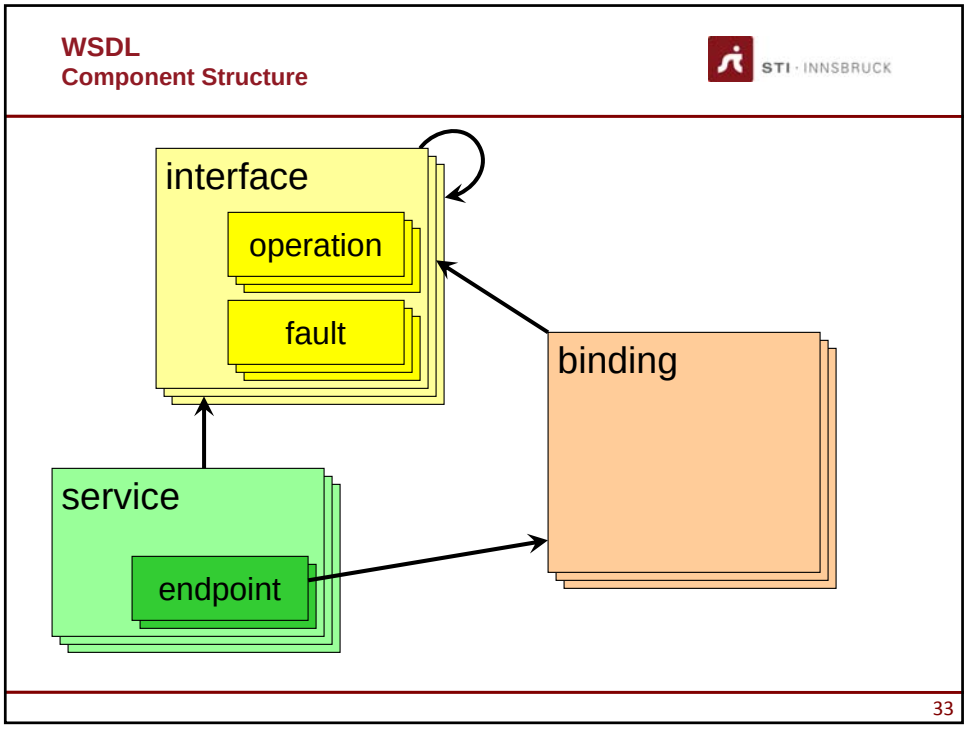
31

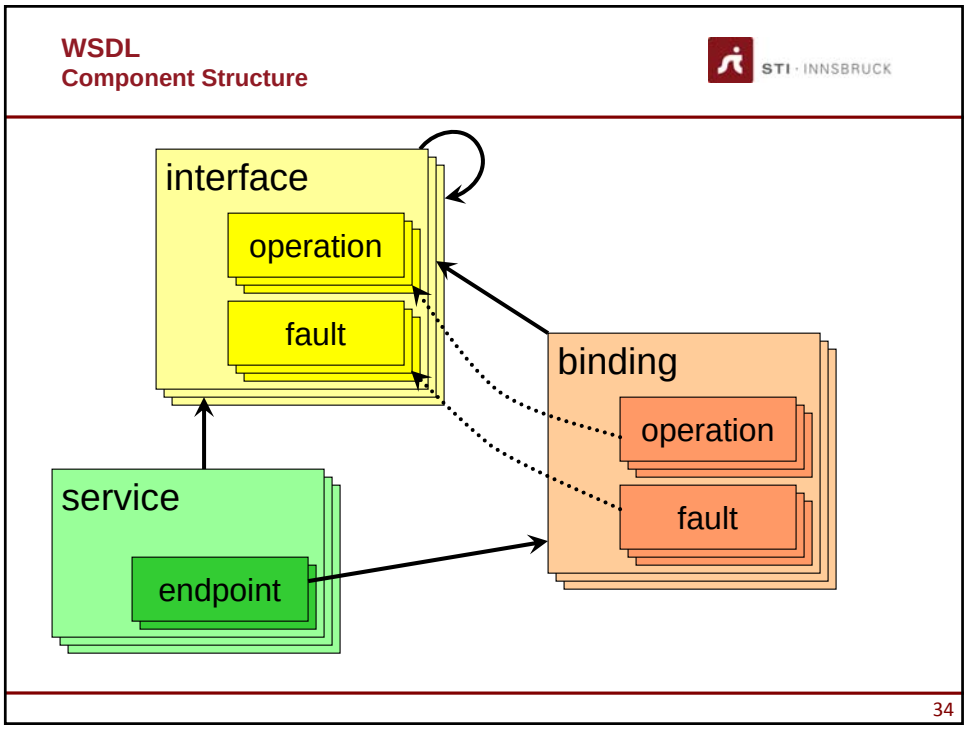
WSDL
Component Structure

STI · INNSBRUCK



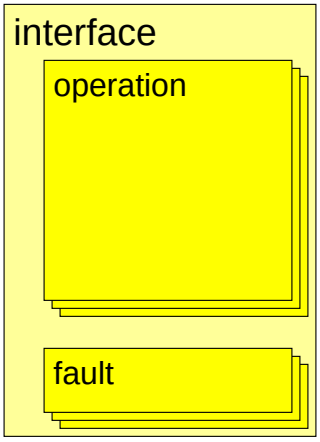
32


WSDL
Component Structure
STI · INNSBRUCK
interface
operation
fault
binding
service
endpoint
33


WSDL
Component Structure
STI · INNSBRUCK
interface
operation
fault
binding
operation
fault
service
endpoint
34

WSDL
Component Structure

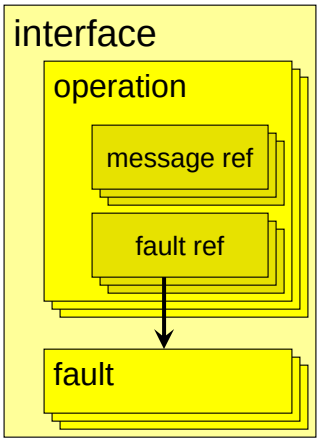
STI · INNSBRUCK



35

WSDL
Component Structure

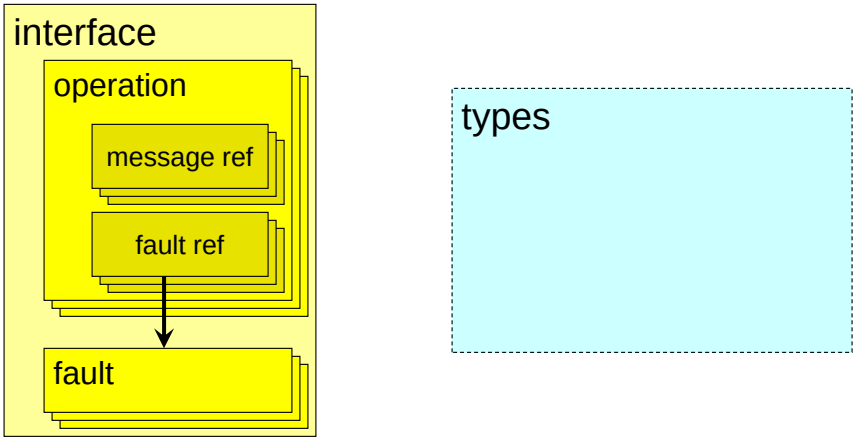
STI · INNSBRUCK



36

WSDL
Component Structure

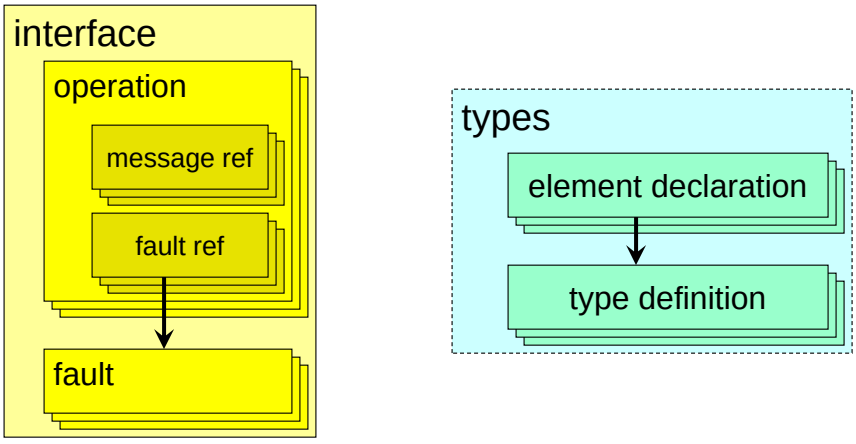
STI · INNSBRUCK



37

WSDL
Component Structure

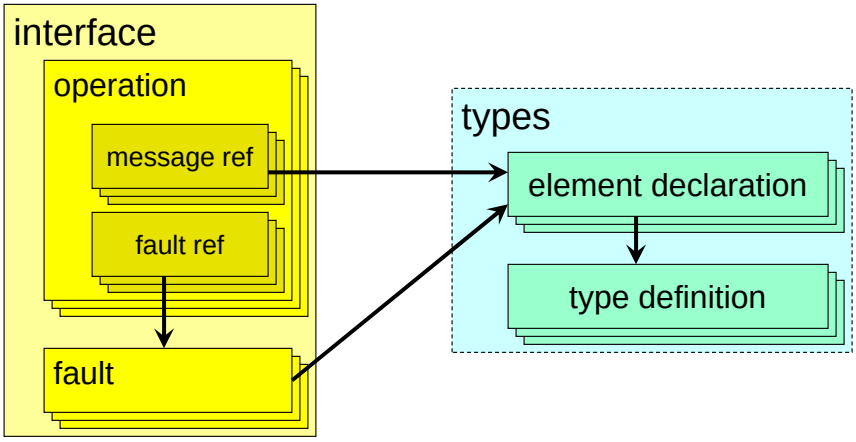
STI · INNSBRUCK



38

WSDL
Component Structure

STI · INNSBRUCK



39

WSDL
Interface

STI · INNSBRUCK

- Design of application interface
  - Possibly *extending* other interfaces
- Operations
  - Message exchange pattern (MEP)
  - Input/output messages, faults
  - Referencing XML elements defined in **types**
- Faults
  - Used and reused by operations

40

WSDL
Interface Example

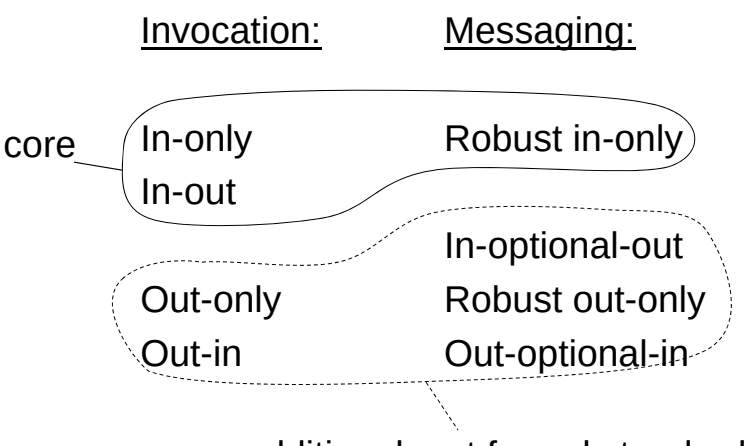
STI · INNSBRUCK

```
<interface name="Banking">
    <operation name="transfer"
            pattern="http://www.w3.org/2004/08/wsdl/in-out">
        <input element="Transfer"/>
        <output element="Balance"/>
        <outfault ref="InvalidBankAccount"/>
        <outfault ref="InsufficientFunds"/>
    </operation>
    <operation name="balance" safe="true"
            pattern="http://www.w3.org/2004/08/wsdl/in-out">
        <input element="BalanceRequest"/>
        <output element="Balance"/>
        <outfault ref="InvalidBankAccount"/>
    </operation>
    <fault name="InvalidBankAccount"
            element="InvalidAccountInfo" />
    <fault name="InsufficientFunds"
            element="InsufficentFundsInfo" />
</interface>
```

41

WSDL
Predefined MEPs

STI · INNSBRUCK



42

WSDL
Invocation MEPs

STI · INNSBRUCK

- In-only: a single input message
  - No faults
- In-out: a single input message, a single output message
  - A fault may replace output message
- Out-only and Out-in mirror images of the above

43

WSDL
Messaging MEPs

STI · INNSBRUCK

- Robust in-only: a single input message
  - May trigger a fault
- In-optional-out: a single input message, possibly an output message
  - Either message may trigger a fault
- Robust out-only, Out-optional-in mirrored

44

WSDL
Bindings

STI · INNSBRUCK

- Networking details necessary for accessing the service
- Copies interface structure
- SOAP and HTTP bindings provided

45

WSDL
Bindings

STI · INNSBRUCK

- SOAP binding:
  - XML message in SOAP envelope
  - Transport using a SOAP protocol binding (HTTP)
- HTTP binding:
  - Web-friendly
  - XML message in payload, or as parameters in the URI

46

WSDL
Service

STI · INNSBRUCK

- A logical node of the application
- One interface
- Multiple alternate endpoints
- Endpoints may have different bindings
  - E.g. SOAP over HTTP for a public endpoint, and SOAP over JMS for the intranet

47

WSDL
Service Example

STI · INNSBRUCK

```
<service          name="HypoTirol"
                  interface="Banking">
    <endpoint     name="visible"
                  binding="HTTP"
                  address="http://ht.at/bankingSvc" />
    <endpoint     name="tls"
                  binding="SecureHTTP"
                  address="https://ht.at/bankingSvc" />
</service>
```

48

WSDL Summary

STI · INNSBRUCK

- Concrete service
- Abstract and reusable Interface
- Network binding

- WSDL does not imply implementation
  - CORBA IDL requires objects
- Exchange of XML business documents
- Extensible in many ways

49

UDDI

STI · INNSBRUCK

- Universal Description, Discovery and Integration
- A Web service registry API specification
  - Business-oriented service publication, discovery (initially limited search capabilities)
  - Itself has Web service interface
  - Useful for intranet registries
- A failed public service

50

UDDI
Structure

STI · INNSBRUCK

- Business entity
  - Organization information, contact
- Business Service
  - A group of related services
- Binding Template
  - Information on how to access the service
- tModel (technical model)
  - Any kind of specification, e.g. WSDL
  - Also for classification, categorization

51

WS-Policy
Overview

STI · INNSBRUCK

- Non of the previous solutions is addressing the non-functional aspects in the context of Web Services
  - e.g., a Web Service can be accessed only if a particular security constrains (like secure channel communication) hold
- WS-Policy is an XML-based set of specifications to advertise and specify service-related non-functional metadata
  - Security, Quality of service, Privacy, Transactional policies
- WS-Policy main specifications
  - WS-Policy Framework
    - Provides model and syntax to describe and communicate policies
  - WS-Policy Assertions
    - Defines a common set of policy assertions for Web services
  - WS-Policy Attachments
    - General-purpose mechanisms for associating such policies with the subjects to which they apply
- Capability vs. requirement policy

52

WS-Policy
A Policy

STI · INNSBRUCK

- Policy is a collection of policy alternatives (OR)
  - Auth tokens: Kerberos OR X509
- Policy alternative is a collection of policy assertions (AND)
  - Auth token AND secure channel
- Policy assertions specify
  - traditional policies that will ultimately manifest on the wire (authentication scheme, transport protocol selection), and
  - critical to proper service selection and usage (privacy policy, QoS characteristics).
- Two "operators" (XML tags) are used to make statements about policy combinations:
  - *wsp:ExactlyOne* - asserts that only one child node must be satisfied.
  - *wsp:All* - asserts that all child nodes must be satisfied.
- The intersection is a new policy that complies with both their requirements/capabilities

53

WS-Policy
An Example of the Security Policy

STI · INNSBRUCK

```
<wsp:Policy>
  <wsp:ExactlyOne>
    <wsse:SecurityToken>
      <wsse:TokenType>wsse:Kerberosv5TGT
      </wsse:TokenType>
    </wsse:SecurityToken>
    <wsse:SecurityToken>
      <wsse:TokenType>wsse:X509v3
      </wsse:TokenType>
    </wsse:SecurityToken>
  </wsp:ExactlyOne>
</wsp:Policy>
```

54

WS-BPEL
Overview

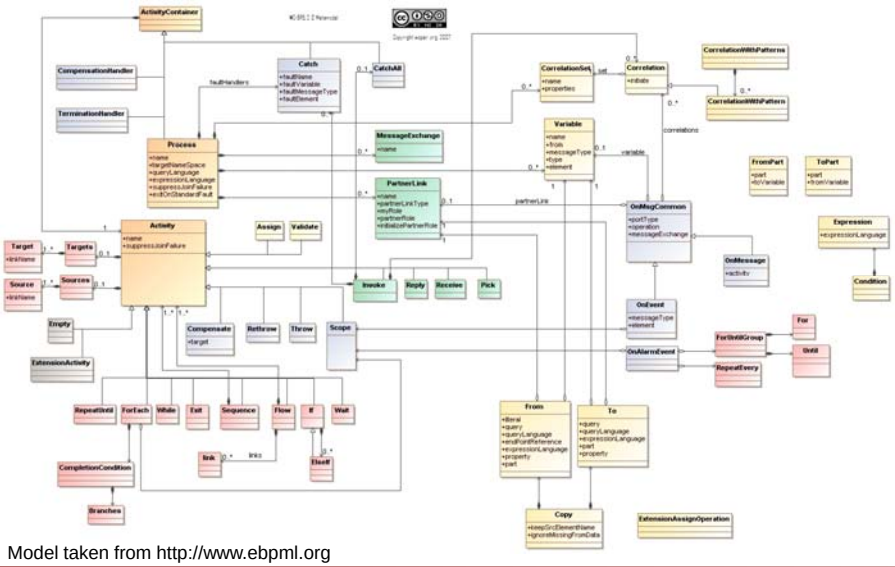
STI · INNSBRUCK

- WS-BPEL is an orchestration executable language for specifying interactions with Web Service
  - Defines an interoperable integration model for Web Service-based processes.
  - Describes high-level state transition interactions of a process.
- The language supports following basic facilities:
  - Message exchanges,
  - Property-based message correlation mechanism,
  - XML and WSDL typed variables,
  - Structured programming-language concepts such as if-then-else, while, sequence, and flow,
  - Scoping system to allow encapsulation of logic with local-variables, fault-, compensation- and event- handlers,
  - Serialized scopes to control concurrent access to variables,
  - Language plug-in model which allows expression writing in the language such as XPath

55

WS-BPEL
WS-BPEL v2.0 MetaModel

STI · INNSBRUCK



Model taken from http://www.ebpml.org

56

WS-BPEL
Main elements

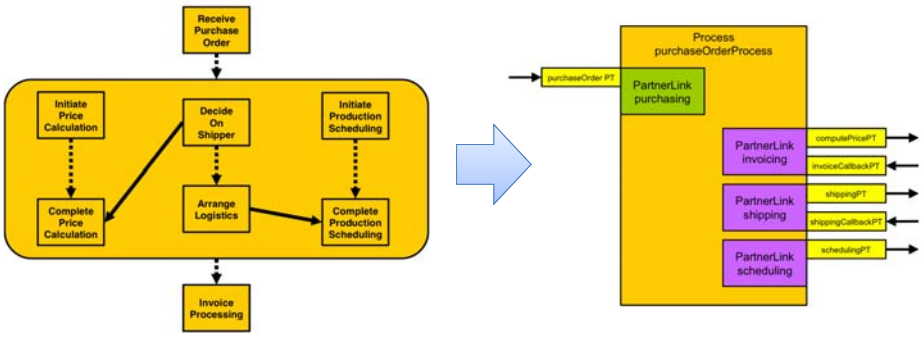
STI · INNSBRUCK

- Process
  - Root element for a WS-BPEL process specification
- Variable
  - Holding messages that constitute a part of the state of a business process.
- Activity
  - Perform the process logic.
  - Basic vs. structured activities.
- PartnerLink
  - Models peer-to-peer conversational partner relationships.
- Catch
  - Scope-based custom fault-handling activities.
- Correlation
  - Used to differentiate instances of the business processes.
  - The declaration of correlation relies on declarative properties of messages.
- onEvent
  - Scope-based concurrent answering to events
  - onEvent vs, onAlarm

57

WS-BPEL
An Example

STI · INNSBRUCK



Example figures and WSBPEL specification taken from *OASIS standard Web Services Business Process Execution Language Version 2.0, 11 April 2007*

58

WS-BPEL
An Example

STI · INNSBRUCK

```
<process name="purchaseOrderProcess" targetNamespace=http://example.com/ws-bp/purchase
         xmlns=http://docs.oasis-open.org/wsbpel/2.0/process/executable
         xmlns:lns="http://manufacturing.org/wsdl/purchase">

  <partnerLinks>
    <partnerLink name="purchasing" partnerLinkType="lns:invoicingLT" myRole="invoiceRequester"
     partnerRole="invoiceService" />
    <partnerLink name="shipping" partnerLinkType="lns:shippingLT" myRole="shippingRequester"
     partnerRole="shippingService" />
    <partnerLink name="scheduling" partnerLinkType="lns:schedulingLT" partnerRole="schedulingService" />
  </partnerLinks>

  <variables>
    <variable name="PO" messageType="lns:POMessage" />
    <variable name="Invoice" messageType="lns:InvMessage" />
    <variable name="shippingRequest"messageType="lns:shippingRequestMessage" />
    <variable name="shippingInfo"messageType="lns:shippingInfoMessage" />
    <variable name="shippingSchedule"messageType="lns:scheduleMessage" />
  </variables>

  <faultHandlers>
    <catch faultName="lns:cannotCompleteOrder"faultVariable="POFault"
     faultMessageType="lns:orderFaultType">
      <reply partnerLink="purchasing"portType="lns:purchaseOrderPT" operation="sendPurchaseOrder"
     variable="POFault" faultName="cannotCompleteOrder" />
    </catch>
  </faultHandlers>
  …
```

59

WS-BPEL
An Example

STI · INNSBRUCK

```
…
<sequence>
  <receive partnerLink="purchasing" portType="lns:purchaseOrderPT" operation="sendPurchaseOrder"
           variable="PO" createInstance="yes">
  </receive>
  <flow>
    <links>
      <link name="ship-to-invoice" />
      <link name="ship-to-scheduling" />
    </links>
    <sequence>
      <assign>
        <copy>
          <from>$PO.customerInfo</from>
          <to>$shippingRequest.customerInfo</to>
        </copy>
      </assign>
      <invoke partnerLink="shipping" portType="lns:shippingPT" operation="requestShipping"
              inputVariable="shippingRequest" outputVariable="shippingInfo">
        <sources>
          <source linkName="ship-to-invoice" /> </sources>
      </invoke>
      <receive partnerLink="shipping" portType="lns:shippingCallbackPT" operation="sendSchedule"
               variable="shippingSchedule">
        <sources>
          <source linkName="ship-to-scheduling" />
        </sources>
      </receive>
    </sequence>
    …
```

60

WS-BPEL
An Example

STI · INNSBRUCK

```
…
<sequence>
  <invoke partnerLink="invoicing" portType="lns:computePricePT"
          operation="initiatePriceCalculation" inputVariable="PO">
  </invoke>
  <invoke partnerLink="invoicing" portType="lns:computePricePT" operation="sendShippingPrice"
          inputVariable="shippingInfo">
    <targets><target linkName="ship-to-invoice" /></targets>
  </invoke>
  <receive partnerLink="invoicing" portType="lns:invoiceCallbackPT" operation="sendInvoice"
           variable="Invoice" />
</sequence>
<sequence>
  <invoke partnerLink="scheduling" portType="lns:schedulingPT"
          operation="requestProductionScheduling" inputVariable="PO">
  </invoke>
  <invoke partnerLink="scheduling" portType="lns:schedulingPT" operation="sendShippingSchedule"
          inputVariable="shippingSchedule">
    <targets><target linkName="ship-to-scheduling" /></targets>
  </invoke>
</sequence>
</flow>
<reply partnerLink="purchasing" portType="lns:purchaseOrderPT" operation="sendPurchaseOrder"
       variable="Invoice">
</reply>
</sequence>
</process>
```


The Rest of the Specs

STI · INNSBRUCK

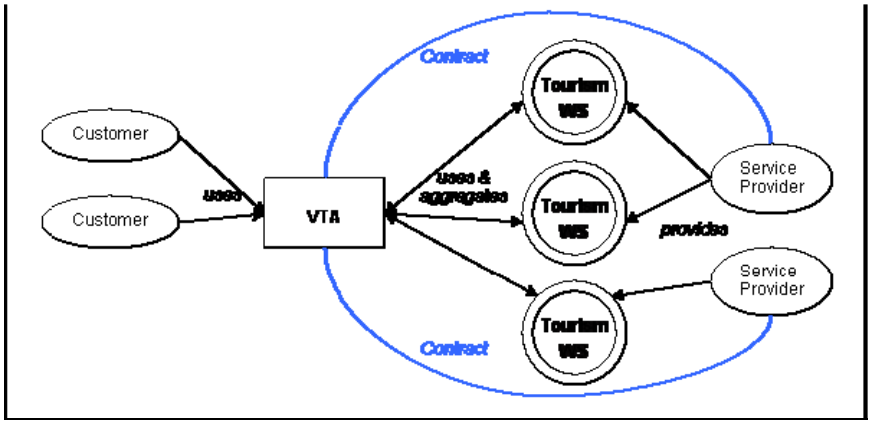
- Security: authentication, encryption
- Reliable messaging
  - "Get the message there or tell me"
- Transactions: ACID, Compensating
- WS-BPEL: Scripting Web Services
- Interoperability: WS-I Basic Profile
- … and still more

STI · INNSBRUCK

ILLUSTRATION BY A LARGER EXAMPLE

Virtual Travel Agency

STI · INNSBRUCK



Blue Hotel WSDL (1)

STI · INNSBRUCK

```
<?xml version="1.0" encoding="utf-8" ?>
<description
  xmlns="http://www.w3.org/ns/wsdl"
  targetNamespace=
    "http://www.bluehotel.com/wsdl/BlueHotelService"
  xmlns:tns=
    "http://www.bluehotel.com/wsdl/BlueHotelService"
  xmlns:bhns=
    "http://www.bluehotel.com/schemas/BlueHotelService"
  xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsdlx= "http://www.w3.org/ns/wsdl-extensions">

  <documentation>
    This document describes the Blue Hotel Web service.
  </documentation>
```

65

Blue Hotel WSDL (2)

STI · INNSBRUCK

```
<types>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.bluehotel.com/schemas/BlueHotelService"
      xmlns="http://www.bluehotel.com/schemas/BlueHotelService">
    <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="tCheckAvailabilityResponse"/>
    <xs:complexType name="tCheckAvailabilityResponse">
     <xs:sequence>
       <xs:element  name="roomType" type="xs:string"/>
       <xs:element  name="rateType" type="xs:string"/>
       <xs:element  name="rate" type="xs:double"/>
     </xs:sequence>
    </xs:complexType>
    <xs:element name="invalidDataError" type="xs:string"/>
  </xs:schema>
</types>
```

66

Blue Hotel WSDL (3)

STI · INNSBRUCK

```
<interface name = "BlueServiceInterface" >

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

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

</interface>
```

67

Blue Hotel WSDL (4)

STI · INNSBRUCK

```
 <binding name="BlueServiceSOAPBinding"
     interface="tns:BlueServiceInterface"
     wsoap:protocol=
          "http://www.w3.org/2003/05/soap/bindings/HTTP/"
     type="http://www.w3.org/ns/wsdl/soap">
   <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="BlueService"
          interface="tns:BlueServiceInterface">
     <endpoint name="reservationEndpoint"
          binding="tns:BlueServiceSOAPBinding"
          address="http://www.bluehotel.com/BlueService"/>
 </service>
</description>
```

68

Blue Hotel UDDI Binding Template

STI · INNSBRUCK

```
<bindingTemplate
    bindingKey=
        "uuid:36F1B765-BDB3-4837-828D-8284301E5A2A"
    serviceKey=
        "uuid:40E6D5A8-3E16-4f01-99DA-035229685A40">
  <description xml:lang="en">
        SOAP binding for Blue Hotel service
  </description>
  <accessPoint URLType="http">
        http://www.bluehotel.com/hotel:80/soap
  </accessPoint>
  <tModelInstanceDetails>
    <tModelInstanceInfo tModelKey=
        "uuid:AE1B645F-CF2F-491f-811A-4868705F5904"/>
  </tModelInstanceDetails>
</bindingTemplate>
```

69

Blue Hotel UDDI tModel

STI · INNSBRUCK

```
<tModel tModelKey=
         "uddi:WE1B6Q5F-CF2F-491f-811A-4868705F5904"
        operator="http://www.bluehotel.com/hotel"
        authorizedName="George Blue">
  <name>BlueHotelInterface Port Type</name>
  <description>
         An interface for the Blue Hotel service
  </description>
  <overviewDoc><overviewURL>
          http://www.bluehotel.com/services/
          BlueHotelService.wsdl
  </overviewURL></overviewDoc>
</tModel>
```

70

Blue Hotel SOAP Request

STI · INNSBRUCK

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope soap:encodingStyle=
               "http://www.w3.org/2001/12/soap-encoding"
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
  <soap:Body xmlns:bhns=
         "http://www.bluehotel.com/wsdl/BlueHotelService">
    <bhns:checkAvailability>
      <bhns:checkInDate>2009-03-24</bhns:checkInDate>
      <bhns:checkOutDate>2009-03-30</bhns:checkOutDate>
      <bhns:roomType>Single</bhns:roomType>
    </bhns:checkAvailability>
  </soap:Body>
</soap:Envelope>
```

71

Blue Hotel SOAP Response

STI · INNSBRUCK

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope soap:encodingStyle=
               "http://www.w3.org/2001/12/soap-encoding"
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
  <soap:Body xmlns:bhns=
         "http://www.bluehotel.com/wsdl/BlueHotelService">
    <bhns:checkAvailabilityResponse>
      <bhns:roomType>Single</bhns:roomType>
      <bhns:rateType>Discount</bhns:rateType>
      <bhns:rate>150.50</bhns:rate>
    </bhns:checkAvailabilityResponse>
  </soap:Body>
</soap:Envelope>
```

72

STI · INNSBRUCK

SUMMARY

Web Services Summary

STI · INNSBRUCK

- Technology for service-oriented systems
- Core: protocol + IDL (SOAP + WSDL)
- Many extension specifications
- Messaging paradigm
  - Sometimes RPC
- Biggest benefit: XML
  - Vendor-neutral, platform-independent
  - Interoperable
  - Easy to begin and play with
  - But hidden in the frameworks
    - Unless you're a developer of the framework

Possible Extensions

STI · INNSBRUCK

- Semantic Web Services, Linked Services
  - Automating the use of Web services
  - Using semantic technologies
- RESTful services
  - Real **Web** services
  - Integrated with the Web
- Microservices
  - Small services
  - Domain-driven design and virtualisation

75

References

STI · INNSBRUCK

- Mandatory reading:
  - G. Alonso, F. Casati, H. Kuno, V. Machiraju, 2004. *Web Services, Concepts Architectures and Applications*. Springer Verlag ISBN 3-540-44008-9.
  - WSDL http://w3.org/TR/wsdl20
  - SOAP http://w3.org/TR/soap12
  - UDDI http://uddi.xml.org/

76

References

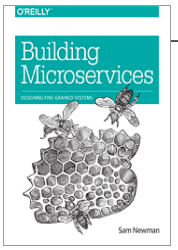
STI · INNSBRUCK

- Further reading:
  - O'Reilly books:

2002

2007

2015



  - XOP http://w3.org/TR/xop10
  - MTOM http://w3.org/TR/soap12-mtom
  - WS-Addressing http://w3.org/TR/ws-addr-core
  - WS-Policy http://w3.org/TR/ws-policy
  - WS-BPEL http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
  - OASIS http://oasis-open.org/
  - W3C http://w3.org/

77

References

STI · INNSBRUCK

- Wikipedia links:
  - http://en.wikipedia.org/wiki/Web_Services
  - http://en.wikipedia.org/wiki/Web_Services_Description_Language
  - http://en.wikipedia.org/wiki/SOAP
  - http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration
  - http://en.wikipedia.org/wiki/Business_Process_Execution_Language
  - http://en.wikipedia.org/wiki/WS-Addressing
  - http://en.wikipedia.org/wiki/XML-binary_Optimized_Packaging
  - http://en.wikipedia.org/wiki/WS-Policy
  - http://en.wikipedia.org/wiki/Web_services_protocol_stack
- YouTube links (SOAP, WSDL, etc. implementation in Eclipse):
  - https://www.youtube.com/watch?v=mKjvKPlb1rA&list=PLqq-6Pq4lTTZTYpk_1DOowOGWJMIH5T39 (SOAP Web Service Basics at Java Brains channel)

78

Next Lecture

STI · INNSBRUCK

#	Title
1	Introduction
2	Web Science + TourPack project (separate slideset)
3	Service Science
4	Web services
5	**Web2.0 services**
6	Semantic Web
7	Semantic Web Service Stack (WSMO, WSML, WSMX)
8	OWL-S and the others
9	Semantic Services as a Part of the Future Internet and Big Data Technology
10	Lightweight Annotations
11	Linked Services
12	Applications
13	Mobile Services

79

Questions?

STI · INNSBRUCK

80