VSR | EDU

Software Service Engineering

Software Service Engineering

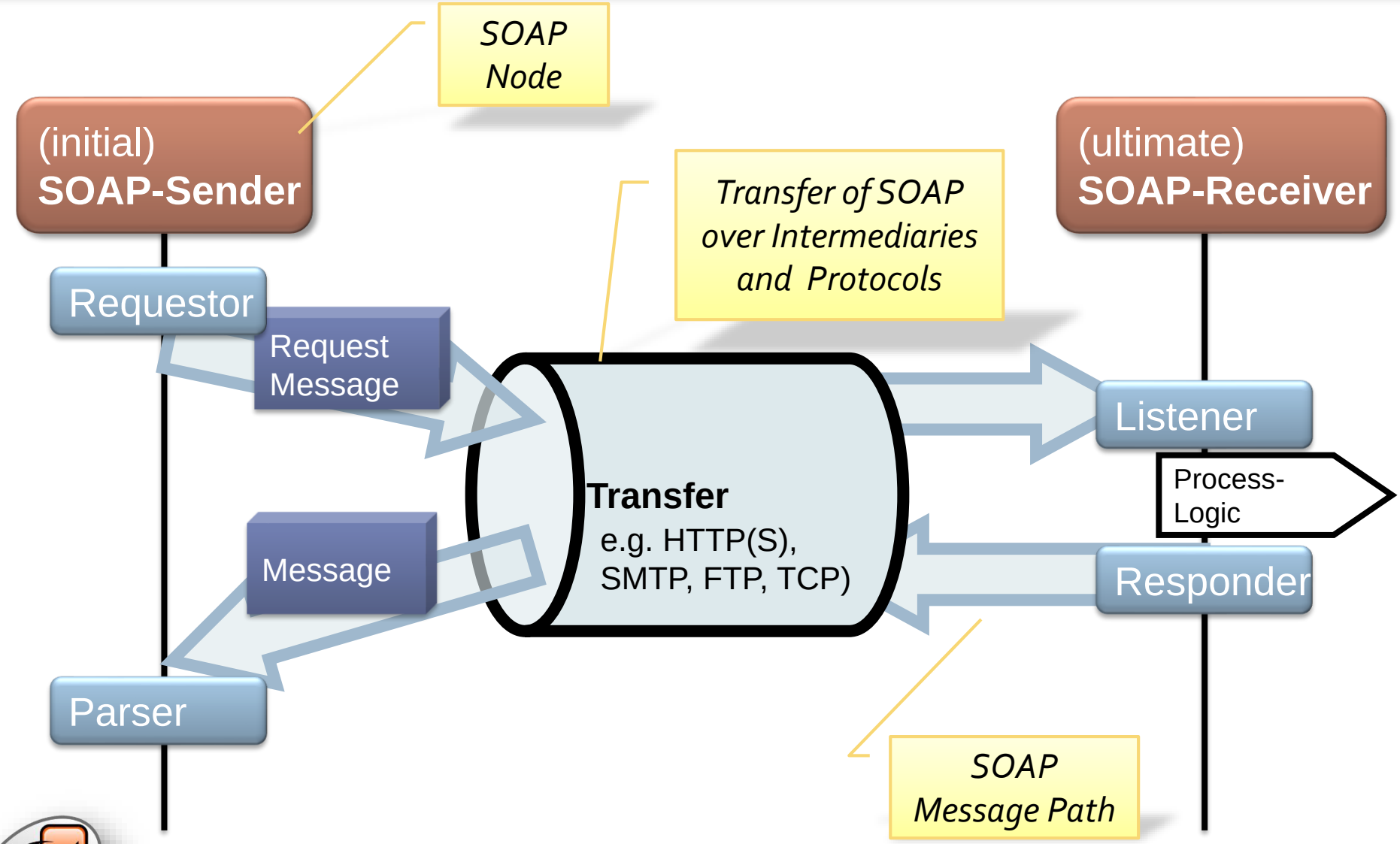
Prof. Dr.-Ing. Martin Gaedke
Technische Universität Chemnitz
Fakultät für Informatik
Professur Verteilte und selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

SOAP in Action – How?

- To work with SOAP one requires:
  - A client, which constructs and sends SOAP requests
  - A server, which understands SOAP requests, implements/executes the required functionality, creates a response in the form of a SOAP response and sends it back to the Client
  - Possibly, SOAP intermediaries, which route the messages
- This scenario is independent of the underlying technology, programming languages, platform
  - It is ultimately realized by a Web Application

SOAP in Action



Message Exchange Patterns

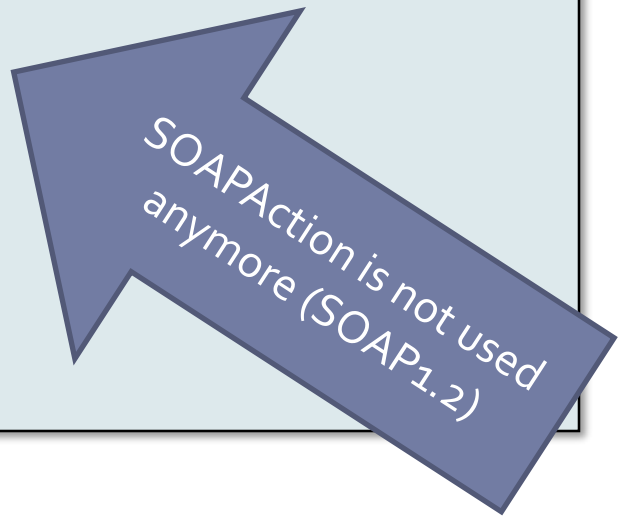
- Template, devoid of application semantics, that describes a generic pattern for the exchange of messages between agents.
- Describes the relationships (e.g., temporal, causal, sequential, etc.) of multiple messages exchanged in conformance with the pattern, as well as the normal and abnormal termination of any message exchange conforming to the pattern.
- SOAP patterns (more patterns are defined in WSDL spec):
  - Request-Response Message Exchange Pattern
  - Response Message Exchange Pattern

SOAP with HTTP Binding

- SOAP-Request via HTTP-POST-Request

```
POST /WebCalculator/Calculator.asmx HTTP/1.1
Content-Type: text/xml
...
SOAPAction: "http://tempuri.org/Add"
Content-Length: 386

<?xml version="1.0"?>
<soap:Envelope ...>
  ...
</soap:Envelope>
```



SOAP-Envelope Example (1)

- SOAP-Schema XML document

```
<?xml version="1.0"?>
<soap:Envelope ...>
  <soap:Header ...>
      <hb1:headerblock1 soap:mustUnderstand/>
      <hb2:headerblock2/>...
  </soap:Header>
  <soap:Body>
    <MyQuery xmlns="http://tempuri.org/">
      <n1>12</n1>
      <n2>10</n2>
    </MyQuery >
  </soap:Body>
</soap:Envelope>
```

SOAP-Envelope Example (2)

- Data structures serialized in XML

```
<soap:Envelope ...>
 <soap:Body>
  <MyQueryResult xmlns="http://tempuri.org/">
   <result>
    <Description>Plastic Novelties Ltd</Description>
    <Price>129</Price>
    <Ticker>PLAS</Ticker>
   </result>
  </MyQueryResult>
 </soap:Body>
</soap:Envelope>
```

SOAP Example (1)

- SOAP-Request through HTTP-POST

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn

<env:Envelope
   xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <m:symbol>DIS</m:symbol>
       </m:GetLastTradePrice>
   </env:Body>
</env:Envelope>
```

SOAP Example (2.1)

- SOAP-Response over HTTP

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<soap:Envelope
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <m:GetLastTradePriceResponse xmlns:m="Some-URI">
        <Price>34.5</Price>
      </m:GetLastTradePriceResponse>
   </soap:Body>
</soap:Envelope>
```

SOAP Example (2.2)

- SOAP-Fault (error message) over HTTP

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<soap:Envelope
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Body>
      <soap:Fault>
         <faultcode>SOAP: MustUnderstand</faultcode>
         <faultstring>SOAP Must Under Error</faultstring>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

SOAP - Final Considerations

- SOAP security
  - End-to-end security – what does it mean in SOAP context?
  - HTTPS can be used in the HTTP context, does it enable end-to-end security?
- The strength of SOAP lies in its extensibility (similarly to HTTP)
  - Which leads to:
    - a variety of protocols that build up on SOAP
    - a lot of extensions, which systematically expand SOAP's capabilities
    - a variety of further bindings that enable using other protocols to transfer SOAP messages
- SOAP (as well as HTTP) is one of the pillars of Web Services

SOAP - Demo

http://vsr-demo.informatik.tu-chemnitz.de/webservices/SoapWebService/Service.asmx

Service

Click here for a complete list of operations.

Add

Test

The test form is only available for requests from the local machine.

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced

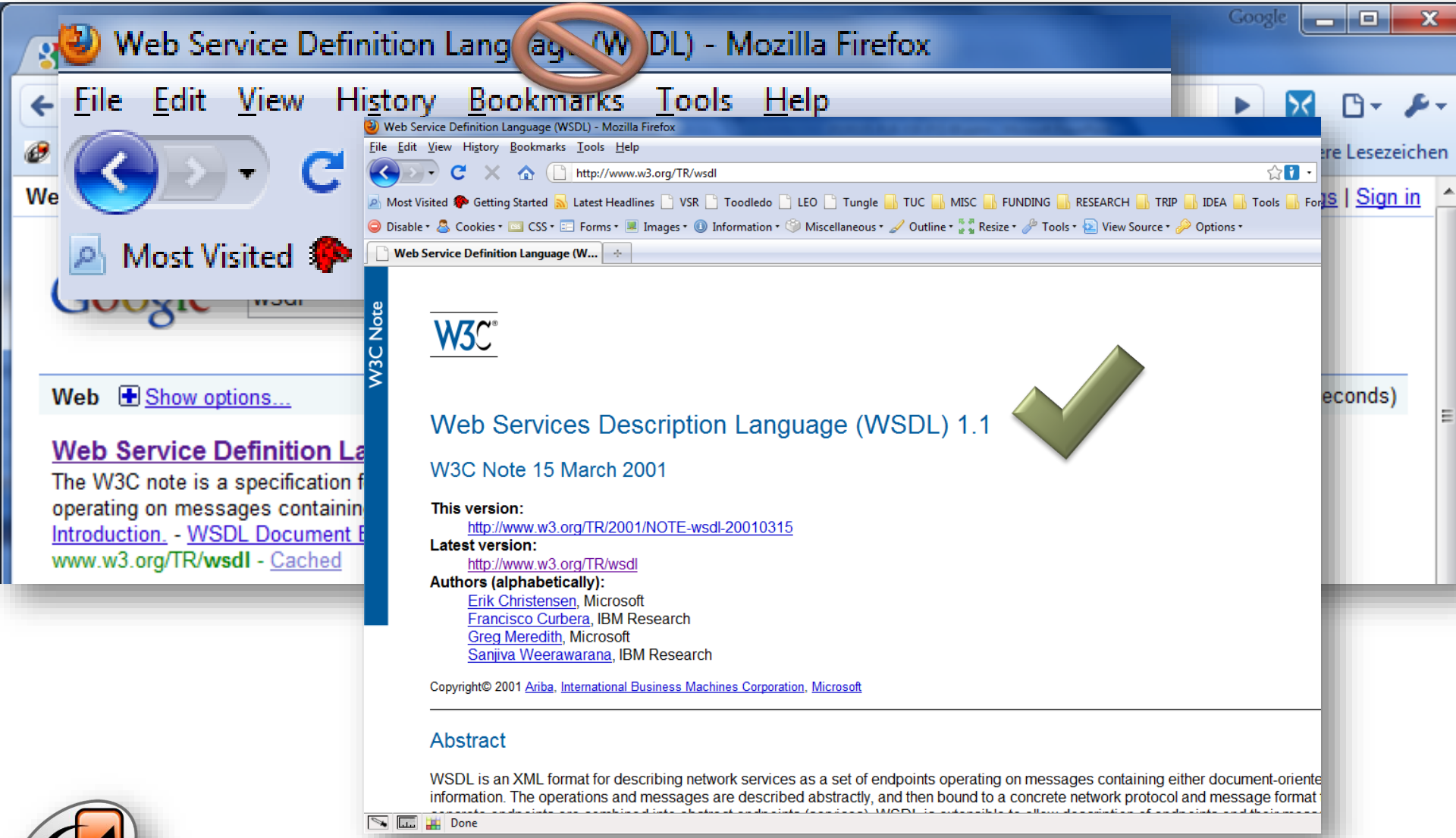
```
POST /SoapWebService/Service.asmx HTTP/1.1
Host: pauline.informatik.tu-chemnitz.de
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://vsr.informatik.tu-chemnitz.de/edu/2008/pvs/soapwebservice/Add"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.
  <soap:Body>
    <Add xmlns="http://vsr.informatik.tu-chemnitz.de/edu/2008/pvs/soapwebservice">
      <a>int</a>
      <b>int</b>
    </Add>
  </soap:Body>
</soap:Envelope>
```

Chapter 2

WEB SERVICES DESCRIPTION LANGUAGE (WSDL)

WSDL stands for...



WSDL – Introduction

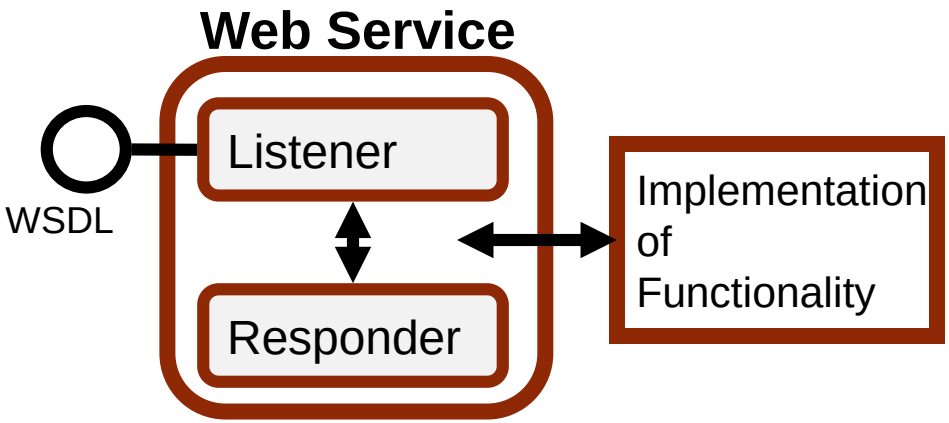
- Web Services Description Language (WSDL)
  - Use of XML to describe functionality as a set of endpoints
  - Similarly to an IDL: methods, arguments and return values are described
  - Unlike an IDL, message encoding and exchange methods are described as well
- History:
  - WSDL 1.1: W3C Note 15 March 2001
  - http://www.w3.org/TR/wsdl
  - Independent development efforts of IBM and Microsoft
  - Current: Web Services Description Language (WSDL) 2.0 – W3C Recommendation 26 June 2007

Describing Web Services

- Web Services Description Language 1.1
  - Acronym: WSDL
  - Status: W3C Note 15 March 2001
  - http://www.w3.org/TR/wsdl
  - Independent efforts from IBM and Microsoft
- WSDL is for describing Web Services
  - Defines XML-based grammar for describing network services as a set of endpoints
  - Describes their methods, arguments, return values and how to use

Web Services Architecture

- WSDL: Core element of the Web Service Architecture stack (Endpoint definition language)



Simplified Web Service Stack (WS-I Basic Profile 1.0 compliant)

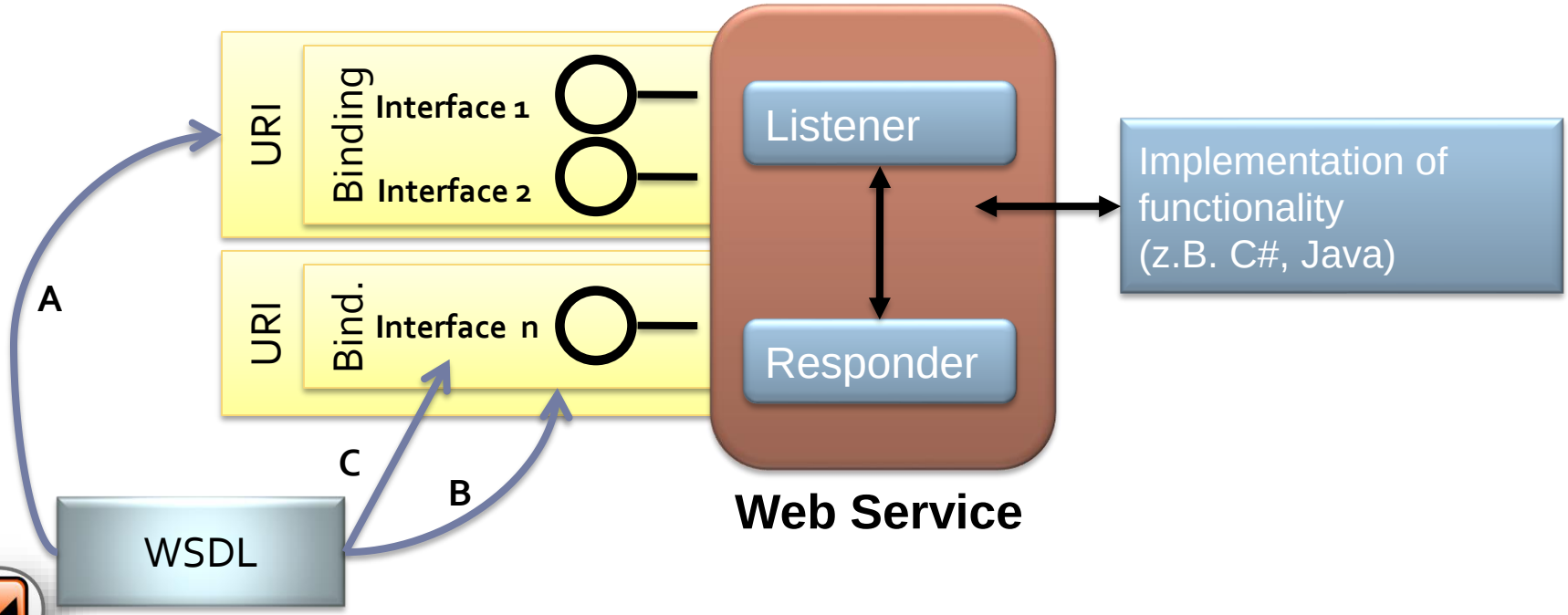
UDDI (service discovery)
WSDL (service description)
XSD (service description)
SOAP (messaging)
XML 1.0 + Namespaces (messaging)

WSDL Goals

- Extensibility wrt.
  - New Transport protocols
  - New Encoding rules
- Abstraction wrt.
  - Endpoints and Messages
  - THEN mapped onto **n** concrete transports and encodings
- Reuse wrt.
  - Definitions – reuseable to create new definitions

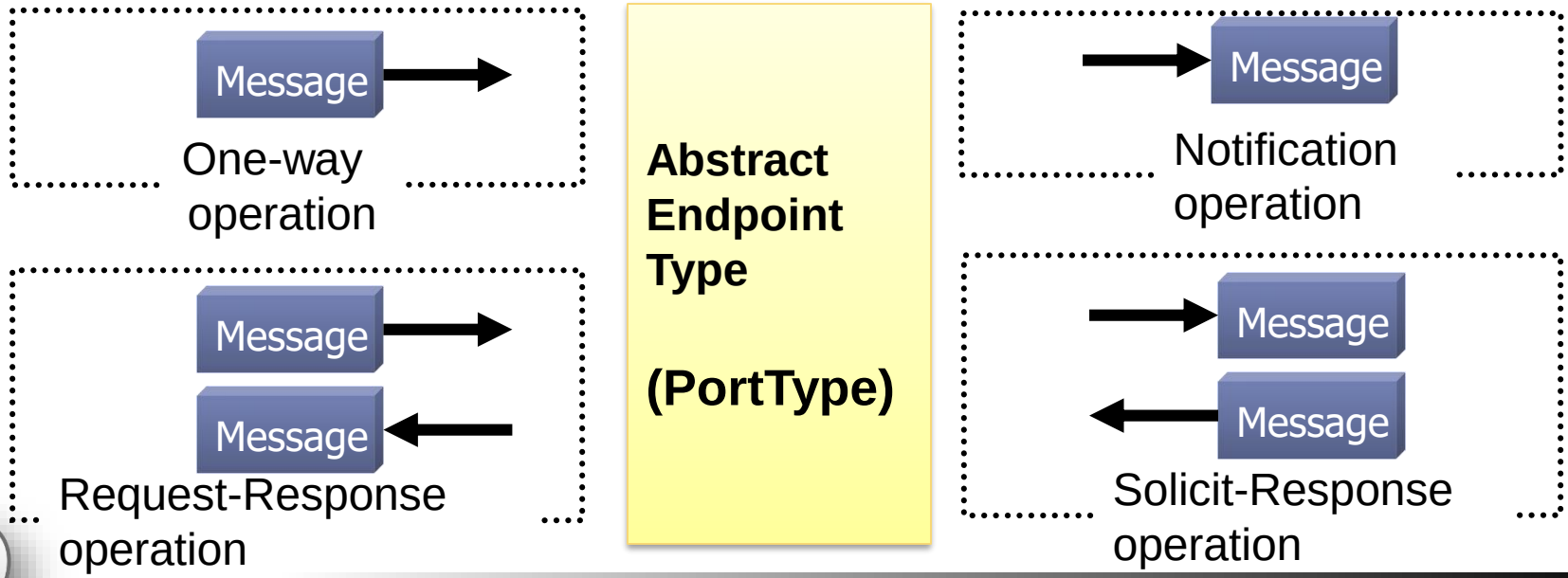
Web Service - Description

- WSDL is the central element of Web Services Technologies in WSA
  - Technology-independent XML description of endpoints (ABC)



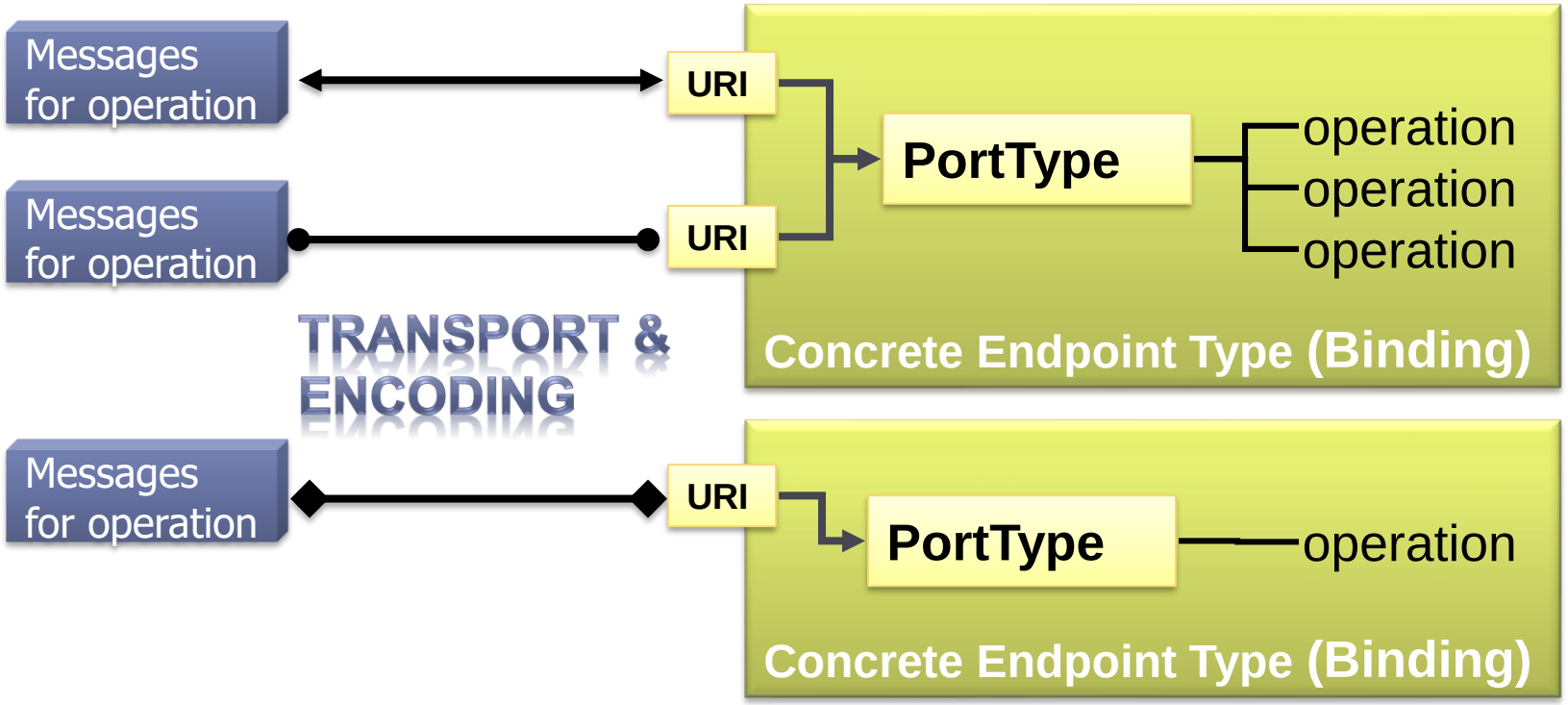
WSDL: Abstract Endpoint Type

- **Abstract Endpoint Type** (PortType / Contract) – (Possible) Part of WSDL specification
  - Describes: Message types, operations and parameters
  - Set of message flows (operations), which are expected from a dedicated endpoint type (independent of technology in use)



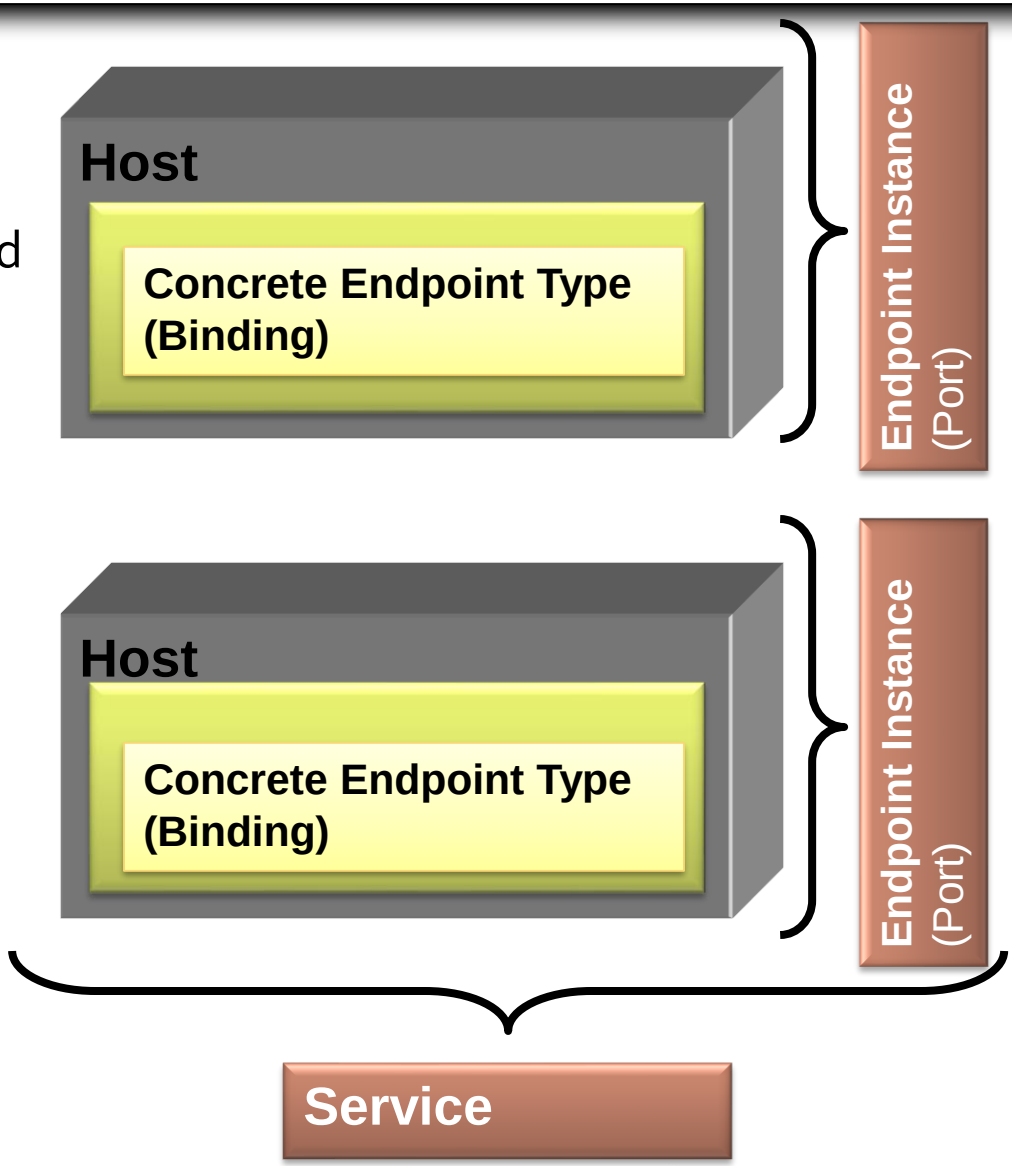
WSDL: Concrete Endpoint Type

- **Concrete Endpoint Type** (Binding) – Defines transport and encoding particulars for a portType



WSDL: Service Definition

- **Port** (Endpoint Instance)
  - Addresses of an endpoint and the according binding (address is a URI)
  - Attention: Do not confuse Port with a TCP port!

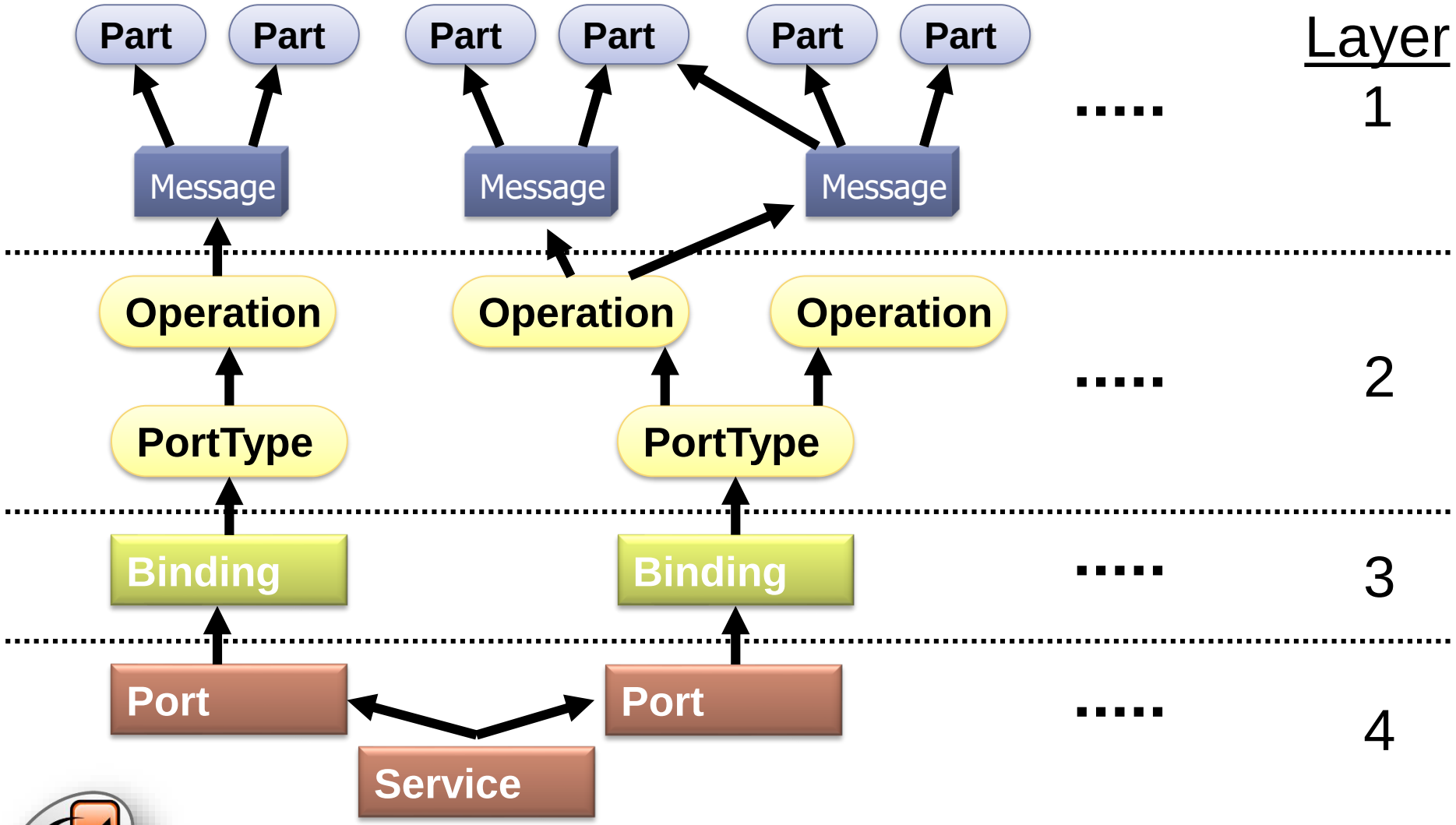


- **Service**
  - A set of related endpoint instances

Vocabulary (1)

- All WSDL elements belong to a WSDL Namespace: http://schemas.xmlsoap.org/wsdl/
- Namespaces for WSDL Binding
  - SOAP Binding: http://schemas.xmlsoap.org/wsdl/soap/
  - HTTP GET and POST Binding: http://schemas.xmlsoap.org/wsdl/http/
  - WSDL MIME binding: http://schemas.xmlsoap.org/wsdl/mime/
  - Various other…
- Development of WSDL specification
- Problem: WSDL-first vs. Code-first
  - **Attention! Follow the rules of WS-I (http://www.oasis-ws-i.org)**

Vocabulary (2)



WSDL – Example (1)

Part

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
          xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
          xmlns:tns="http://example.org/beispielws"
          xmlns:s="http://www.w3.org/2001/XMLSchema"
          xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
          xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
          targetNamespace="http://example.org/beispielws"
          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://example.org/beispielws">
      <s:element name="HelloWorld">
        <s:complexType />
      </s:element>
      <s:element name="HelloWorldResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="HelloWorldResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

WSDL – Example (2)

Part

```
        <s:element name="Add">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="a" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="b" type="s:int" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="AddResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="AddResult" type="s:int" />
    </s:sequence>
  </s:complexType>
</s:element>
</s:schema>
</wsdl:types>
```

WSDL – Example (3)

Message

```
<wsdl:message name="HelloWorldSoapIn">
  <wsdl:part name="parameters" element="tns:HelloWorld" />
</wsdl:message>
<wsdl:message name="HelloWorldSoapOut">
  <wsdl:part name="parameters" element="tns:HelloWorldResponse" />
</wsdl:message>

<wsdl:message name="AddSoapIn">
  <wsdl:part name="parameters" element="tns:Add" />
</wsdl:message>
<wsdl:message name="AddSoapOut">
  <wsdl:part name="parameters" element="tns:AddResponse" />
</wsdl:message>
```

WSDL – Example (4)

Operation

PortType

```
<wsdl:portType name="ServiceSoap">
  <wsdl:operation name="HelloWorld">
    <wsdl:input message="tns:HelloWorldSoapIn" />
    <wsdl:output message="tns:HelloWorldSoapOut" />
  </wsdl:operation>

  <wsdl:operation name="Add">
    <wsdl:input message="tns:AddSoapIn" />
    <wsdl:output message="tns:AddSoapOut" />
  </wsdl:operation>
</wsdl:portType>
```

WSDL – Example (5)

Binding

Operation

```
<wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="HelloWorld">
    <soap:operation soapAction="http://example.org/beispielws/HelloWorld" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>

  <wsdl:operation name="Add">
    <soap:operation soapAction="http://example.org/beispielws/Add" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

WSDL – Example (6)

Binding

Operation

```
<wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="HelloWorld">
    <soap12:operation soapAction="http://example.org/beispielws/HelloWorld" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>

  <wsdl:operation name="Add">
    <soap12:operation soapAction="http://example.org/beispielws/Add" style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

WSDL – Example (7)

Port | Service

```
<wsdl:service name="Service">
  <wsdl:port ="" name="ServiceSoap" binding="tns:ServiceSoap">
    <soap:address location="http://.../WSExample/Service.asmx" />
  </wsdl:port>

  <wsdl:port ="" name="ServiceSoap12" binding="tns:ServiceSoap12">
    <soap12:address location="http://.../WSExample/Service.asmx" />
  </wsdl:port>
</wsdl:service>

</wsdl:definitions>
```

Web Service Demo

- WSDL focus and discussion
- Problem: Multiple vs. single binding
- Approach: Code First vs. WSDL First
- Example: Complex data types
- WS-I Basic Profile

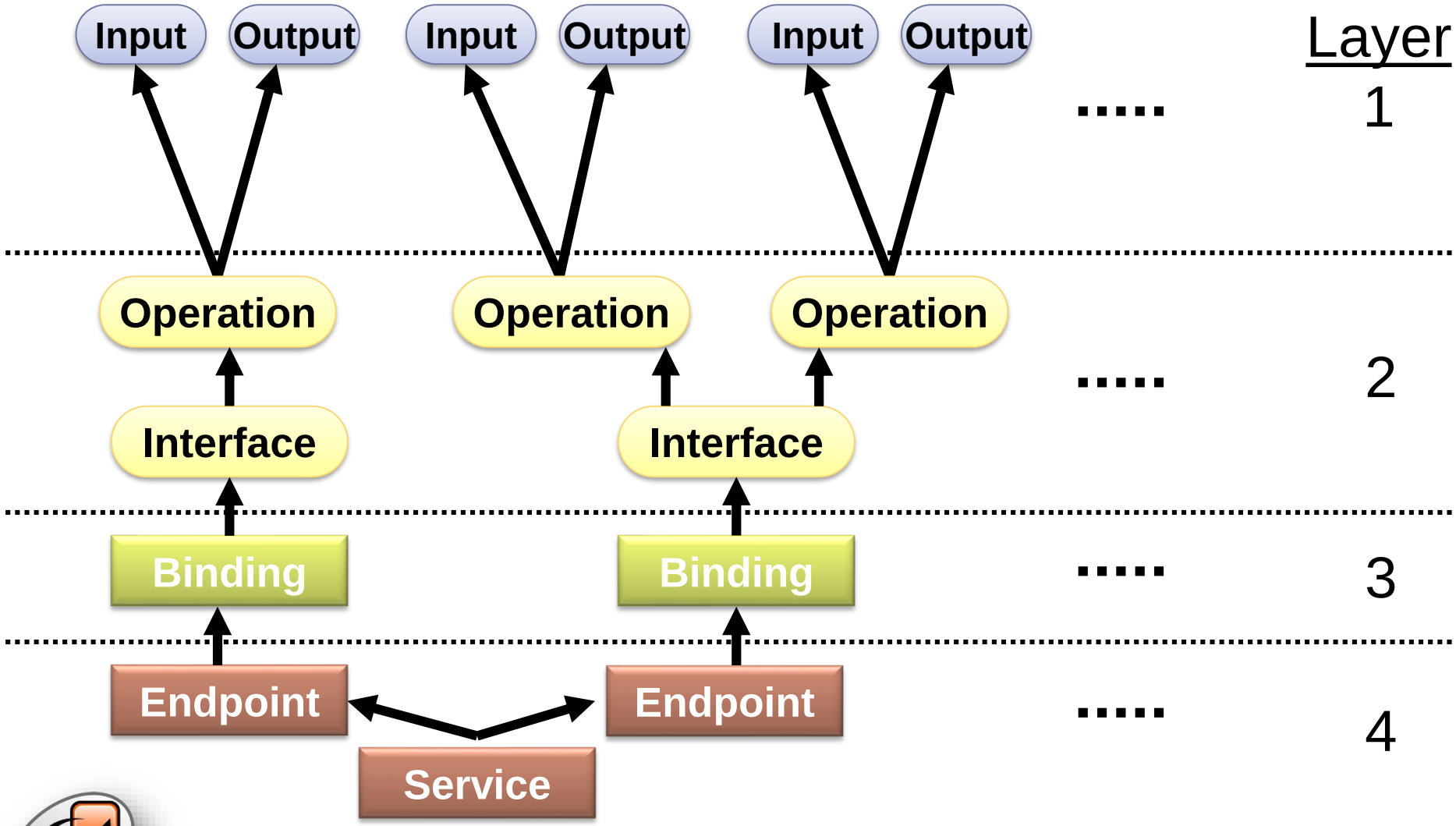
Chapter 3

WSDL 2.0

Changes

- Lower complexity than WSDL 1.1
- Port
  - Renamed to *Endpoint*
- PortType
  - Renamed to *Interface*
  - supporting inheritance by using *extends* attribute
- Message
  - Removed, instead inputs, outputs, faults etc. directly typed
- Improved support for non-SOAP Web services (cf. Part IV REST)

Revised Concepts



WSDL 2.0 Example

Interface Operation Input Output

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description
          targetNamespace="http://ticketAgent.example.com/"
          xmlns:tns="http://ticketAgent.example.com/"
          xmlns:wsdl="http://www.w3.org/ns/wsdl"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.w3.org/ns/wsdl
http://www.w3.org/2007/06/wsdl/wsdl20.xsd">
<wsdl:types>
  <xs:import schemaLocation="TicketAgent.xsd"
namespace="http://ticketAgent.example.com/TicketAgent.xsd" />
</wsdl:types>
<wsdl:interface name="TicketAgent">
  <wsdl:operation name="listFlights" pattern="http://www.w3.org/ns/wsdl/in-out">
    <wsdl:input element="tns:listFlightsRequest"/>
    <wsdl:output element="tns:listFlightsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="reserveFlight" pattern="http://www.w3.org/ns/wsdl/in-out">
    <wsdl:input element="tns:reserveFlightRequest"/>
    <wsdl:output element="tns:reserveFlightResponse"/>
  </wsdl:operation>
</wsdl:interface>
```

Adapted from: http://www.w3.org/TR/wsdl20/

WSDL 2.0 Example

Service | Endpoint | Binding

```
<binding name="HttpBinding" interface="tns:TicketAgent"
                    type="http://www.w3.org/ns/wsdl/http">
  <operation ref="tns:listFlights" whttp:method="GET" />
  <operation ref="tns:reserveFlight" whttp:method="POST"
     whttp:inputSerialization="application/x-www-form-urlencoded" />
</binding>

<service name=„TicketAgentService" interface="tns:TicketAgent">
  <endpoint name="TicketAgentService Endpoint"
                          binding="tns:HttpBinding"
                          address="http://ticketAgent.example.com/" />
</service>

</wsdl:description>
```

Adapted from: http://www.w3.org/TR/wsdl20/

VSR | EDU

Software Service Engineering

Software Service Engineering

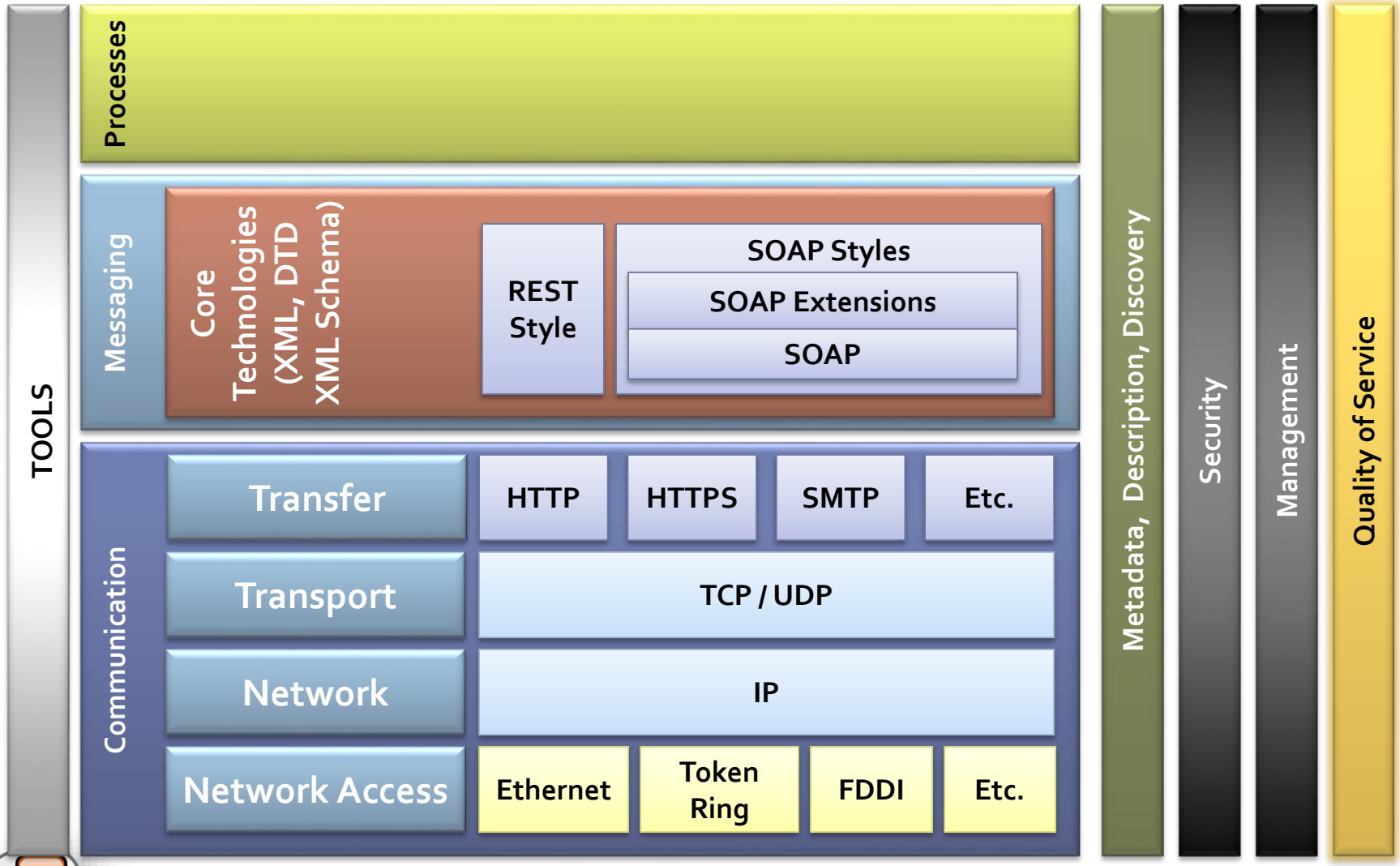
Prof. Dr.-Ing. Martin Gaedke
Technische Universität Chemnitz
Fakultät für Informatik
Professur Verteilte und selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

Chapter 4

WEB SERVICES STACK

Basic Technologies and Extensions



Messaging (1)

- Basic Technology
  - SOAP
- Routing/Addressing
  - Transport technology-independent transfer of messages
  - Examples: *WS-Addressing*, WS-MessageDelivery
- Multiple Message Sessions
  - Transport technology-independent processing of XML resources using Web services
  - Examples: WS-Enumeration, *WS-Transfer*

Messaging (2)

- Events and Notification
  - Event-based architecture implementation independent of transport technology
  - Example: WS-Eventing, WS-Notification
- Reliable Messaging
  - Secure message exchange implementation independent of transport technology
  - Example: WS-Reliable Messaging, WS-Reliability (obsolete)
- Message Packaging
  - Message Transmission Optimization Mechanism (W3C Recommendation 25 January 2005) – for efficient transfer of binary data to and from Web Services. The focus is on transmission optimization of base64 encoded data.
  - Data is transferred as MIME Multipart/Related XML-binary Optimized Package (XOP Package)
  - Example: MTOM (Attachments)

WS-Addressing

- Web Services Addressing (WS-Addressing)
  - W3C Member Submission 10 August 2004
  - http://www.w3.org/Submission/ws-addressing/
  - WS Addressing provides a transport technology neutral mechanism of addressing Web Services and their messages

WS-Addressing – Example (1)

- Specified Request structure

```
<?xml version="1.0" encoding="utf-8" ?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
            xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <S:Header>
    <wsa:MessageID>
      uuid:12345678-1234-5678-ABCD-123456789ABC
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://FromCompanyA.example.org/Buyer</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To S:mustUnderstand="1">http://ToComapnyB.example.org/Purchasing</wsa:To>
    <wsa:Action>http://comapnyB.example.org/SubmitOrder</wsa:Action>
  </S:Header>
  <S:Body>
    <!--XML-Code for Order-->
  </S:Body>
</S:Envelope>
```

WS-Addressing – Example (2)

- Specified Reply structure

```
<?xml version="1.0" encoding="utf-8" ?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
            xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
            xmlns:example="http://Schemas.ToComapnyB.example.org/Purchasing">
  <S:Header>
    <wsa:MessageID>
      uuid:aaaabbbb-cccc-dddd-eeee-wwwwwwwwwww
    </wsa:MessageID>
    <wsa:RelatesTo>
      uuid:12345678-1234-5678-ABCD-123456789ABC
    </wsa:RelatesTo>
    <wsa:To S:mustUnderstand="1">
      http://FromCompanyA.example.org/Buyer
    </wsa:To>
    <wsa:Action>http://ToComapnyB.example.org/OrderReceived</wsa:Action>
  </S:Header>
  <S:Body>
    <example:OrderReceived/>
  </S:Body>
</S:Envelope>
```

WS-Transfer

- Web Services Transfer (WS-Transfer)
  - W3C Member Submission 27 September 2006
  - http://www.w3.org/Submission/WS-Transfer/
  - Specification describes a generic SOAP-based protocol to process any XML-based representation of a Web Service-based resource
  - Idea: provide Create, Read, Update and Delete as a Web Service to enable processing of any XML resource independently of communication mechanisms
  - Uses WS Addressing
- Implementation of the idea
  - Resource Operations: Get, Put, Delete
  - Resource Factory Operation: Create
  - Faults

WS-Transfer – Get

- Resource Operation Get

```
<s:Envelope …="">
  <s:Header …="">
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …="">
    ...
  </s:Body>
</s:Envelope>
```

```
<s:Envelope …="">
  <s:Header …="">
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …="">
    xs:any
    ...
  </s:Body>
</s:Envelope>
```

Resource operations behaving accordingly:
Put – PutResponse
Delete – DeleteResponse

WS-Transfer – Create

- Resource Factory Operation: Create

```
<s:Envelope …="">
  <s:Header …="">
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
    </wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …="">
    xs:any
    ...
  </s:Body>
</s:Envelope>
```

```
<s:Envelope …="">
  <s:Header …="">
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse
    </wsa:Action>
    <wsa:RelatesTo>xs:anyURI</wsa:RelatesTo>
    <wsa:To>xs:anyURI</wsa:To>
    …
  </s:Header>
  <s:Body …="">
    <wxf:ResourceCreated>endpoint-reference</wxf:ResourceCreated>
    xs:any ?
  </s:Body>
</s:Envelope>
```

WS-Transfer – Example: Create

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:example="http://schemas.example.com/resource-model" >
  <s:Header>
    <wsa:ReplyTo>
      <wsa:Address>
        soap://sender.example.org/
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>soap://www.example.org/websvc/</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
    </wsa:Action>
    <wsa:MessageID>
      uuid:12345678-abcd-1234-eeff-123456781234
    </wsa:MessageID>
  </s:Header>
  <s:Body>
    <example:Product>
      <example:title>Seife</example:title>
      <example:price>1.22</example:price>
    </example:Product>
  </s:Body>
</s:Envelope>
```

Metadata, Description, Discovery (1)

- Basic technology
  - WSDL and semantic extension
  - UDDI
- Policy
  - Specification to enable Web Services to describe their respective usage rules or to allow consumers to describe such requirements
  - Policies can apply to security, quality, etc
  - Examples: WS-Policy, WS-PolicyAssertions, WS-PolicyAttachment

- WS Policy Example
  - wsp:ExactlyOne, wsp:All

```
<wsp:Policy
    xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" >
  <wsp:ExactlyOne>
    <sp:Basic256Rsa15 />
    <sp:TripleDesRsa15 />
  </wsp:ExactlyOne>
</wsp:Policy>
```

Metadata, Description, Discovery (2)

- Discovery
  - Describes specification of Service discovery independent of communication technology
  - Example: WS-Discovery
    - Defines a Multicast Discovery Protocol for Web Services localization
- Metadata Retrieval
  - Describes specification of obtaining Web Services' metadata independent of communication technology
  - Example: WS-MetadataExchange

WS-MetadataExchange

- Web Services Metadata Exchange (WS MetadataExchange / WS-Mex)
  - Specification http://schemas.xmlsoap.org/ws/2004/09/mex/
  - Web Service offers various metadata to let the other endpoints (consumers) know how one can communicate with it, for example, WSDL, Policy etc.
    - Example: http://ws.example.org/service?WSDL
    - Problem (1): Convention for WSDL does not allow end-to-end communication at the message level
    - Problem (2): Convention for WSDL, what about other metadata?
- Idea: Protocol for metadata exchange independent of communication mechanisms
  - Solution: Metadata is resource of a Web Service.
  - Resource is made available over WS-Transfer in terms of Request/Response

WS-MetadataExchange – Example

- http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Request

```
<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:mex="http://schemas.xmlsoap.org/ws/2004/09/mex" >
  <s:Header>
    <wsa:To>http://ws.example.org/webservice</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Request
    </wsa:Action>
    <wsa:MessageID>
      urn:uuid:12345678-4321-dddd-cccc-abcdef6543212
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://consumer.example.org</wsa:Address>
    </wsa:ReplyTo>
  </s:Header>
  <s:Body>
    <mex:GetMetadata>
      <mex:Dialect>http://schemas.xmlsoap.org/ws/2004/09/policy</mex:Dialect>
      <mex:Identifier>http://ws.example.org/webservice/policy</mex:Identifier>
    </mex:GetMetadata>
  </s:Body>
</s:Envelope>
```

WS-MetadataExchange – Example

- http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Response

```
<s:Envelope
       xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:wsa="http://www.w3.org/2005/08/addressing"
       xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
       xmlns:mex="http://schemas.xmlsoap.org/ws/2004/09/mex">
  <s:Header>
    <wsa:To>http://consumer.example.org</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Response
    </wsa:Action>
    <wsa:RelatesTo>
      urn:uuid:12345678-4321-dddd-cccc-abcdef6543212
    </wsa:RelatesTo>
  </s:Header>
  <s:Body>
    <mex:Metadata>
      <mex:MetadataSection
         Dialect="http://schemas.xmlsoap.org/ws/2004/09/policy"
         Identifier="http://ws.example.org/webservice/policy">
        <wsp:Policy>
            <!-- Policy description-->
        </wsp:Policy>
      </mex:MetadataSection>
    </mex:Metadata>
  </s:Body>

</s:Envelope>
```

Processes

- Business Domain
  - Different approaches, e.g. OASIS
  - Portal and Presentation: WSRP
- Transactions and Business Processes
  - WS-BusinessActivity
  - WS-AtomicTransaction
  - BPEL4WS
  - WS-Humantask, BPEL4People
- Aggregation, Choreography, Composition and Coordination
  - WS-Coordination
  - WS-Choreography / WS-CDL

Management

- Distributed Management
  - Web Services Distributed Management (WSDM)
    - Management of Web Services (MOWS)
    - Management using Web Services (MUWS)
    - WS-Resource Framework (WS-RF)
  - WS-Manageability
- Provisioning
  - OASIS: "Provisioning is the automation of all the steps required to manage (setup, amend & revoke) user or system access entitlements or data relative to electronically published services".
  - Specification of mechanisms (APIs and schemas) to realize interoperability of provisioning systems based on SOAP message exchange
  - Example: WS-Provisioning

Security & Co.

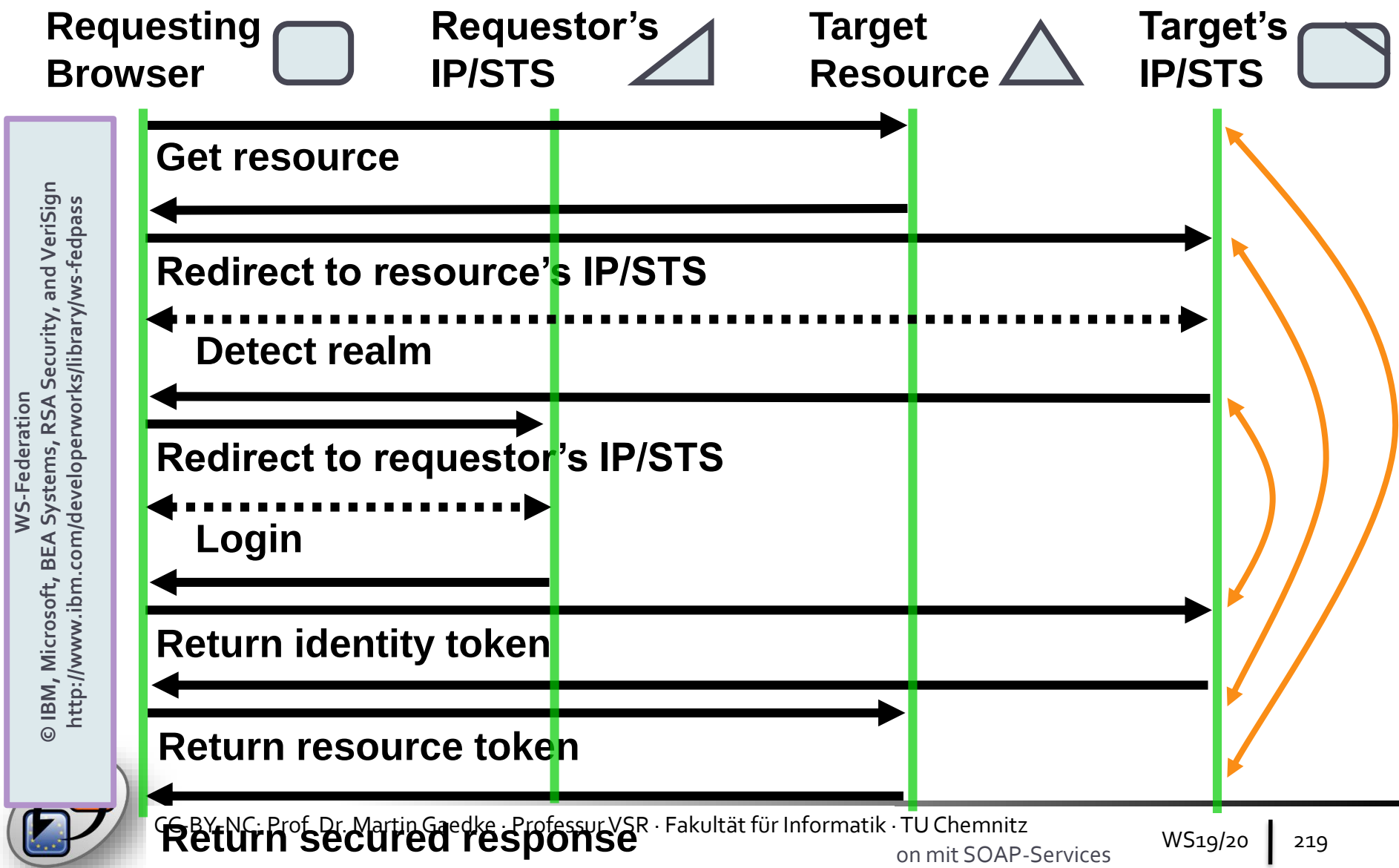
- Security
  - Specifies a protocol for secure end-to-end communication. SOAP messages are get signed and encrypted. WS-Security provides a framework for securing SOAP messages using the W3C recommendations regarding XML-Signature Syntax and Processing (XML Encryption, XML Signature)
  - Example: WS-Security
- Security Policy and Secure Conversation
  - Extend the capabilities of WS-Security accordingly
  - Examples: WS-SecurityPolicy, WS-SecureConversation
- Trusted Message
  - SOAP-based mechanisms for mediation of trust relationships as well as request / invalidation of security tokens
  - Example: WS-Trust
- Further approaches
  - Examples: Privacy (WS-Privacy), Authorization (WS-Authorization)

- Current trend: Federated Identity
  - Example: WS-Federation
  - Example: http://webcomposition.net/idfs

Section

FEDERATION MIT SOAP-SERVICES

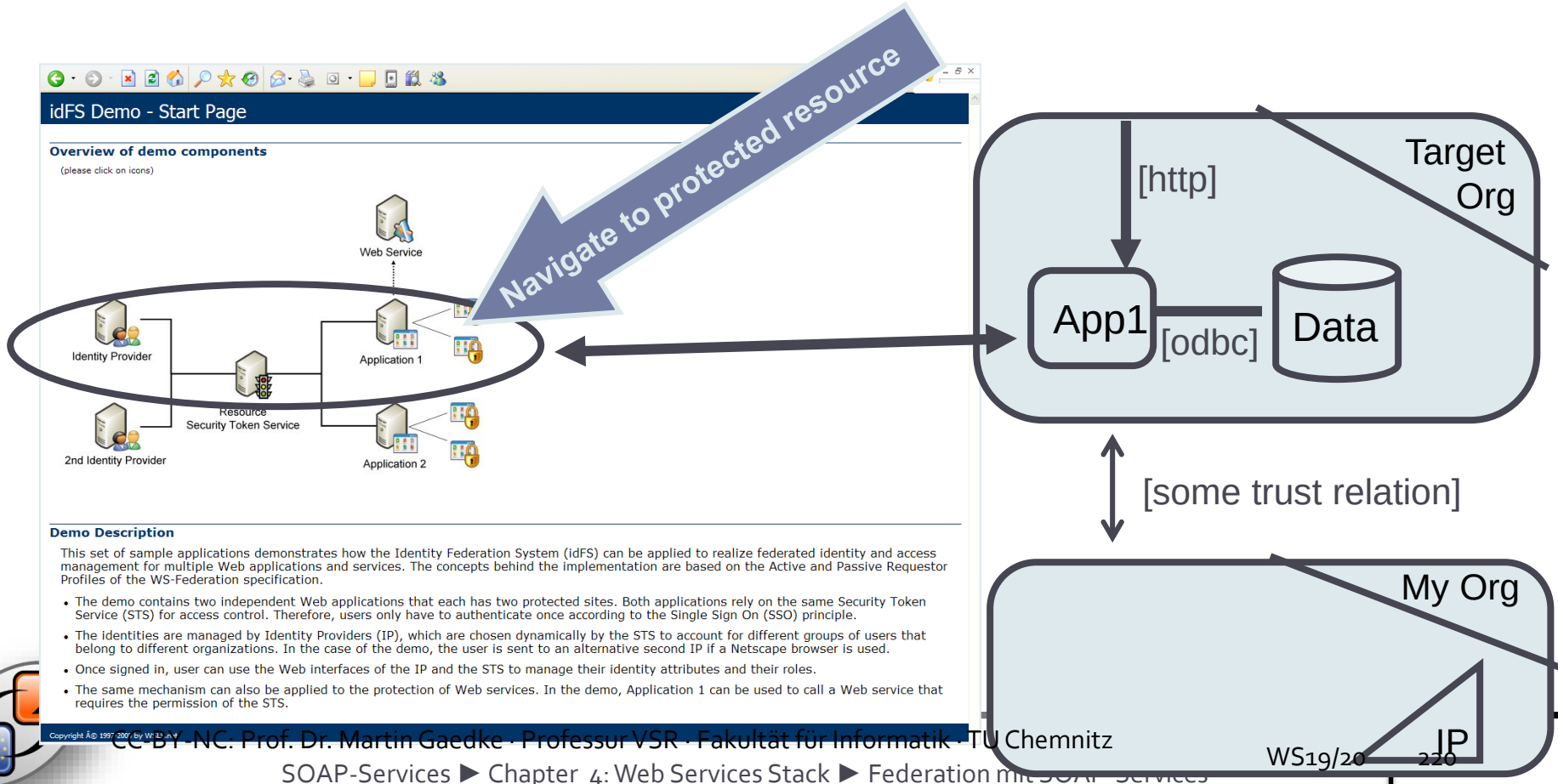
WS-Federation PRP Sample



WS-Federation
© IBM, Microsoft, BEA Systems, RSA Security, and VeriSign
http://www.ibm.com/developerworks/library/ws-fedpass

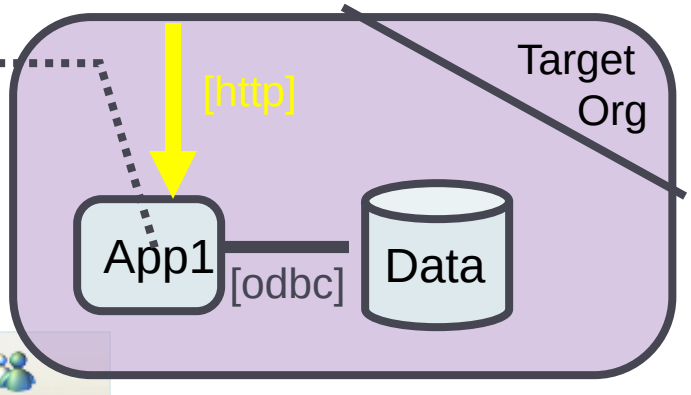
Example – WAM Applied

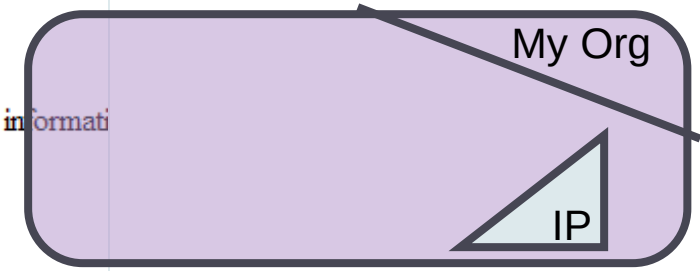
- http://webcomposition.net/idfs/demo/
- SSO with WS-Federation (PRP)



Call Protected Resource

No access token with permissions for this protected resource available (security token signed by STS of Target Org)





idFS (c) 2004-2005 MWRG, University of Karlsruhe

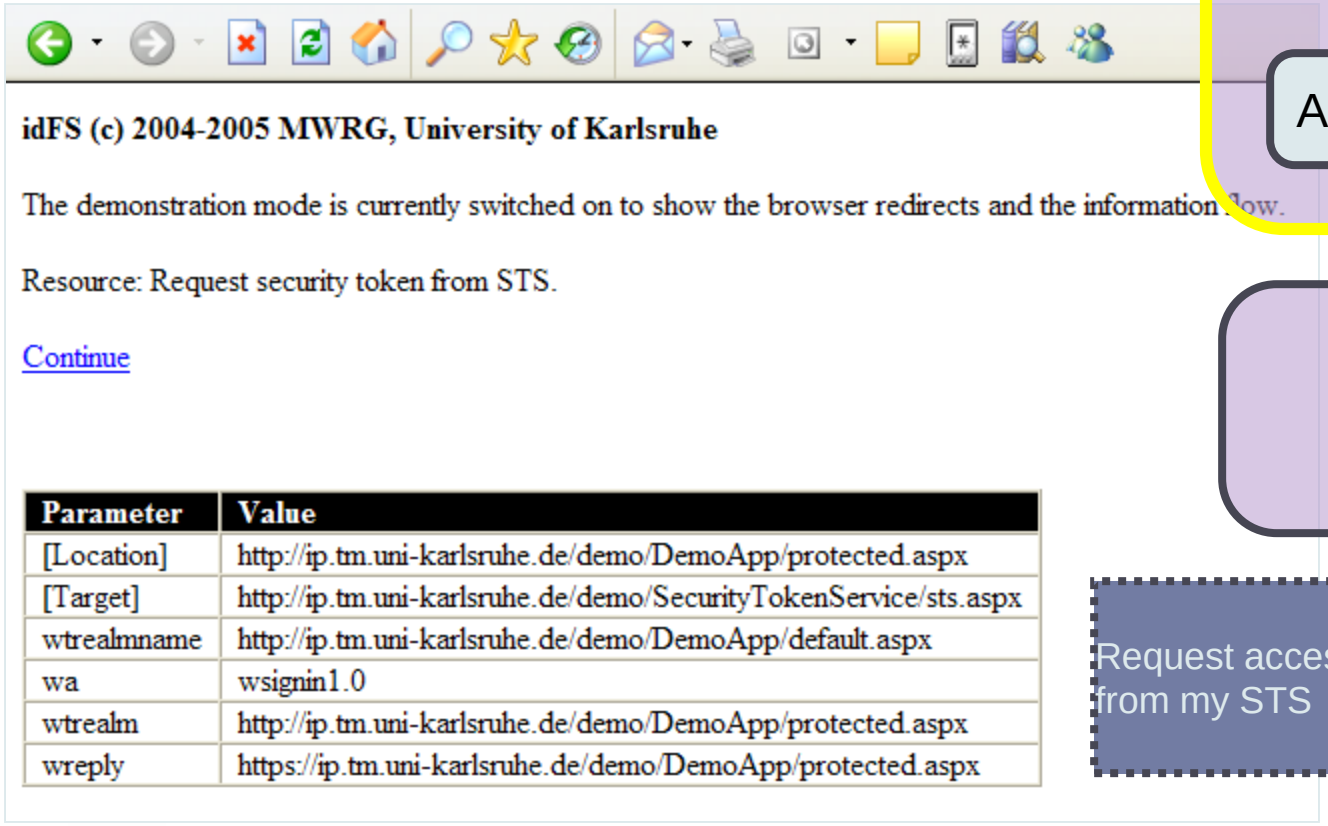
The demonstration mode is currently switched on to show the browser redirects and the informati

Resource: Request security token from STS.

Continue

Parameter	Value
[Location]	http://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
[Target]	http://ip.tm.uni-karlsruhe.de/demo/SecurityTokenService/sts.aspx
wtrealmname	http://ip.tm.uni-karlsruhe.de/demo/DemoApp/default.aspx
wa	wsignin1.0
wtrealm	http://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
wreply	https://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx

Protected Resource → STS



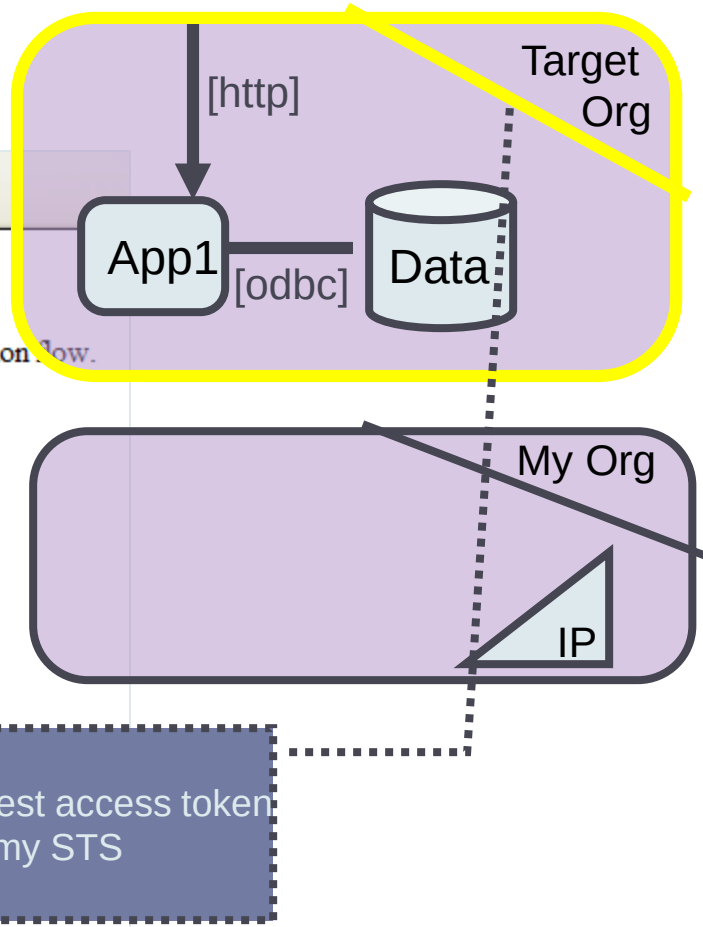
idFS (c) 2004-2005 MWRG, University of Karlsruhe

The demonstration mode is currently switched on to show the browser redirects and the information flow.

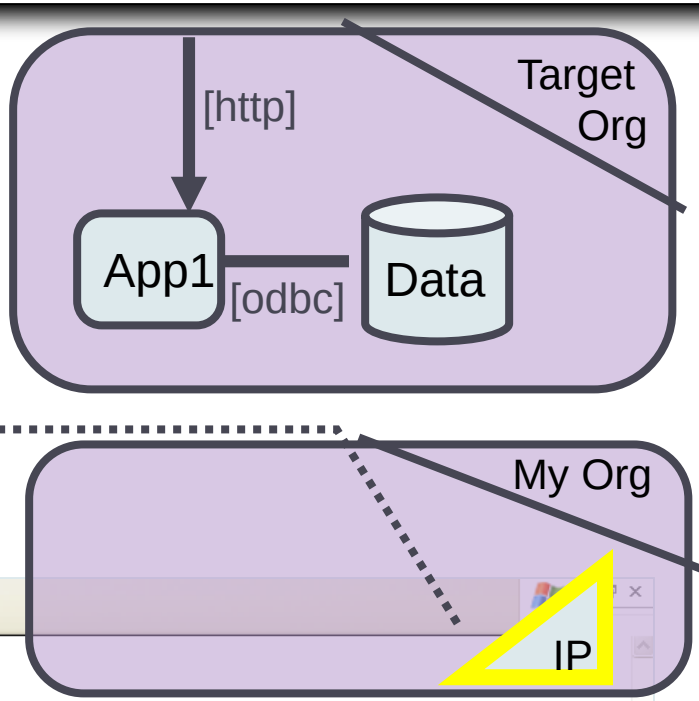
Resource: Request security token from STS.

Continue

Parameter	Value
[Location]	http://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
[Target]	http://ip.tm.uni-karlsruhe.de/demo/SecurityTokenService/sts.aspx
wtrealmname	http://ip.tm.uni-karlsruhe.de/demo/DemoApp/default.aspx
wa	wsignin1.0
wtrealm	http://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
wreply	https://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx



STS→IP



Request identity token from IP for authentication (security token signed by IP)

idFS (c) 2004-2005 MWRG, University of Karlsruhe

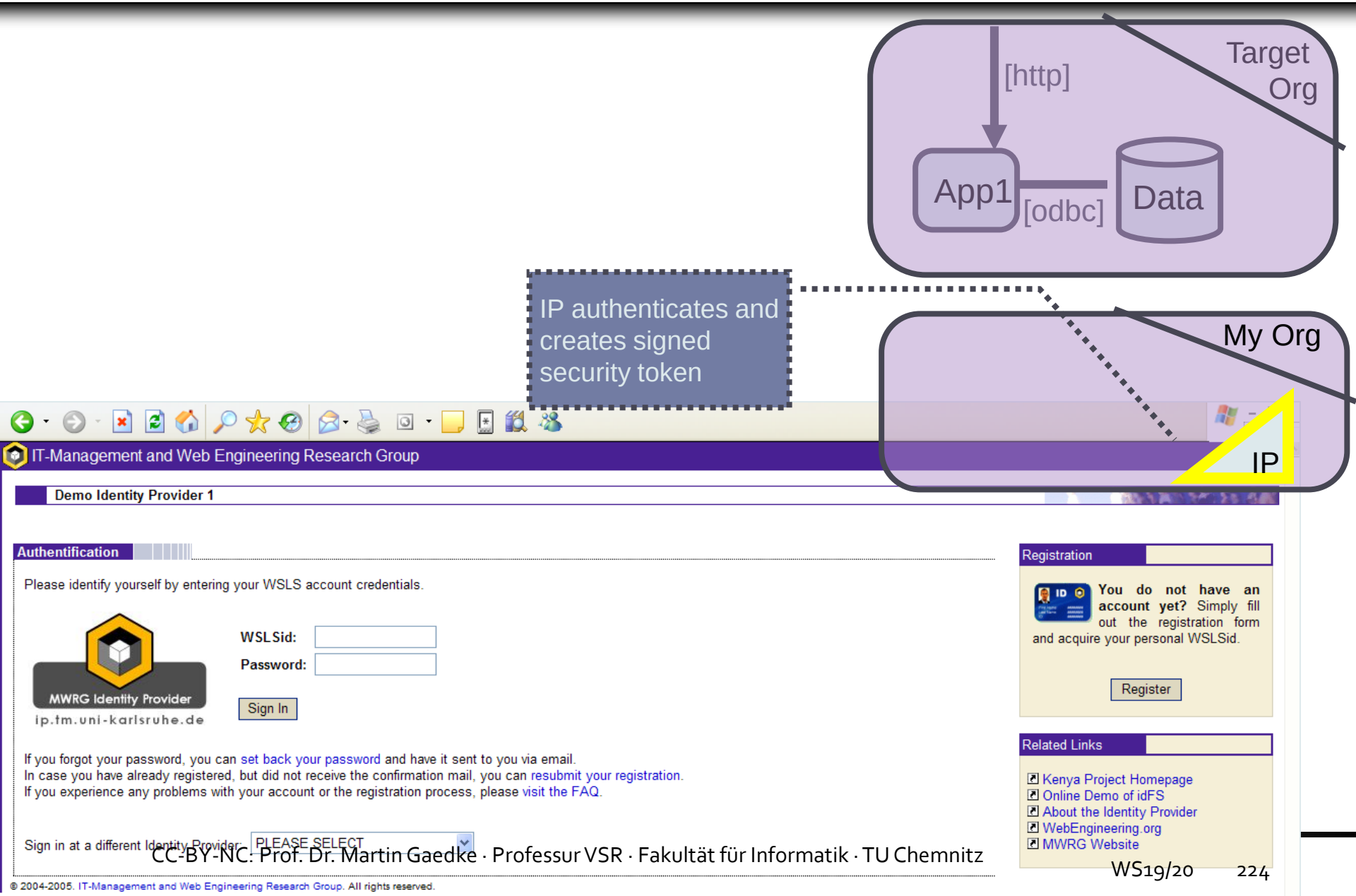
The demonstration mode is currently switched on to show the browser redirects and the information flow.

STS: Pass on token request to the IP.

Continue

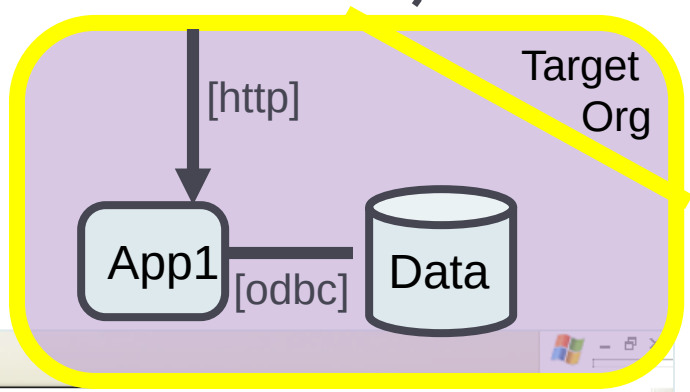
Parameter	Value
[Location]	http://ip.tm.uni-karlsruhe.de/demo/SecurityTokenService/sts.aspx?wtrealmname=http://ip.tm.uni-karlsruhe.de/demo/DemoApp/default.aspx&wa=wsignin1.0&wtrealm=http://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx&wreply=https://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
[Target]	https://ip.tm.uni-karlsruhe.de/demo/IdentityProvider/Default.aspx
wtrealmname	http://ip.tm.uni-karlsruhe.de/demo/DemoApp/default.aspx
wa	wsignin1.0
wtrealm	http://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
wctx	https://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
wreply	https://ip.tm.uni-karlsruhe.de/demo/SecurityTokenService/sts.aspx

IP - Authenticate



IP→STS

Post identity token (IP security token) to STS

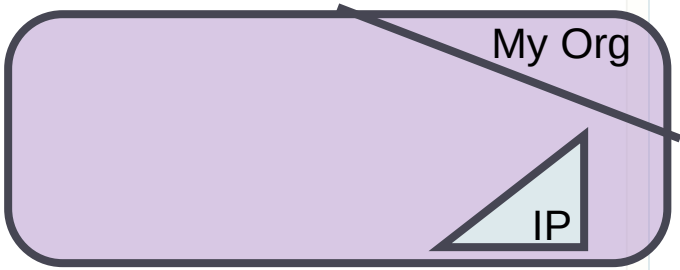


idFS (c) 2004-2005 MWRG, University of Karlsruhe

The demonstration mode is currently switched on to show the browser redirects and the information flow.

IP: Post requested token.

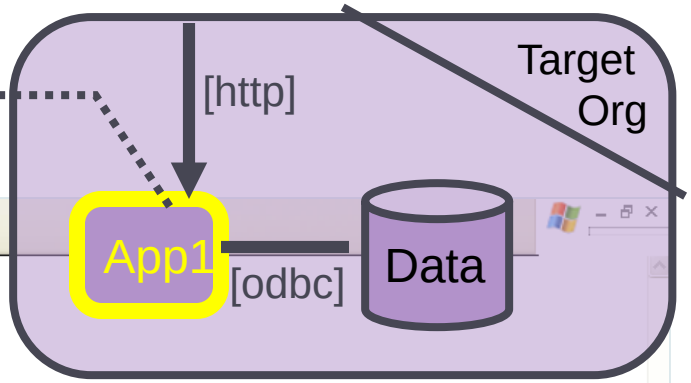
Continue



Parameter	Value
[Location]	https://ip.tm.uni-karlsruhe.de/demo/IdentityProvider/Default.aspx?wtrealmname=http://ip.tm.uni-karlsruhe.de/demo/DemoApp/default.aspx&wa=wsignin1.0&wtrealm=http://ip.tm.uni-karlsruhe.c
[Target]	https://ip.tm.uni-karlsruhe.de/demo/SecurityTokenService/sts.aspx
wa	wsignin1.0
wctx	https://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
wresult	<soap:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ws open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-284356be-ef18-4dac-a333-efbe2eb03fe0">MIIBzzCCATigAwIBAgIEAmuIHzANBgkqhkiG9w0 xmlns="http://www.w3.org/2000/09/xmldsig#"><SignedInfo><CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" /><SignatureMethod Algorithm="ht Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" /><DigestValue>cy+U3Y7d3yftnSDzoXpV/9KKC14=</DigestValue></Reference><Reference URI="#timestamp"><Transforms><Tra 1.0#X509v3" /></wsse:SecurityTokenReference></KeyInfo></Signature></wsse:Security><wsa:To>http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous</wsa:To></soap:Heac 978cd3836306</NameIdentifier></Subject></AuthenticationStatement><AttributeStatement><Subject><NameIdentifier>a2725518-c4de-4e4b-8d48-978cd3836306</NameIdentifier></Sub xsi:type="xsd:string">meinecke@tm.uka.de</AttributeValue></Attribute><Attribute AttributeName="IsModifiable" AttributeNamespace="http://wsls.net/2004/05/WSLS/Federation/Identity"><. AttributeNamespace="http://wsls.net/2004/05/WSLS/Federation/Identity"><AttributeValue xsi:type="xsd:string">wa</AttributeValue></Attribute><Attribute AttributeName="FirstName" Attrib AttributeNamespace="http://wsls.net/2004/05/WSLS/Federation/Identity"><AttributeValue xsi:type="xsd:string" /></Attribute><Attribute AttributeName="Source" AttributeNamespace="http://v Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" /><DigestValue>n29fafKTPd0RGp1Tccuy6c9YUuw=</DigestValue></Reference></SignedInfo><SignatureValue>STiqVk1cocoiIj0DN xmlns="http://www.w3.org/2000/09/xmldsig#"><X509Certificate>MIIBzzCCATigAwIBAgIEAmuIHzANBgkqhkiG9w0BAQQFADAsMQswCQYDVQQGEwJERTEdMBsGA1UEAxMUaW

STS→Protected Resource

STS checks the identity encoded in the IP security token and creates the requested access token (security token signed by STS) for App1



idFS (c) 2004-2005 MWRG, Univ

The demonstration mode is currently switched on to show the browser redirects and the information flow.

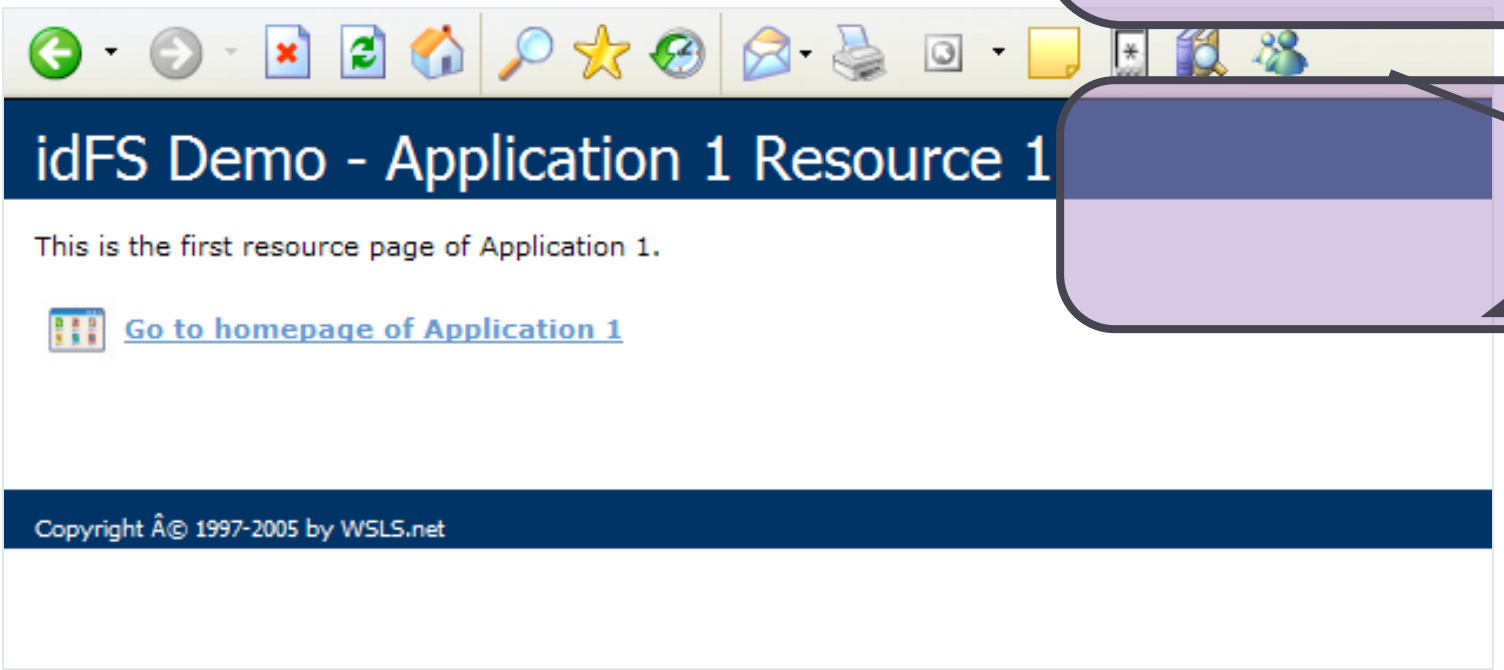
STS: Post requested token to the resource.

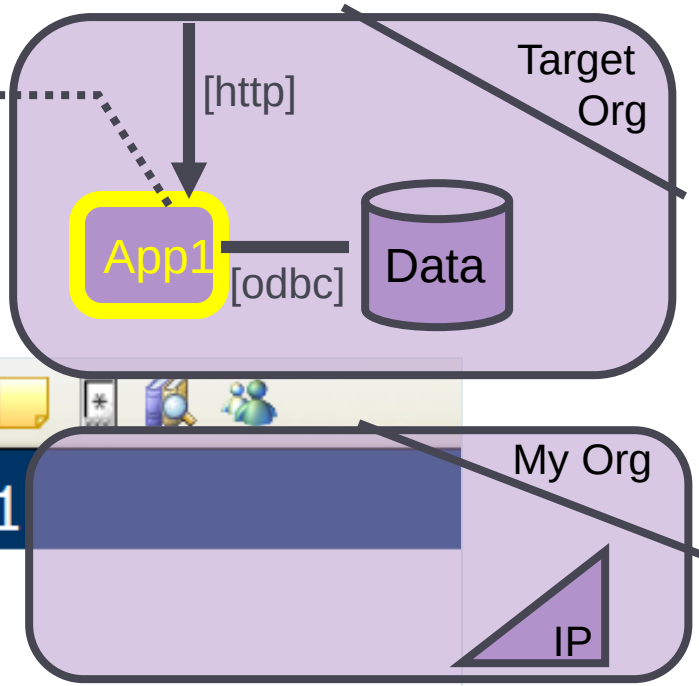
Continue

Parameter	Value
[Location]	https://ip.tm.uni-karlsruhe.de/demo/SecurityTokenService/sts.aspx
[Target]	https://ip.tm.uni-karlsruhe.de/demo/DemoApp/protected.aspx
wa	wsignin1.0
wresult	<soap:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:ws wss-wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-4b975ec2-e3bb-4c79-b2ac-2ad56873692a">MIIBzzCCATigAwIBAgIEAnuIHzANBgkqhkiG9w0BAQQFADAsMQswCQYDVQQ Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" /><SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" /><Reference URI="#body"><Transforms xml-c14n-20010315" /></Transforms><DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" /><DigestValue>2DmuHlOWVek/dxRKF/FX1j7eFiw=</DigestValue></Referer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" MajorVersion="1" MinorVersion="1" AssertionID="i9zkbyaEYgZJNtclraKSbC AttributeName="LoginName" AttributeNamespace="http://wsls.net/2004/05/WSLS/Federation/Identity"><AttributeValue xsi:type="xsd:string">wa</AttributeValue></Attribute><Attribute Attril AttributeNamespace="http://wsls.net/2004/05/WSLS/Federation/Identity" /><Attribute AttributeName="Source" AttributeNamespace="http://wsls.net/2004/05/WSLS/Federation/Identity"><Att 2a76781842c3</Identifier><ServiceId xmlns="http://www.wsls.net/2002/03/gts/isid" /></ISID><ITX xmlns="http://www.wsls.net/2002/03/gts/contentObject" /><UnitTy xmlns="http://www.wsls.net/2002/03/gts/contentObject">94b4d8fc-fd1a-4451-8805-2a76781842c3</Identifier><Source xmlns="http://www.wsls.net/2002/03/gts/contentObject" /& AttributeNamespace="http://wsls.net/2004/05/WSLS/Federation/Identity"><AttributeValue xsi:type="xsd:string"><Role xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://w xmlns="http://www.wsls.net/2002/03/gts/contentObject">Sample Role</Title><Creator xmlns="http://www.wsls.net/2002/03/gts/contentObject">System</Creator><Subje xmlns="http://www.wsls.net/2002/03/gts/contentObject" /><Coverage xmlns="http://www.wsls.net/2002/03/gts/contentObject" /><Rights xmlns="http://www.wsls.net/2002/03/gts/con xmlns="http://www.wsls.net/2002/03/gts/contentObject">Set</gts1C580465DBFB47DA917BF0BE139D53CA><UnitCost xmlns="http://www.wsls.net/2002/03/gts/contentObject" xmlns="http://www.wsls.net/2002/03/gts/contentObject">authenticated</Description><Publisher xmlns="http://www.wsls.net/2002/03/gts/contentObject" /><Contributor xmlns=" 02T13:59:11.2339683+01:00</LastModified><LastModifier xmlns="http://www.wsls.net/2002/03/gts/contentObject" /><ServiceId xmlns="http://www.wsls.net/2002/03/gts/conter Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" /><DigestValue>haeN2ZodDv/66bsVpO3LiuZZLMU=</DigestValue></Reference></SignedInfo><SignatureValue>DX4Xzz4G7FQXl xmlns="http://www.w3.org/2000/09/xmldsig#"><X509Certificate>MIIBzzCCATigAwIBAgIEAnuIHzANBgkqhkiG9w0BAQQFADAsMQswCQYDVQQGEwJERTEdMBsGA1UEAxMUaW

Show Protected Resource

App1 retrieves the access token (STS security token) and presents data based on the permissions (e.g. roles) encoded in the token





VSR | EDU

Software Service Engineering

Software Service Engineering

Prof. Dr.-Ing. Martin Gaedke
Technische Universität Chemnitz
Fakultät für Informatik
Professur Verteilte und selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

PART IV

REPRESENTATIONAL STATE TRANSFER (REST)

Chapter 1

ARCHITECTURAL STYLES

Part I – What is SSE

- Chapter – Introduction
  - History Middleware, Web, today: Service
  - What is SSE – programming without a call stack
  - Top-Down and Buttom Up
  - Service and Business
  - Chapter – Eigenschaften, Principles
    - Vgl. Eric Wilde & Martin Gaedke Paper
    - Evolution, etc.
- Chapter – Architectural Styles
  - Vgl. Paper mit Eric Wild
  - Architectural styles: vgl. REST-Diss. Fielding
- Chapter – Patterns
  - Pattern common: integration, staging, http://www.enterpriseintegrationpatterns.com/eaipatterns.html
  - API Facade Pattern , vgl. API façade
  - Common patterns: Pdf von APIgee: Errors, Responses, Data, URLs, Versions, Data formats
- Chapter – Challenges & open-(business)world
  - Business-world meets IT
  - Heterogeneity, (Verweis auf Enterprise Application Integration, Business Bezug)
  - Business IT-Alignment
  - Federation principle

Architectural Style (years ago ;-)

- GOTHIC (King-style)
  - Ribbed vault
  - Two arches instead of four
  - High, disrupted walls
  - Rose windows
  - Pointed arches
  - Flying buttresses

Architectural Style

- "An **architectural style** is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style." – *Roy Fielding*

Network-based Architectural Styles

- Focus on specific aspects and styles of the Architecture of a distributed software
- Examples
  - Data-flow-Styles
  - Replication Styles
  - Hierarchical Styles
  - Mobile Code Styles
  - Peer-to-Peer Styles

Network-based Architectural Styles

- Data-flow-Styles: Pipe and Filter
  - Each component (filter) reads streams of data on its inputs and produces streams of data on its outputs, usually while applying a transformation to the input streams and processing them incrementally so that output begins before the input is completely consumed.
- Replication Styles: Cache
  - Replication of the result of an individual request such that it may be reused by later requests

Network-based Architectural Styles

- Hierarchical Styles: Client Server
  - A server component, offering a set of services, listens for requests upon those services. A client component, desiring that a service be performed, sends a request to the server via a connector.
- Mobile Code Styles: Code on Demand
  - A client component has access to a set of resources, but not the know-how on how to process them. It sends a request to a remote server for the code representing that know-how, receives that code, and executes it locally.

Network-based Architectural Styles

- Peer-to-Peer Styles: Event-based Integration
  - The event-based integration style, also known as the implicit invocation or event system style, reduces coupling between components by removing the need for identity on the connector interface.
  - Instead of invoking another component directly, a component can announce (or broadcast) one or more events.

Chapter 2

INTRODUCTION TO REST

Fielding on ReST

- "Representation State Transfer is intended to evoke an image of how a **well-designed Web application** behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by **selecting links** (state transitions), **resulting in the next page** (representing the next state of the application) being **transferred to the user** and rendered for their use." - *Roy Fielding*

REST - Introduction

- Representational State Transfer (REST)
  - Concept created by Roy Fielding as a part of "Architectural Styles and the Design of Network-based Software Architectures" dissertation, which describes an architectural style for distributed networked systems
  - Goal: Good design of distributed web applications
- Therefore, the Web itself is to be regarded as an example of REST
  - However, REST is so much more
  - Imposes high requirements on the components of such a distributed solution

REST - Introduction

- REST is a stronger peculiarity of WSA
  - *Architectural approach* – no API, no product etc.
  - Focus lies on use of the existing standards (HTTP, XML, URI) for realization of Web-based solutions
    - URIs are regarded as a concept
    - Concept of state machines
    - Statelessness as a performance factor (enables caching)
  - Considerations are focused on the interworking of clients, servers, statelessness, uniform interfaces, cache, gateways and proxies

Architectural Style - Implementation

- Viewpoint: Abstraction decline

- Architecture – Style
  - REST as an optimal architectural style in the Web
  - Comparison: GOTHIC as an optimal architecture and arts style of the Middle Ages until approximately 1500 AD
- Architecture
  - Planned design of a system that meets the constraints of the REST architecture style
  - Comparison: GOTHIC, development of an individual building according to the properties set by the GOTHIC style
- Implementation
  - ATOM protocol, GData, Microsoft ADO.NET Data Services as implementations, which meet REST constraints
  - Comparison: GOTHIC, Cologne Cathedral, Notre-Dame de Reims Cathedral

REST – ABC

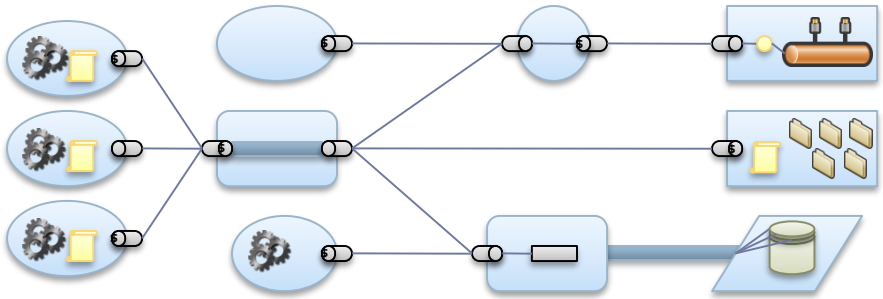
- **A**ddress – URI
- **B**inding – HTTP
- **C**ontract – WADL/Mime-Type/CRUD

	REST	SOAP
Address	URI	URI
Binding	HTTP	HTTP, TCP, etc.
Contract	WADL/Mime-Type/CRUD	WSDL

Chapter 3
IMPLEMENTATION

REST - Introduction

- Now: REST
  - Client-Server model
  - Statelessness
  - Cache
  - Uniform interface
  - Multilayered system
  - Code-on-Demand
  - Data

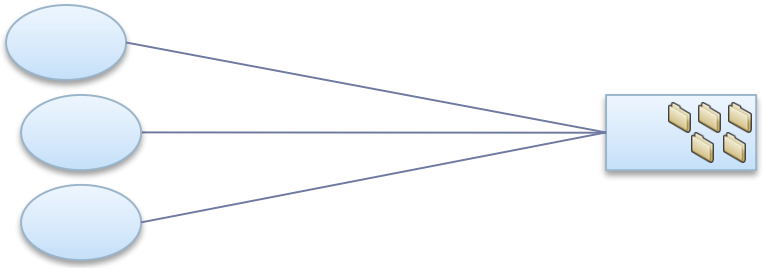


REST – In Detail (1)

- Restriction 1: Client-Server Model
  - "Separation of Concerns": UIX and data are considered strictly separately
  - Enables increased portability
  - Increases scalability by simplifying server components
  - Separation allows for independent evolution of components

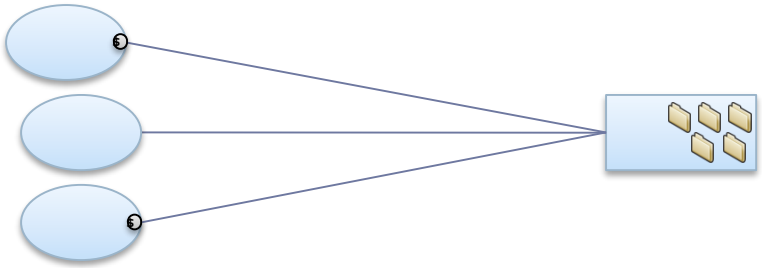
REST – In Detail (2)

- Restriction 2: Statelessness
  - Communication must take place in a stateless manner
  - Request must contain all information necessary to understand it
  - Session information is the client's responsibility – and only his
  - Enables fast post-fault regeneration as no context has to be taken into account
  - Increases scalability, since the server doesn't have to save any states
- Disadvantages?
  - Overhead due to repeated data sending
  - Server loses control over application's behavior



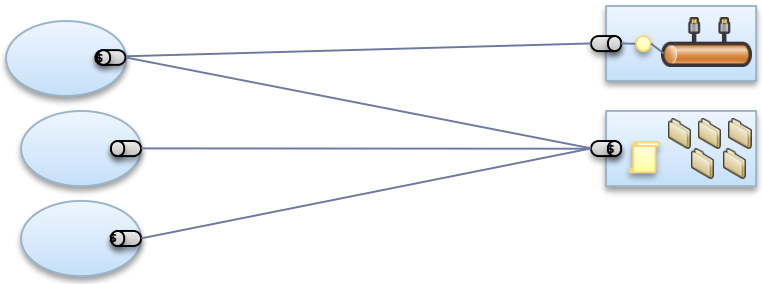
REST – In Detail (3)

- Restriction 3: Cache
  - Data in a Response must be implicitly labeled as cacheable
  - Client can reuse that data for identical Requests
  - Increased scalability by data volume reduction
  - Reduced latency given clever use
- Disadvantages?
  - Declining reliability if data in the cache drastically differs from data on the server



REST – In Detail (4)

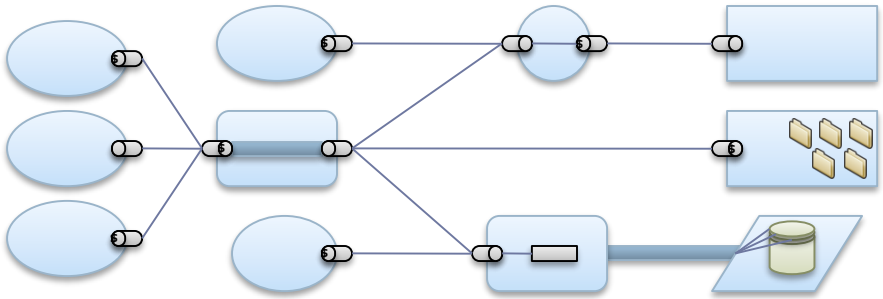
- Restriction 4: Uniform interface
  - Key property of the REST architecture style
  - Software engineering principle of a universal component interface
  - Enables loose coupling of components
- Disadvantages?
  - Efficiency: Data has to be transformed before and after transmission, since it is not transmitted in the application-specific format



REST – In Detail (5)

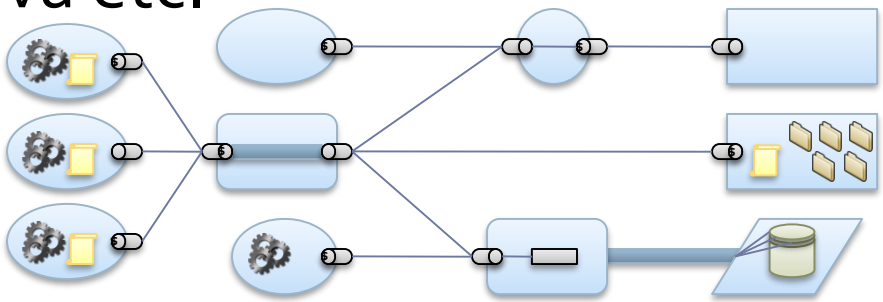
- Restriction 5: Multilayered system
  - System can consist of multiple hierarchically ordered layers
  - Interaction is limited to direct neighbors of the layer
  - Enables encapsulation of legacy systems
  - Protection of new systems from legacy
  - Increased scalability, for example, by load balancing
  - Layers can process data on their own since REST messages contain all the necessary information (see statelessness)
- Disadvantages?
  - Further overhead due to multiple additional layers
  - Latency increase

  → Caching



REST – In Detail (6)

- Restriction 6: Code-On-Demand
  - Optional
  - Client functionality can be extended by loadable code
  - Allows for lightweight clients
  - Current browser technology is based on this concept
    Example: JavaScript, Java etc.
- Disadvantages?
  - Reduced transparency



REST Architectural Elements

- REST as architecture style is not bound to any concrete implementation or specific protocol syntax
- **Resources** are central architecture elements
- **Metadata** is used to describe resources
- **Metadata** can be used to describe metadata – for example, to establish integrity of a message

- How resources can be accessed?

URIs as a Concept?

- URI for resource access

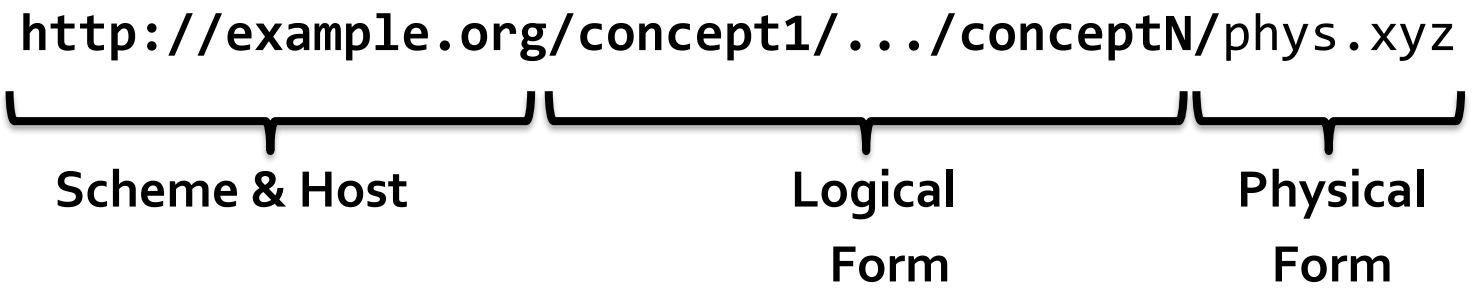
- What is the difference between
  - http://example.org/Zoo/Bears/Knut.php
  - http://example.org/Zoo/Bears/Tonka.html
  - http://example.org/cgi/Bears/Prob.Lembaer
- and
  - http://example.org/Zoo/Bears/Knut
  - http://example.org/Zoo/Baers/Tonka
  - http://example.org/Zoo/Baers/Prob.Lembaer

→ It's a manifestation of HTTP-URIs

URIs – Logical and Physical (2)

- Conceptual form of URLs

  Logical and physical form – Example:



- What kind of advantage do “logical" URLs offer ?
  - Find further aspects and examples for physical and logical forms
  - Advantages: reuse, evolution, independence
  - Example: Switching to another OS, another Web Server, different technology

URIs – Logical and Physical (3)

- **Examples…**
- Initial website
  - http://vsr.informatik.tu-chemnitz.de/people/gaedke.html
- Switch to PHP
  - http://vsr.informatik.tu-chemnitz.de/people/gaedke.php
- Switch to ASP
  - http://vsr.informatik.tu-chemnitz.de/people/gaedke.asp
- Switch to ASP.NET
  - http://vsr.informatik.tu-chemnitz.de/people/gaedke.aspx

- Neutral representation?
  - http://vsr.informatik.tu-chemnitz.de/people/gaedke.xml

URIs – Logical and Physical (4)

- Technology-neutral representation?
  - http://vsr.informatik.tu-chemnitz.de/people/gaedke
- …but who is actually now Gaedke?
  - http://twitter.com/gaedke
    - vs. http://vsr.cs.tu-chemnitz.de/people/gaedke
    - vs. http://…
  - …and what one gets from http://vsr…/gaedke
    - How does one get resources in HTML or XML or RDF?
    - ….or even as an image?

URIs – Logical and Physical (5)

- Approaches in context of "pretty" URIs
  - **Content Negotiation**
    - HTTP Accept attribute & HTTP Redirect / 303 Object moved
  - **URI Aliases**
    - RDF resources can be placed in relationships with others via owl:sameAs
  - **Associated Descriptions**
    - Resource delivers further descriptions, i.e. RDF or XLink

REST und URI

- Client references a resource
  - HTTP protocol is used
  - Referencing is made via (logical) URL
  - URIs are object identifiers

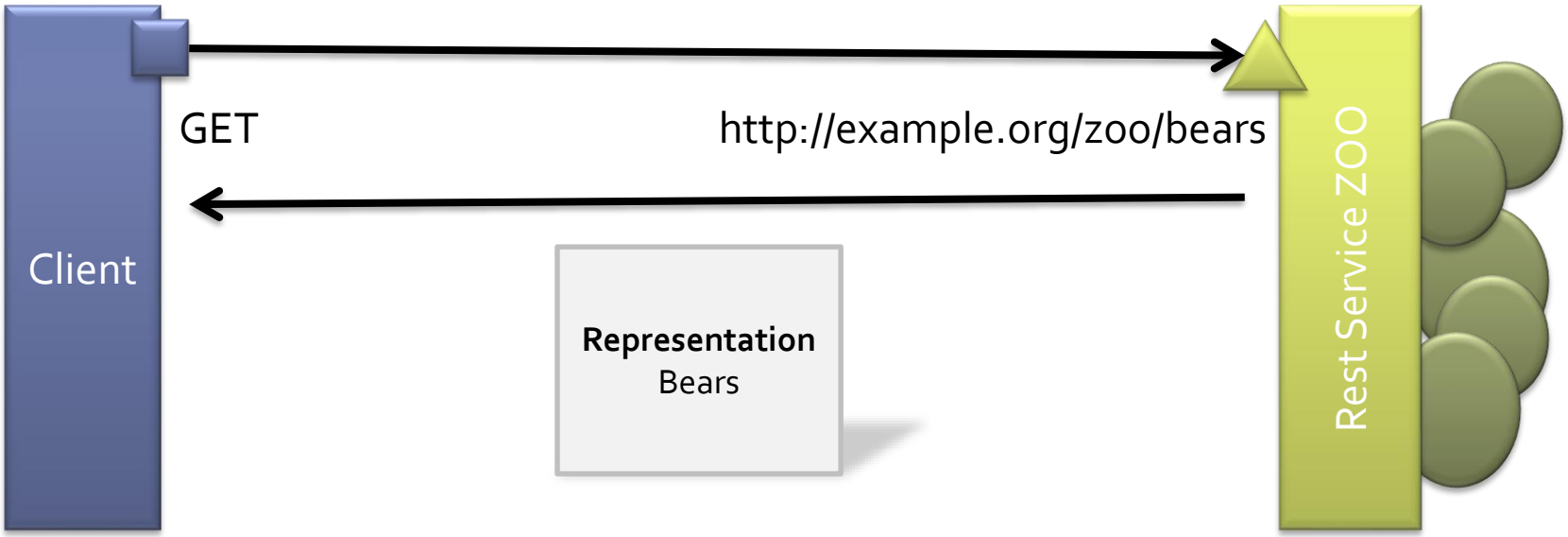
Representation, State & Transfer

- **Representation** – Sequence of bytes and an additional set of metadata which describes the byte sequence.
- Example: Representation of a resource to client
  - Here: a list (Bears.html – is ***one*** physical manifestation of the concept Bears of Zoos Example.org)
  - Client is now in a new **state**
  - Selection of links in Baerenliste.html ***transfers*** the client to a futher state
  - Note: Request/Response are stateless (no session)

ReST by Example: ZOO

- REST uses a simple CRUD-oriented interface, for instance:
  - C – Create via HTTP-POST
  - R – Read via HTTP-GET
  - U – Update via HTTP-PUT
  - D – Delete via HTTP-DELETE
- Note: How the functionality is realized by the REST service stays the service's secret (fosters loose coupling)
  - Representation (i.e. HTML or XML) can be negotiated

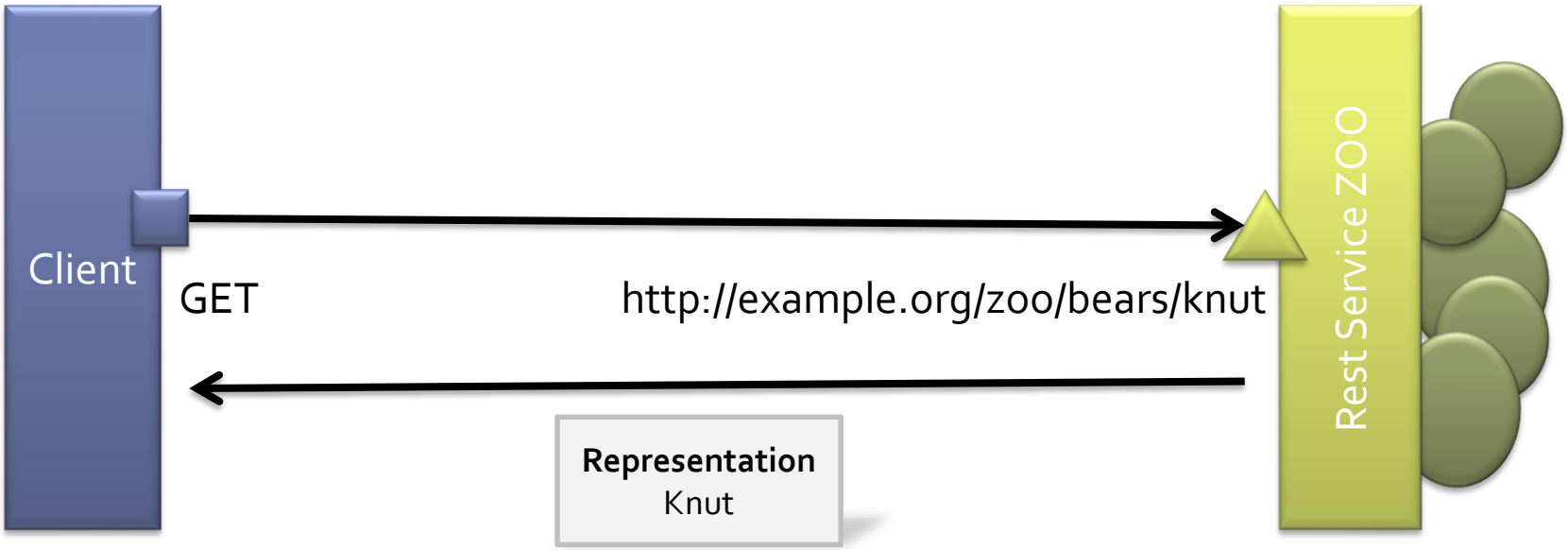
ReST by Example: ZOO (1)




```
<?xml version="1.0"?>
<zoo:Animals xmlns:zoo="http://example.org/animals"
             xmlns:xlink="http://www.w3.org/1999/xlink"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation= "http://example.org/animals http://example.org/animals.xsd">
   <zoo:Animal id="b10" xlink:href="http://example.org/bears/knut"/>
   < zoo:Animal id="b11" xlink:href=" http://example.org/bears/tonka"/>
   < zoo:Animal id="b23" xlink:href=" http://example.org/bears/prob.lembaer"/>
</zoo:Animals>
```

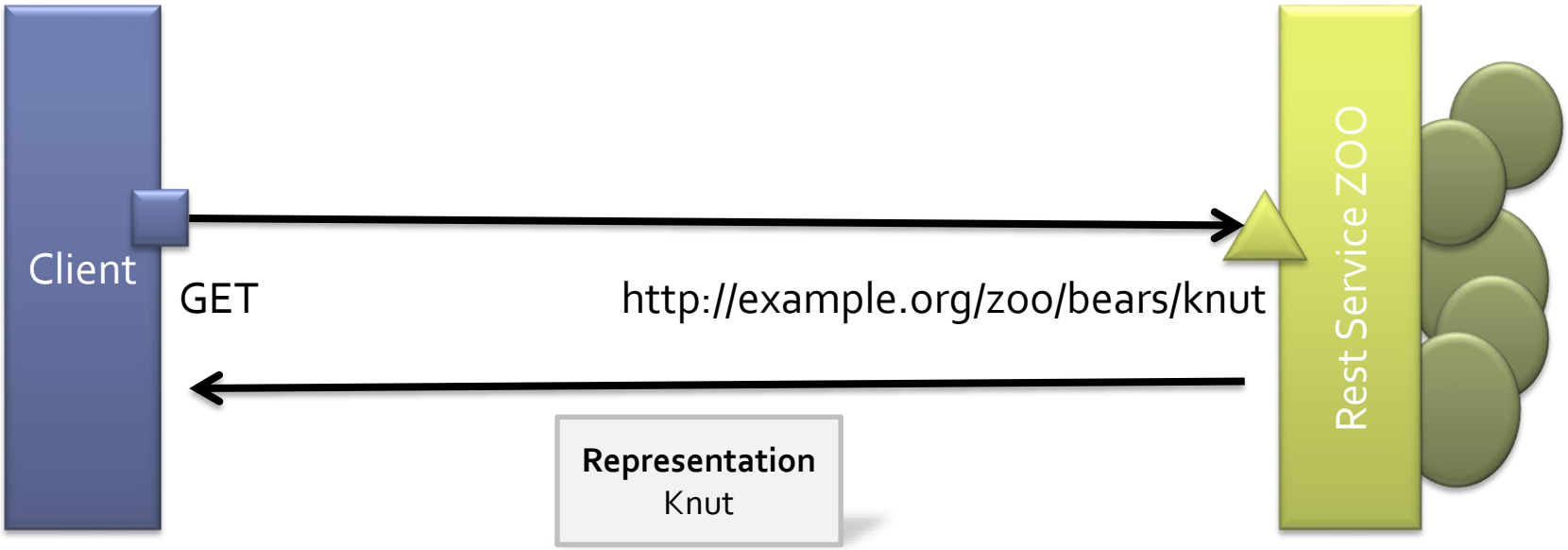
What happens if there are a lot of Bears? How can the set be limited?
For example, representation requires to fill in a search form.

ReST by Example: ZOO (2)




```
<?xml version="1.0"?>
<zoo:Bear   xmlns:zoo="http://example.org/animals"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation= "http://example.org/animals http://example.org/animals.xsd">
            <zoo:AnimalName>Knut</zoo:AnimalName>
            <zoo:AnimalAge>2</zoo:AnimalAge>
            <zoo:Description>Knut is a small friendly ice bear. Born and raised in....
            </zoo:Description>
</zoo:Bear>
```

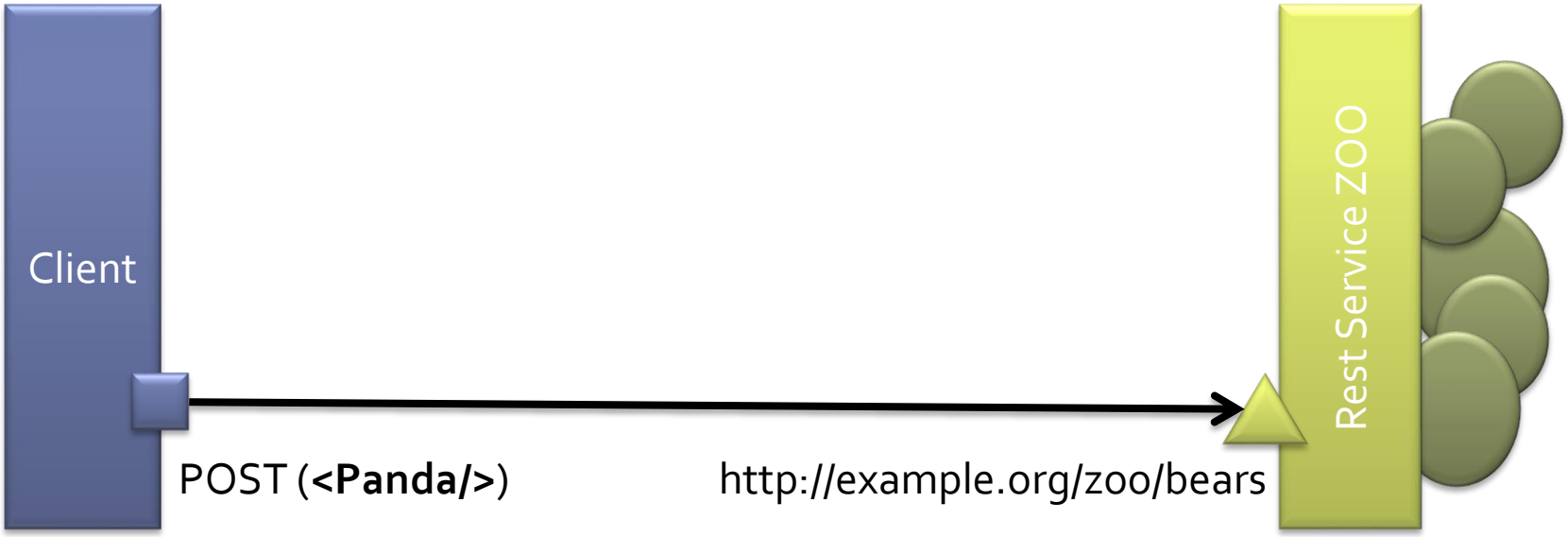
ReST by Example: ZOO (3)



What about other representations?
→ More extensive XML data about Knut
→ There could also be an image on that URL
→ Other formats, such as RDF

→Solution: Encode return format (physical form) in the request, for example: http://example.org/zoo/bears/knut?format=image

REST by Example: ZOO (4)



What can now be sent there and back?

Chapter 4

BEST PRACTICES

Best Practices for REST

- REST is not a strict standard, but an architectural style
  - → flexibility and freedom of structure
  - It requires to know and to understand the basic principles of the web
  - There is no right or wrong – just good or bad
- Target group: developers
- Design goal: minimize bad development choices

Keep simple things simple

- Minimise need for documentation by using intuitive interface
- Keep verbs out of your URLs

```
/getAllDogs
/verifyLocation
/giveDirectOrder
```

```
/dogs
/locations
/orders
```

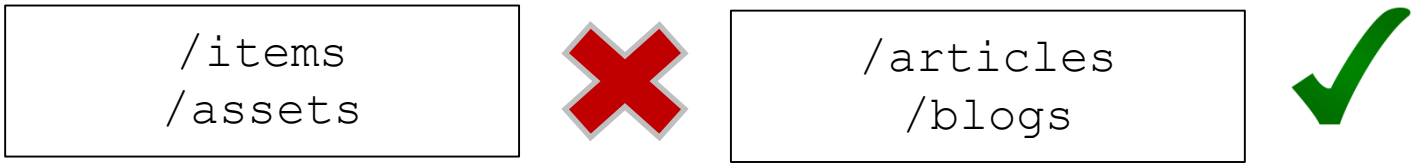
- Use plural and singular nouns consistently

```
/checkins
/Deals
/Product
```

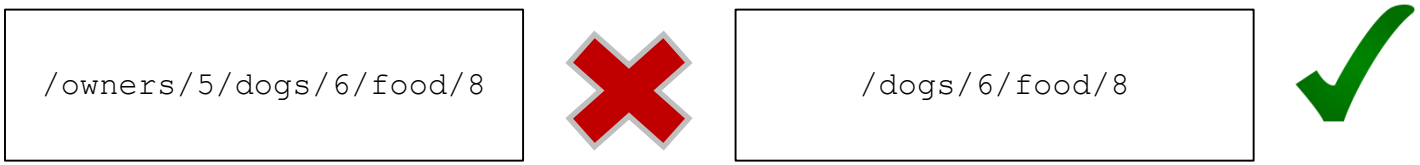
```
/checkins
/deals
/products
```

Keep simple things simple

- Concrete names are better than abstract



- Avoid URLs with deep associations



- Sweep complexity behind the „?“

```
GET /dogs?color=red&state=running&location=park
```

Handling Errors

- Use HTTP status codes:
  - Everything worked – success (2xx)
  - The client did something wrong – client error (4xx)
  - The API did something wrong – server error (5xx)
- Make messages returned in the payload as verbose as possible

```
401 Unauthorized

{"developerMessage" : "Verbose, plain language description
of the problem for the app developer with hints about how
to fix it.", "userMessage":"Pass this message on to the
app user if needed.", "errorCode" : 12345, "more info":
"http://dev.teachdogrest.com/errors/12345"}
```

VSR | EDU

Software Service Engineering

Software Service Engineering

Prof. Dr.-Ing. Martin Gaedke
Technische Universität Chemnitz
Fakultät für Informatik
Professur Verteilte und selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

Handling Errors

- If clients can not handle HTTP codes:
  - Use suppress_response_codes = true

```
/public_timelines.json?
suppress_response_codes=true

HTTP status code: 200 {{response_code" : 401,
"error":"Could not authenticate you.“,…}
```

  - Always return OK
  - Push any response code that we would have put in the HTTP response down into the response message

Versioning

- Never release an API without a version and make the version mandatory
- Specify the version with a 'v' prefix. Move it all the way to the left in the URL so that it has the highest scope (e.g. `/v1/dogs`).
- Maintain at least one version back
- Give developers at least one cycle to react before obsoleting a version (the cycle length depends on the type of application, consumers etc.)

Pagination and partial response

- Partial response allows providing only the required information → add optional fields in a comma-delimited list:

```
/dogs?fields=name,color,location
```

- Make it easy for developers to paginate objects in a database:

```
/dogs?limit=25&offset=50
```

- Include metadata (e.g. total number of results) into the response

Function Calls

- Some APIs may send a response which is not a resource:

```
/calculate
/translate
/convert
```

- Use verbs not nouns:

```
/convert?from=EUR&to=CNY&amount=100
```

- Document these “non-resource” scenarios

Message formats

- Support more than one format
- Specify desired format using URL or Accept-Header

```
/venue.json OR
/venue | Accept: application/json
```

- Use Javascript name conventions while naming the JSON attributes

```
/"created_at": "…"
 /"DateTime": "…"
```

```
/"createdAt": „…"
```

Chapter 5

DESCRIBING REST SERVICES

Web Application Description Language (WADL)

- WADL is designed to provide a machine process-able description of HTTP-based Web applications.
- Web applications:
  - Are based on existing Web architecture and infrastructure
  - Are platform and programming language independent
  - Promote re-use of the application beyond the browser
  - Enable composition with other Web or desktop applications
  - Require semantic clarity in content (representations) exchanged during their use

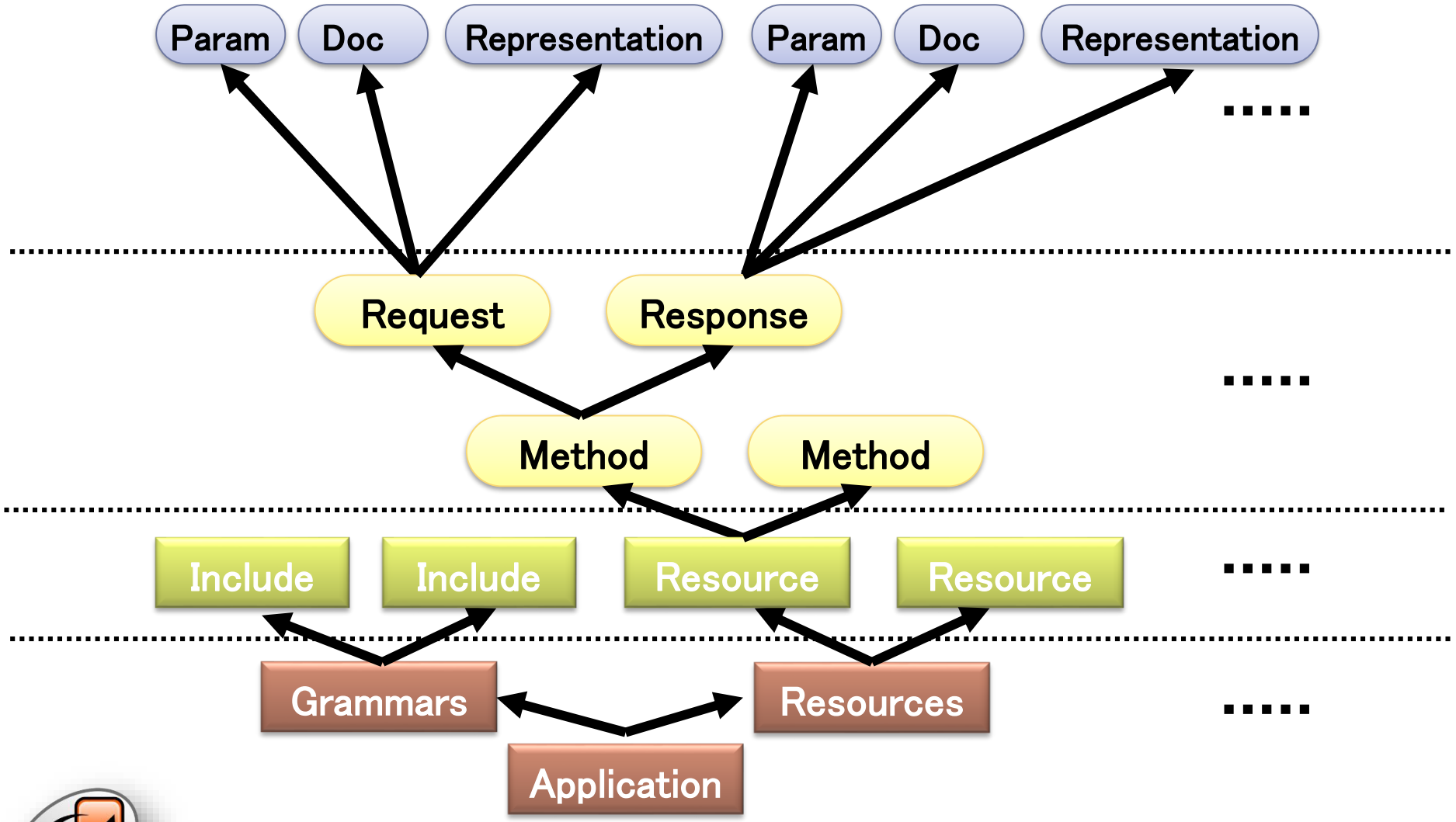
WADL – Use Cases

- Application Modelling and Visualization
  - Support for development of resource modelling tools for resource relationship and choreography analysis and manipulation.
- Code Generation
  - Automated generation of stub and skeleton code and code for manipulation of resource representations.
- Configuration
  - Configuration of client and server using a portable format.

WADL – Vocabulary (1)

- **Grammars** contains definitions of the format of data exchanged during execution.
- An **application** is made up of various **resources.**
- A **resource** is described by a **method** which containes the corresponding HTTP-Verb and some information about the **request** and **response.**

WADL – Vocabulary (2)



WADL - Example

```
<?xml version="1.0"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
 xsi:schemaLocation="http://wadl.dev.java.net/2009/02
wadl.xsd"
 xmlns:tns="urn:yahoo:yn"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:yn="urn:yahoo:yn"
 xmlns:ya="urn:yahoo:api”
 xmlns=http://wadl.dev.java.net/2009/02>
  <grammars>
    <include
      href=“NewsSearchResponse.xsd”/>
    <include
      href=“Error.xsd”/>
  </grammars>

  <resources
base=“http://api.search.yahoo.com/NewsSearchService/V1/">
    <resource path="newsSearch">
      <method name="GET" Id="search">
        …
      </method>
    </resource>
  </resources>

</application>
```

```
<request>
  <param name="appid" type="xsd:string"
    style="query" required="true"/>
  <param name="query" type="xsd:string"
    style="query" required="true"/>
  <param name="type" style="query" default="all">
    <option value="all"/>
    <option value="any"/>
    <option value="phrase"/>
  </param>
  <param name="results" style="query" type="xsd:int"
default="10"/>
  <param name="start" style="query" type="xsd:int"
default="1"/>
  <param name="sort" style="query" default="rank">
    <option value="rank"/>
    <option value="date"/>
  </param>
  <param name="language" style="query" type="xsd:string"/>
</request>
<response status="200">
  <representation mediaType="application/xml"
    element="yn:ResultSet"/>
</response>
<response status="400">
  <representation mediaType="application/xml"
    element="ya:Error"/>
</response>
```

Do we need WADL?

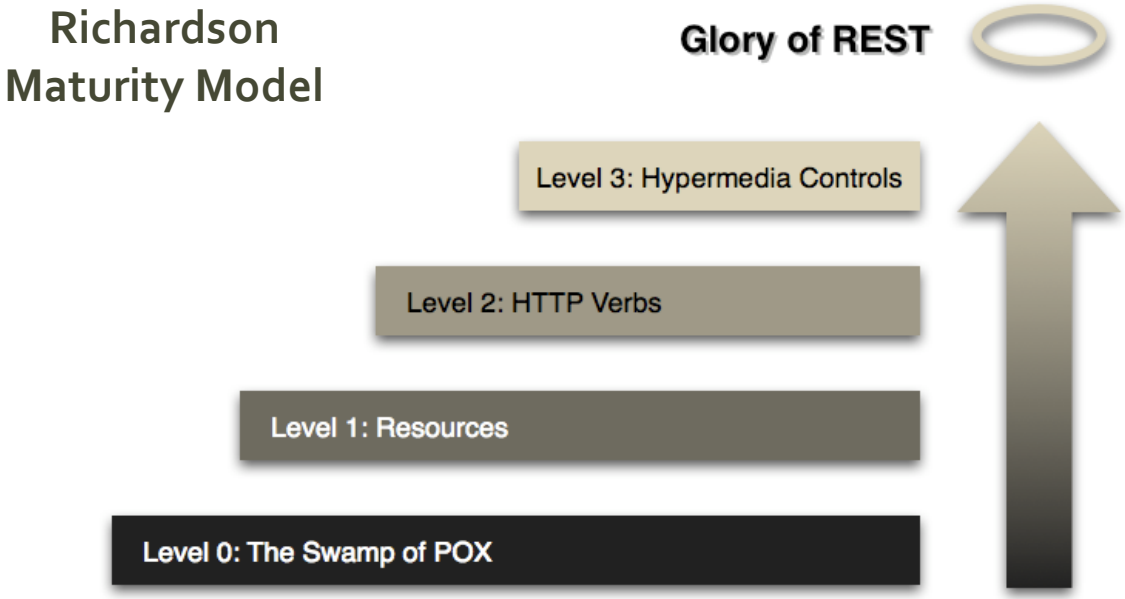
- WSDL 2.0 also describes RESTful Services.
- Human readable descriptions are also used very often. (e.g. Twitter-API)

- **BUT:** RESTful Services describe themselves
  - CRUD – Operations
  - Mime-Type
  - Static contract of WADL vs. dynamic discovery of HTTP

Chapter 6
SUMMARY

Is REST = REST?

- REST-Principles are hard to be compliant with. At least it is difficult – as no tool tells you
- Typical implementation approaches:



Quelle: http://martinfowler.com/articles/richardsonMaturityModel.html

REST Service Final Remarks

- REST – Encoding
  - Is not needed, one uses encoding standards (i.e. HTML, XML and othr MIME types)
- Service description
  - WSDL 2.0, WADL, Mime-Type
- Design aspects
  - Sessions are no longer necessary, no cookies needed, each request is authenticated
  - GET URI – must be free of side effects, i.e. actions/method names do not belong in the URI
- How does a REST service behave in its environment?
  - Consider caching, proxies, gateways
  - Fastest URI access is a network-less one (if the resource is already in cache)

REST Service Final Remarks

- REST services and SOAP-based services?
- Discussion:
  - What are the differences?
  - Dis-/advantages?

- Examples/Demos
  - AWS
  - Google
  - Yahoo!
  - WebComposition/DGS
  - …and many more…

- What about you? Which Web Service can you offer?
  - What is the impact of Web Services? For instance, on society, motivation from a business perspective?

Part V

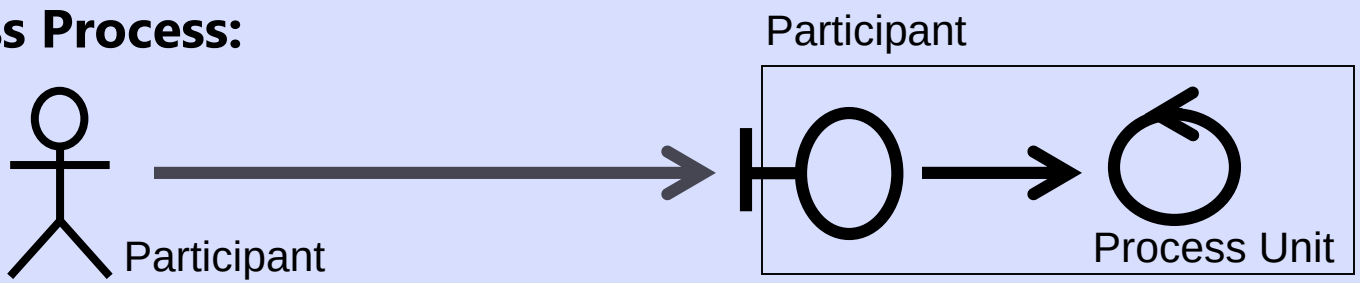
SERVICES / BUSINESS PROCESS

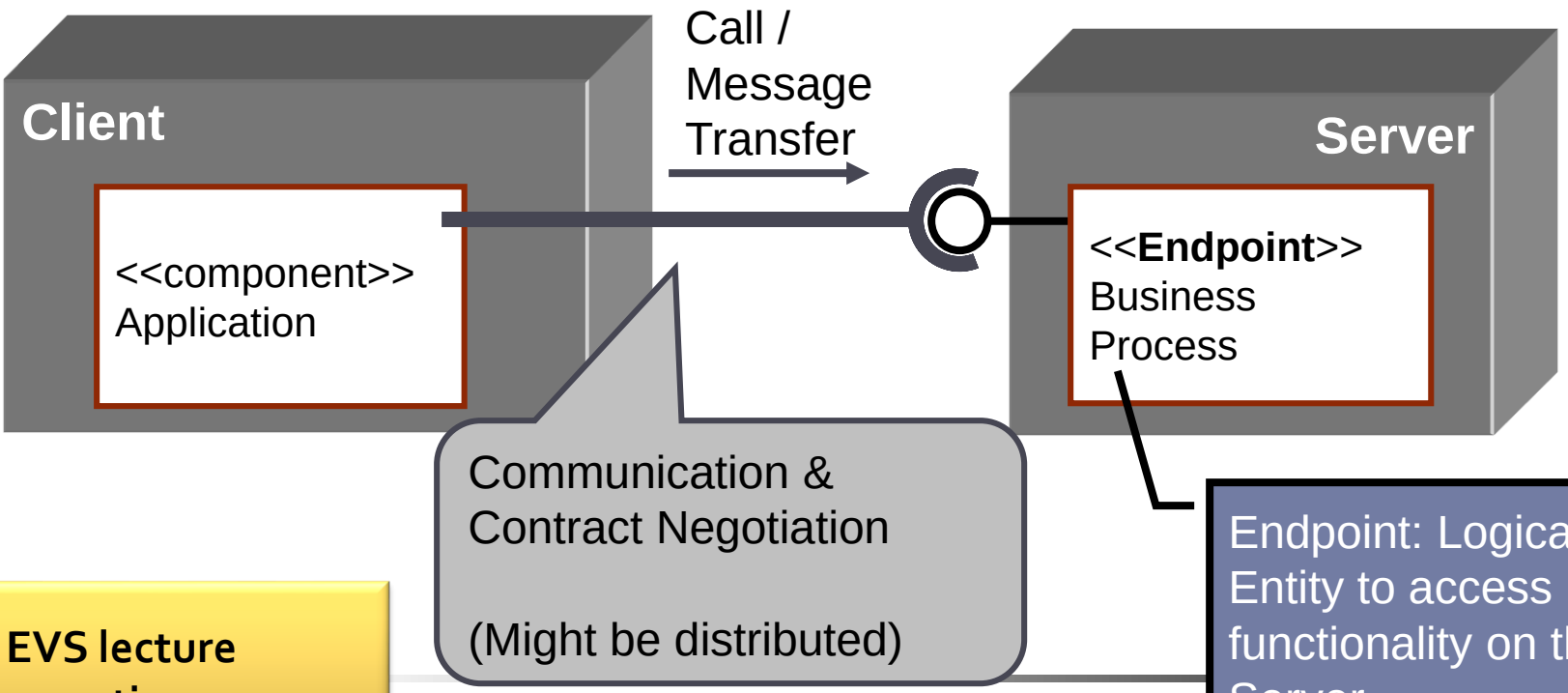
Chapter 1

MODELLING TECHNIQUES

Business Process Deployed

- **Business Process:**



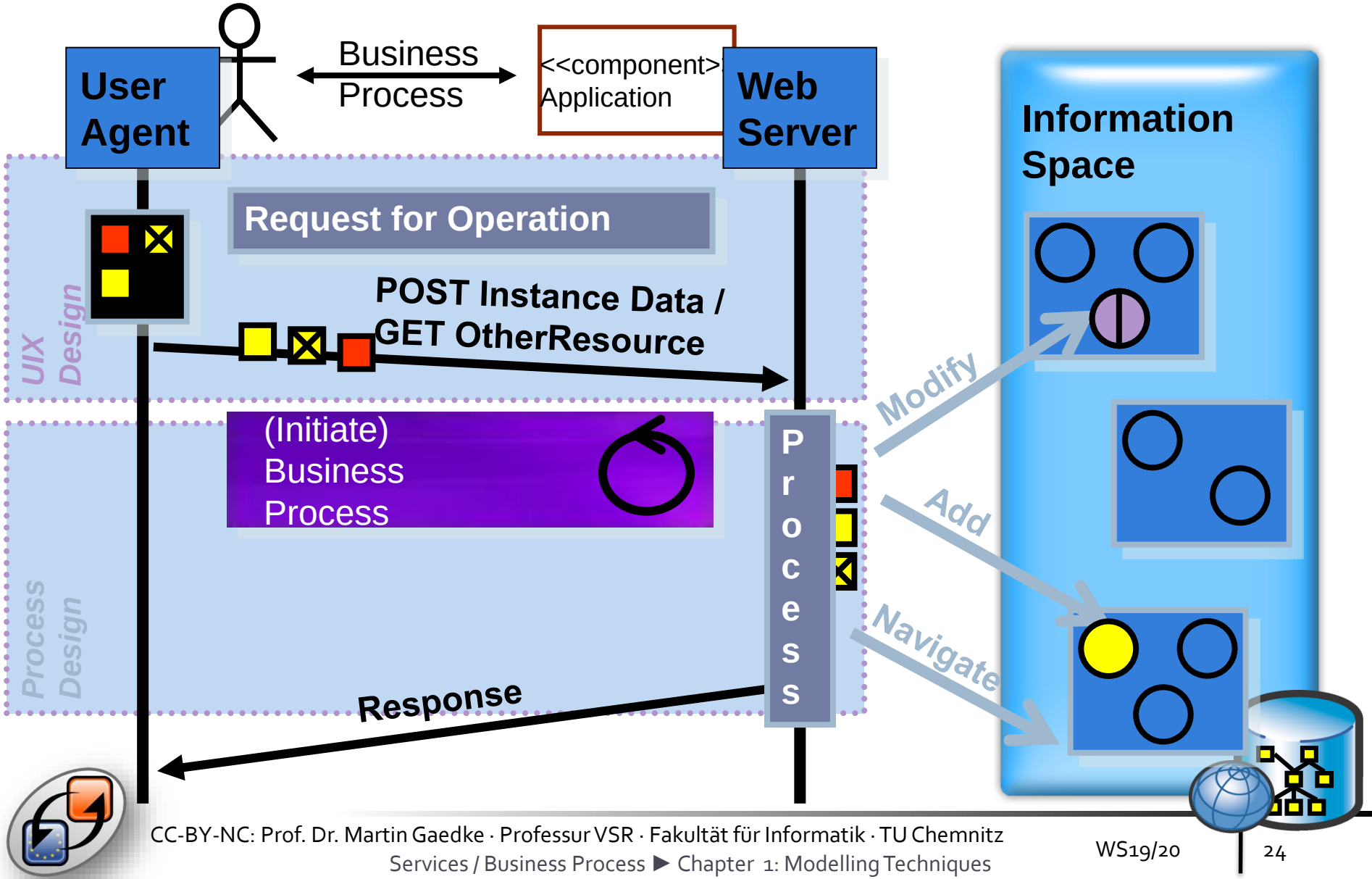


From EVS lecture for preparation

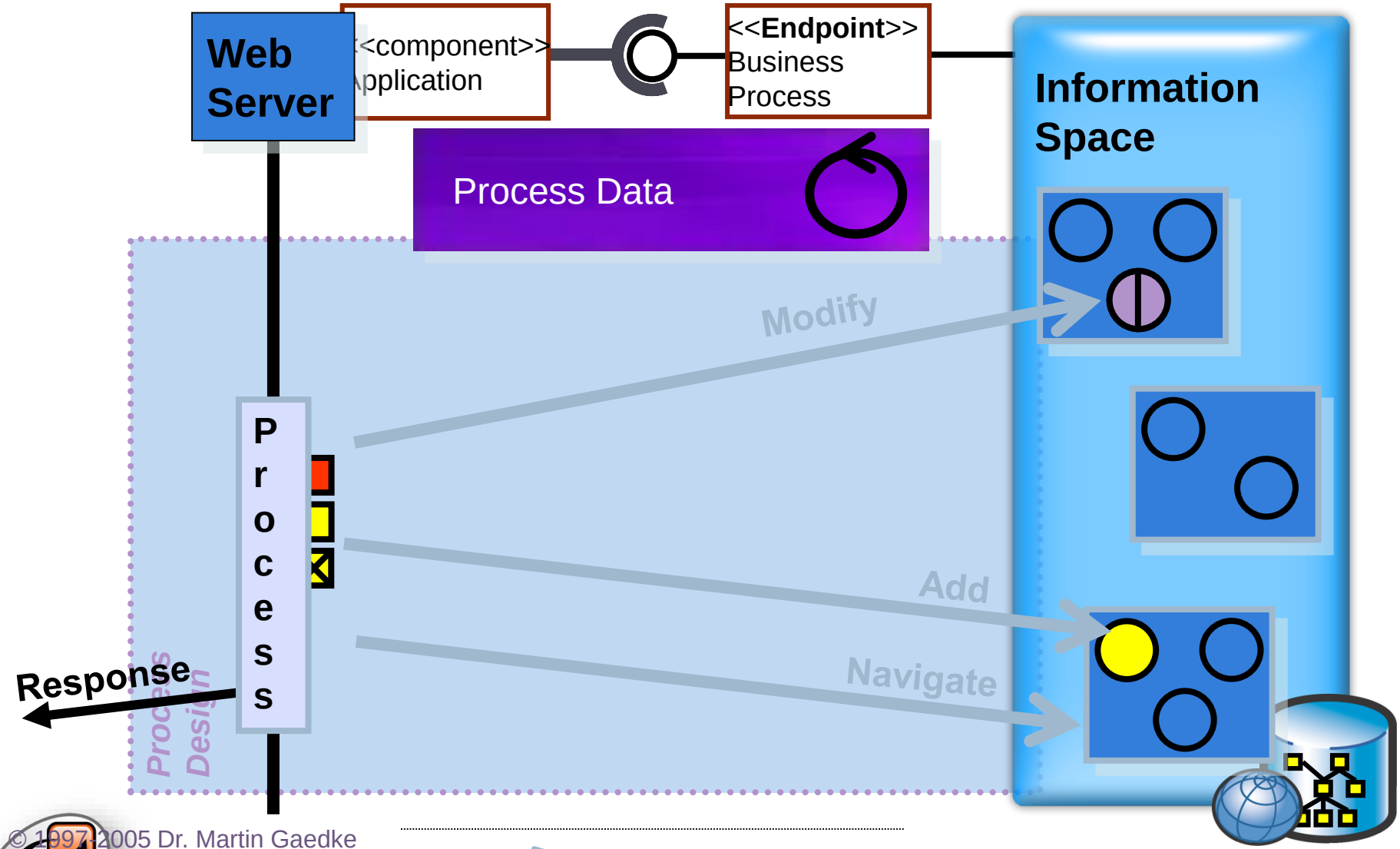
Process Unit

- Process Unit (Process Component)
  - Functional representation of a participant of the business process
  - A unit accessible as Endpoint within the process layer (and usually provided by several endpoints of the service layer)
- Modeling focuses on wiring of process units (respectively their endpoints)
  - Relationships between endpoints are expressed in terms of agreed upon communication patterns
  - Behavior of endpoints is described in abstract terms (Input-/Output-Contracts)

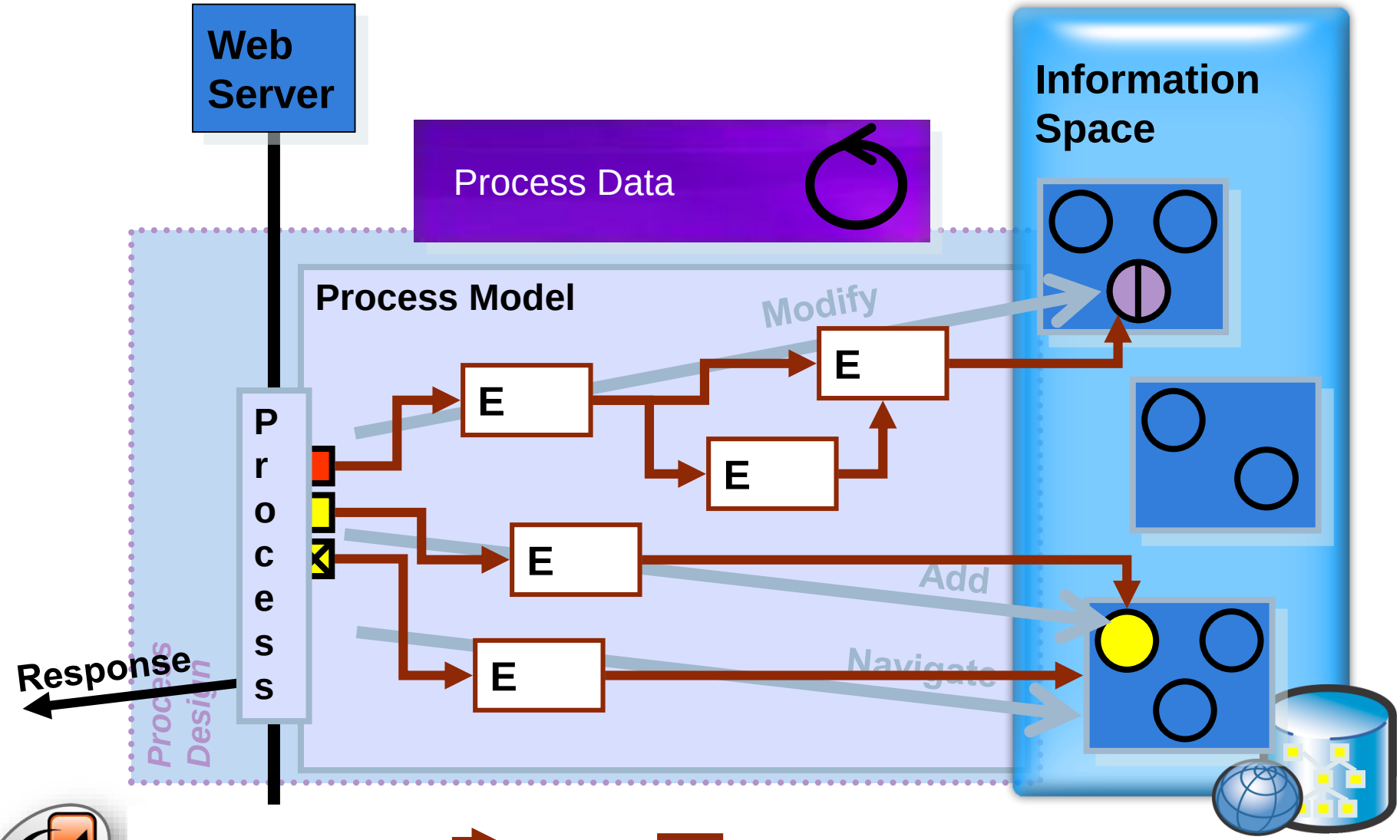
Process and the Web



Process Model and the Web



Process Model in Detail



Tasks to be addressed

- Modelling Techniques (Basics)
  - BPMN – The core of BPMN (especially the graphic Notation - BPD)
    - http://www.bpmn.org/
    - http://www.omg.org/spec/BPMN/1.2/PDF/
    - Not BPMN 2.0
  - WAM
    - Cf. Lecture SVS
    - http://webcomposition.org

Chapter 2

BUSINESS PROCESS MODEL AND NOTATION (BPMN)

What is a Business Process?

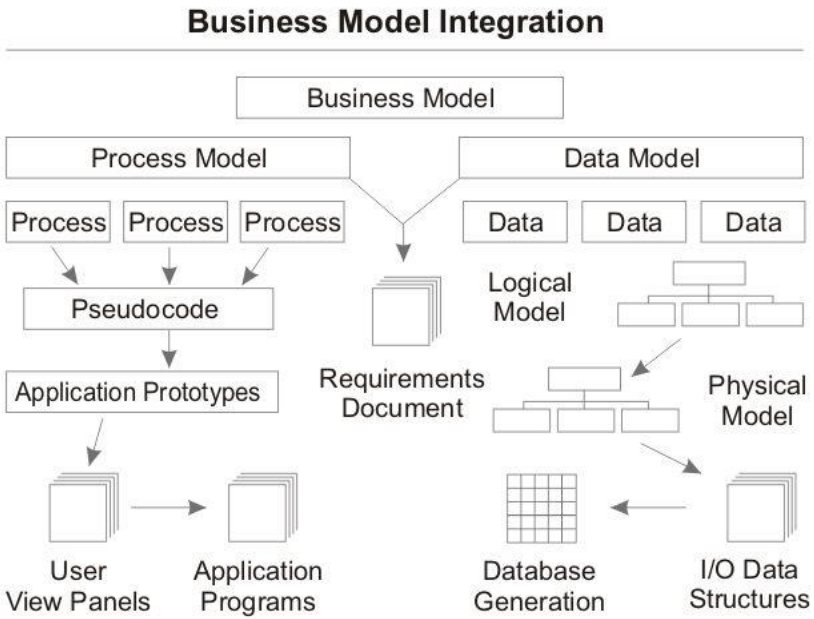
- …a structured, measured set of activities designed to produce a specific output for a particular customer or market… A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs

Thomas Davenport (1993). Process Innovation: Reengineering work through information technology

Paul R. Smith, Marcel Douwe Dekker. Creating a strategic plan for configuration management using Computer Aided Software Engineering (CASE) tools

Business Process Modeling

- Business process models describe how a business works, or more specifically, how they accomplish missions, activities, or tasks



Paul R. Smith, Marcel Douwe Dekker. Creating a strategic plan for configuration management using Computer Aided Software Engineering (CASE) tools

Business Process Modeling

- Benefits:
  - Planning
  - Continuous improvement
  - Knowledge retention and learning
  - Visualization
  - Framework for metrics
  - Compliance, audit, assessments
  - Execution

Business Process Modeling Methods

- Flow Charts
- Petri Net
- Gantt / PERT Diagrams
- UML Activity Diagrams
- Business Process Model and Notation (BPMN)
- Business Process Execution Language (BPEL)
- …

BPMN: Goals and differentiators

- Provides an understandable notation for all business users
- Bridges the gap between business process design and implementation
- Visualizes XML languages designed for execution of business processes (particularly BPEL) with a business-oriented notation
  - Business Process Model and Notation (BPMN)
  - Latest version can be found here: https://www.omg.org/spec/BPMN/

Topics out of Scope

- Organizational sctructures
- Functional breakdowns
- Data and information models
- Strategy
- Business Rules

Uses of BPMN

- Private (Internal) Business Processes
- Abstract (Public) Business Processes
- Collaboration (Global) Business Processes

Private Business Processes

- Workflows internal to a specific organization

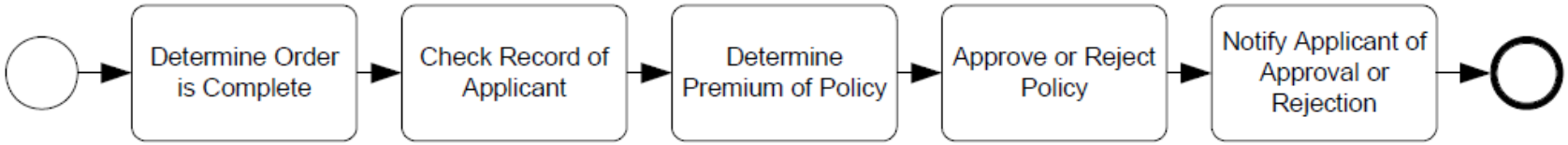


Image: Business Process Model and Notation (BPMN) 1.2. Specification. http://www.omg.org/spec/BPMN/1.2/ (Last Access 09.01.2011)

Abstract (Public) Processes

- „Internal" activities of the process are not shown
- Focus on interactions between several participants

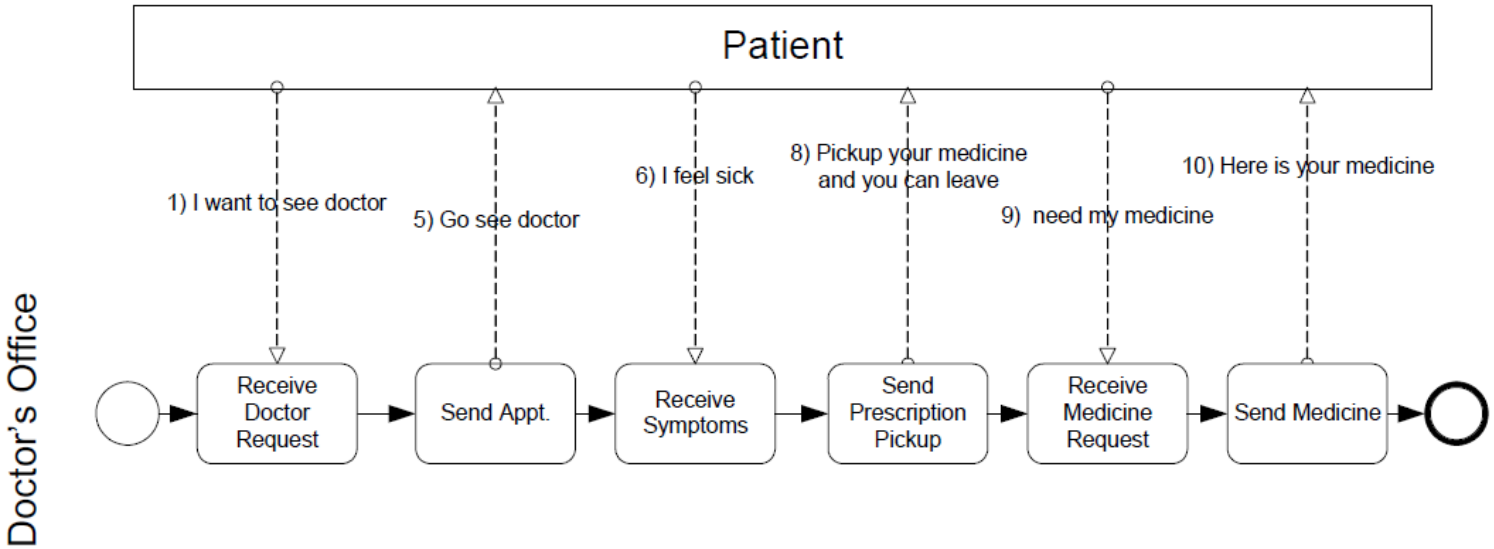


Image: Business Process Model and Notation (BPMN) 1.2. Specification. http://www.omg.org/spec/BPMN/1.2/ (Last Access 26.01.2011)

Collaboration Processes

- Interaction between one or more business entities

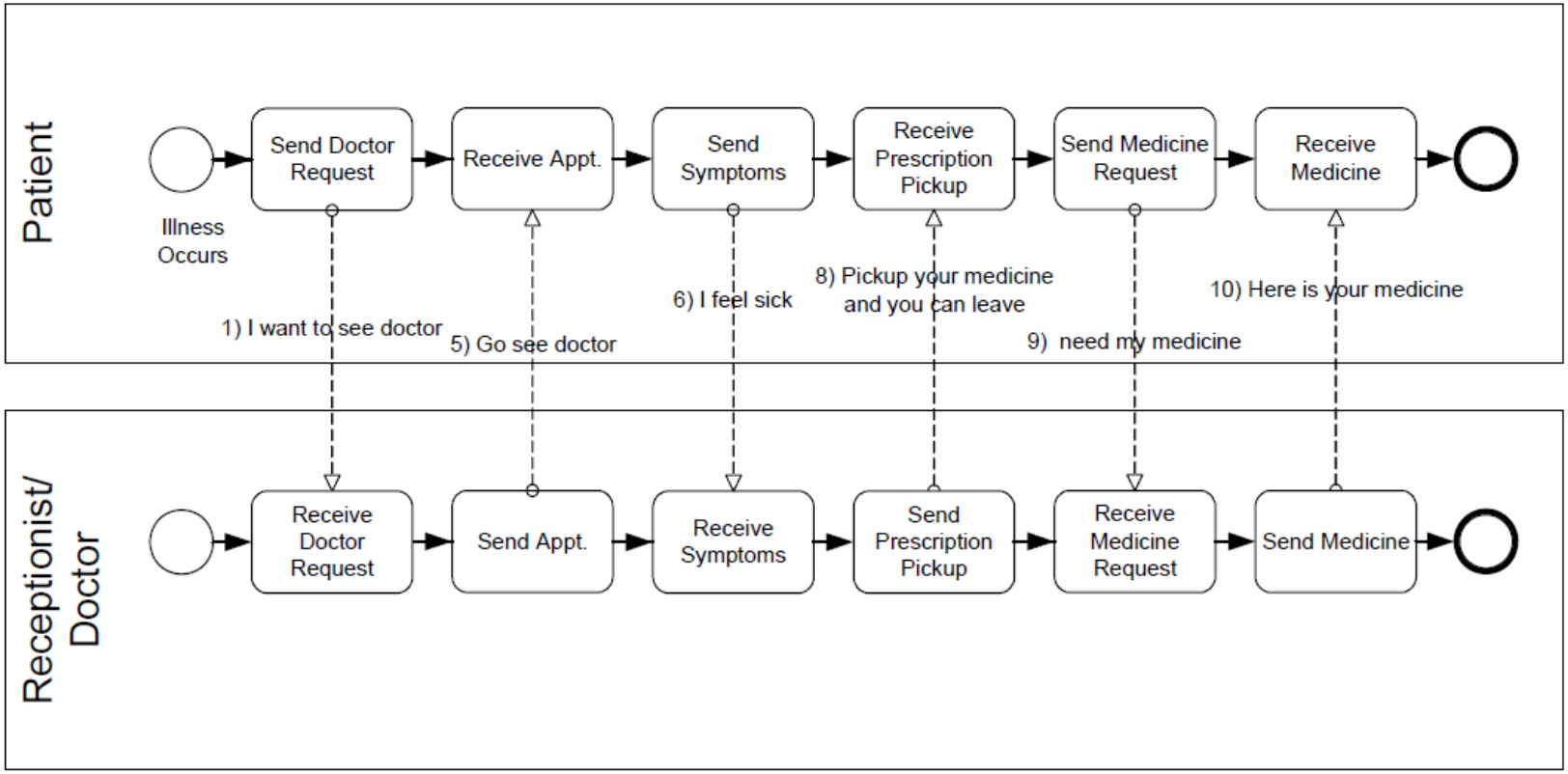
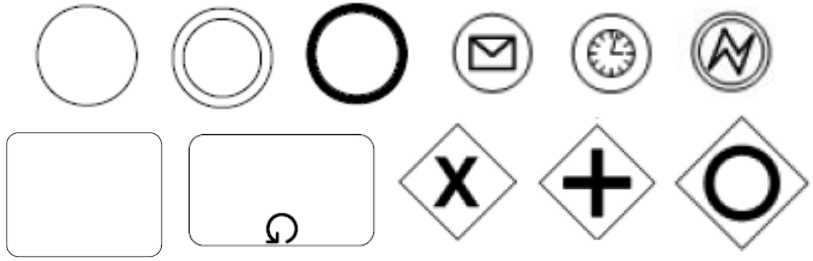
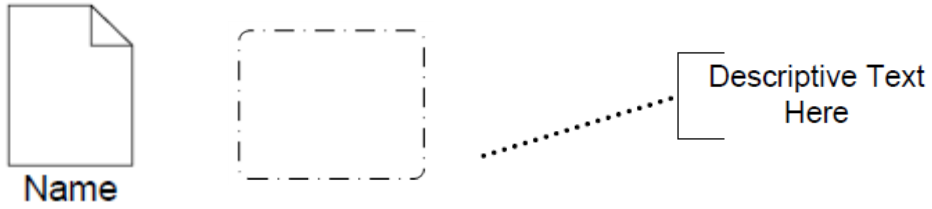


Image: Business Process Model and Notation (BPMN) 1.2. Specification. http://www.omg.org/spec/BPMN/1.2/ (Last Access 26.01.2011)

Business Process Diagram - Overview

- Flow Objects
  - Events
  - Activities
  - Gateways
- Connecting Objects
  - Sequence Flow
  - Message Flow
  - Association
- Swimmlanes
  - Pools
  - Lanes
- Artifacts
  - Data Object
  - Group
  - Annotation

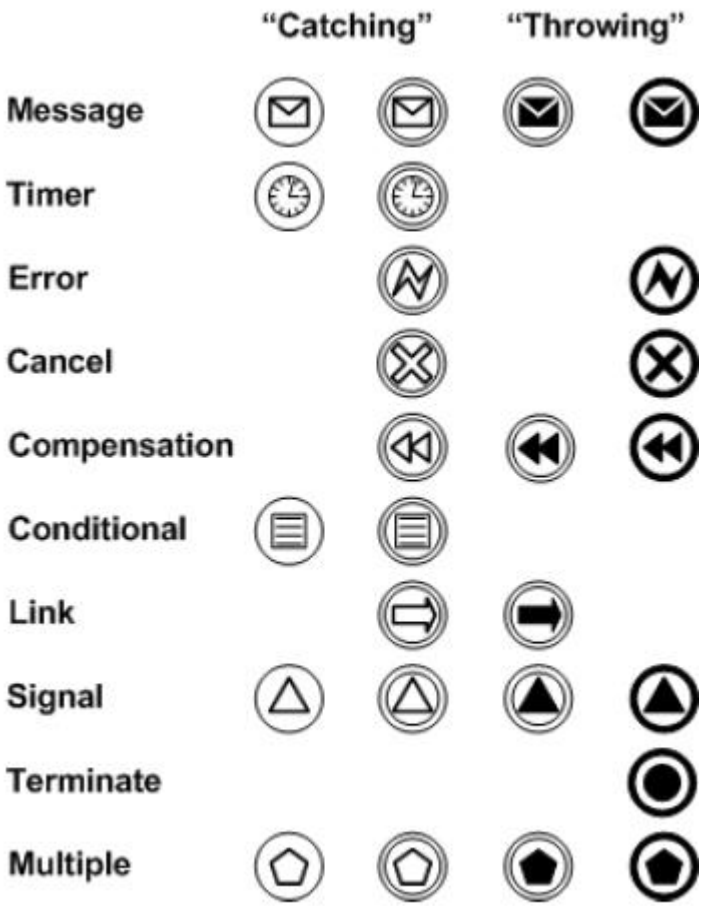




Images: Business Process Model and Notation (BPMN) 1.2. Specification. http://www.omg.org/spec/BPMN/1.2/ (Last Access 26.01.2011)

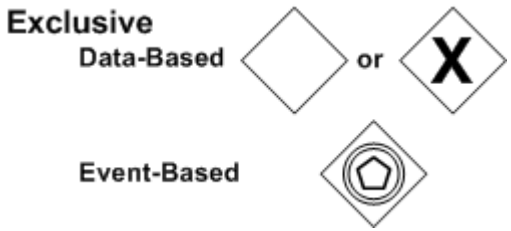
Flow Objects: Events

- „Triggers“ define the cause of the event (Message, Signal,Timer,Exception)
- Start Events can only „catch“ a trigger
- End Events can only „throw“ results
- Intermediate Events can do both

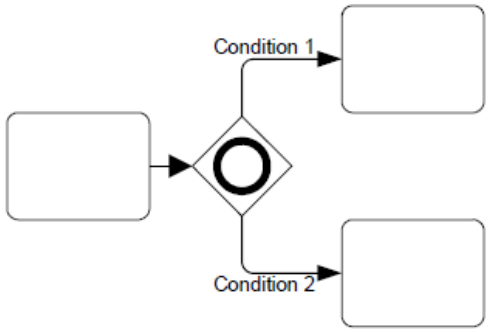


Flow Objects: Gateways

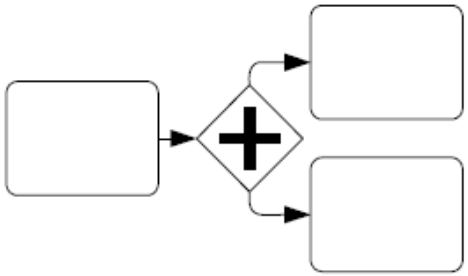
- An **Exclusive Gateway (XOR-split)** restricts the flow such that only one of a set of alternatives may be chosen during runtime



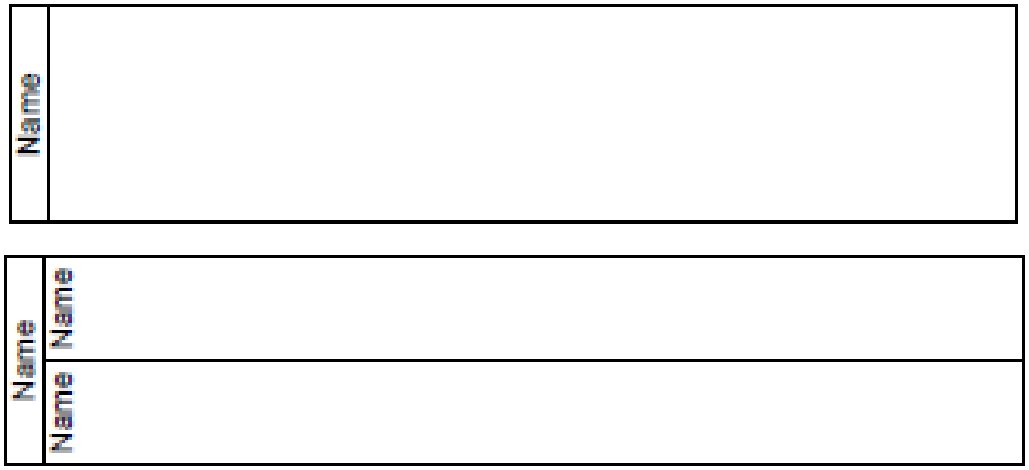
- An **Inclusive Gateway (OR-split)** represents a branching point where Alternatives are based on conditional expressions contained within outgoing Sequence Flow. All Sequence Flow with a True evaluation will be traversed by a Token



- A **Parallel Gateway (AND-split)** divides a path into two or more parallel paths, which enables activities to be performed concurrently



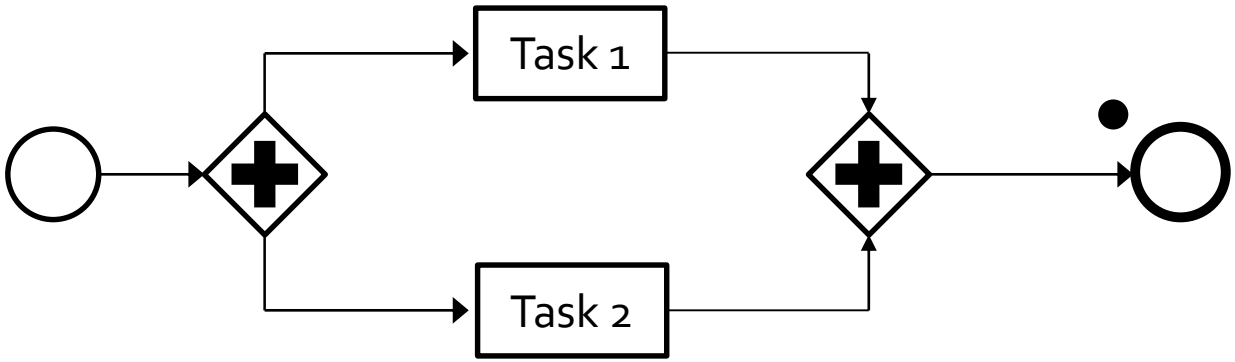
Swimlanes



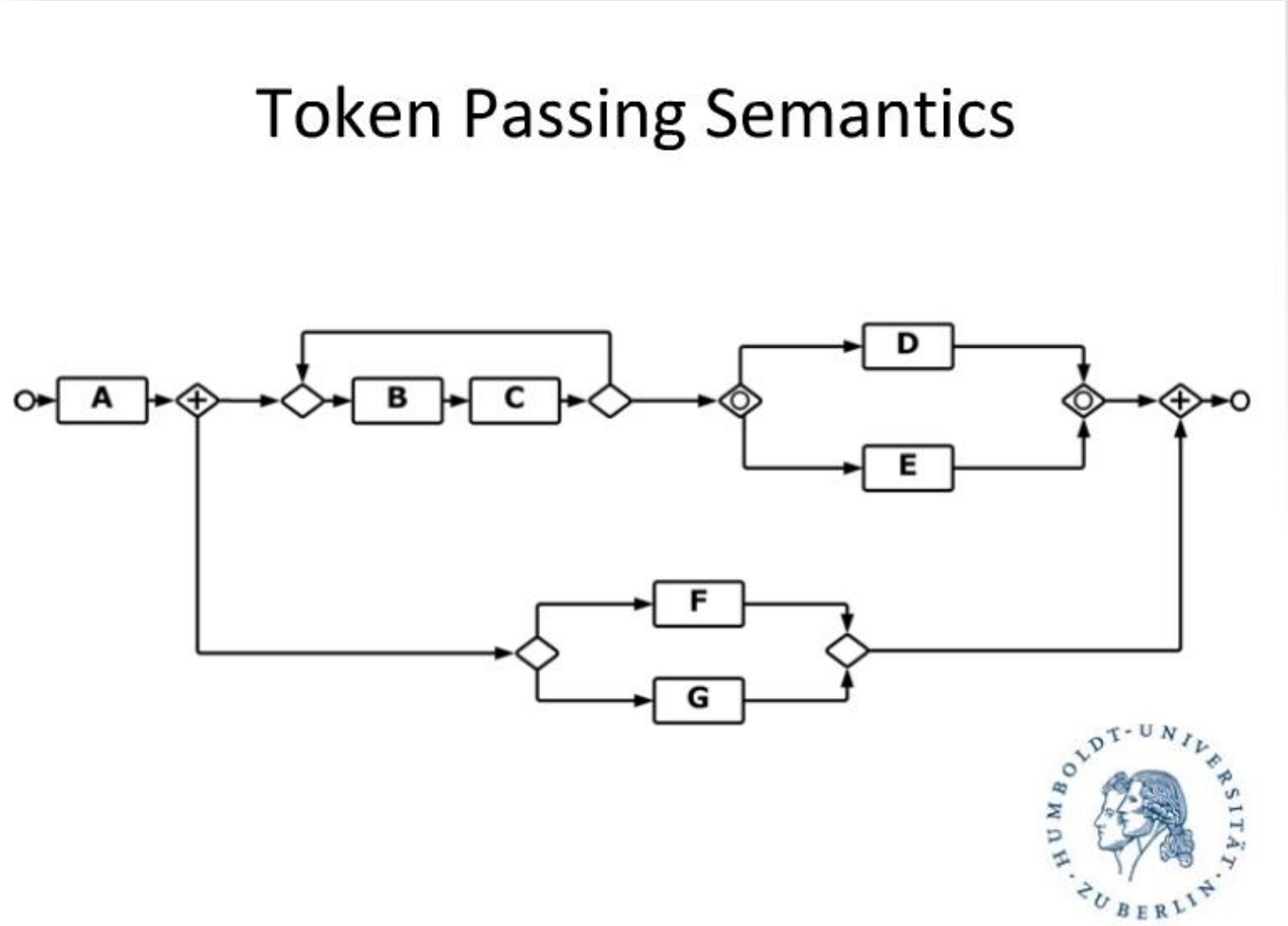
- A **Pool** represents a Participant in a Process
- A **Lane** is a sub-partition within a Pool. Lanes are used to organize and categorize activities within a Pool . The meaning of the Lanes is up to the modeler

Token Passing

- Control object traversing the sequence flow
- Positions of tokens define process state
- Start event generates a token, which is consumed by end event
- Gateways may divide or merge tokens
- Intermediate events may delay the token

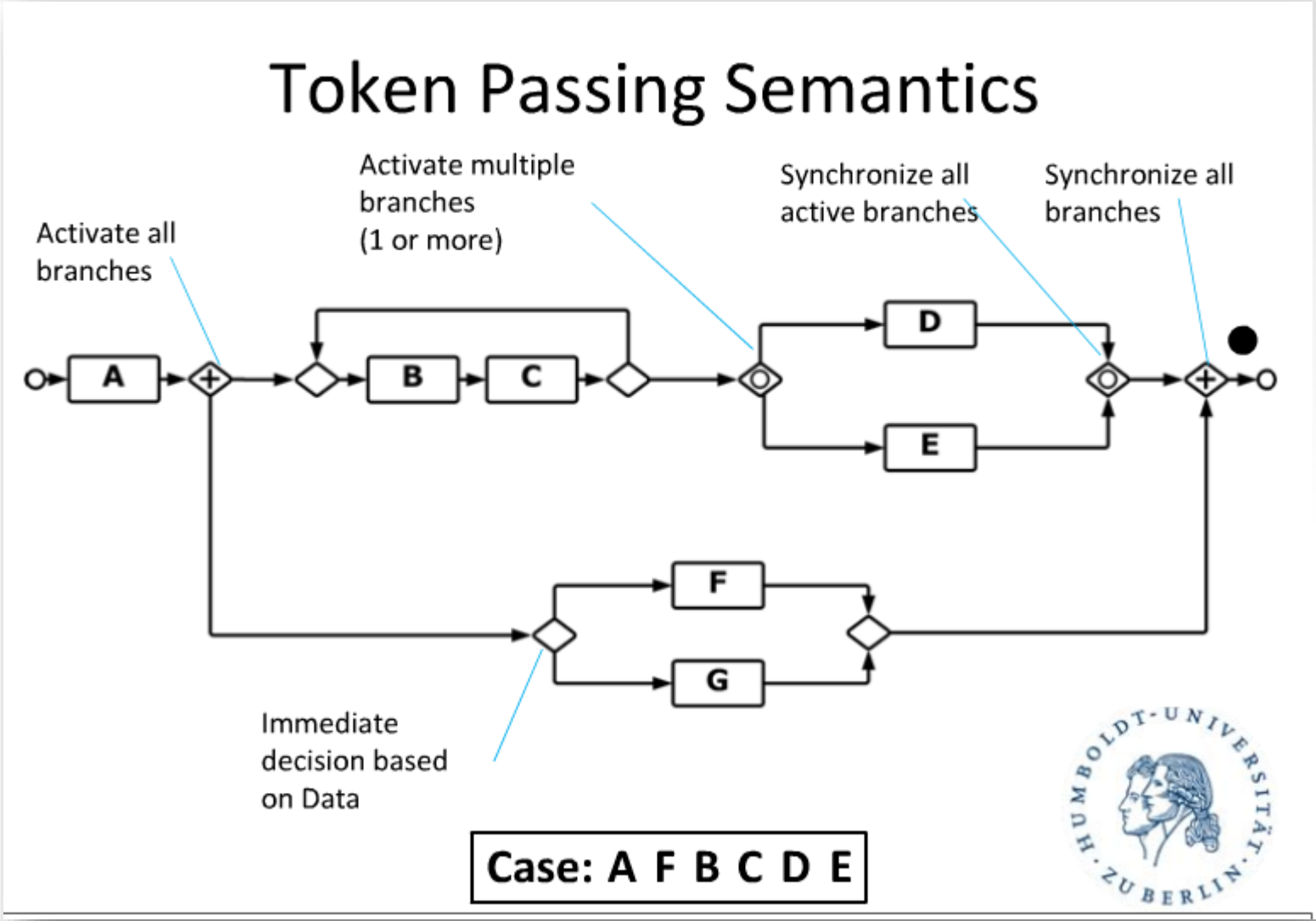


A simple BPMN workflow… simple?

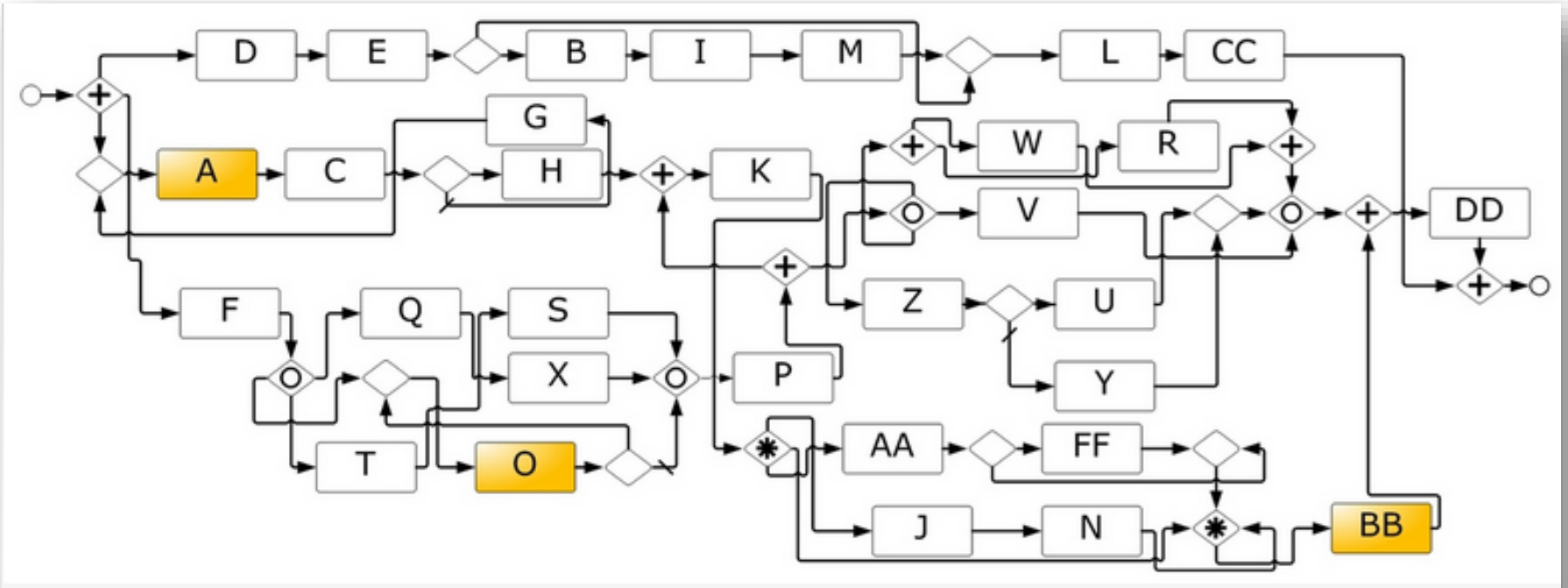


BPMN Self-Test by HU Berlin (no longer exists)

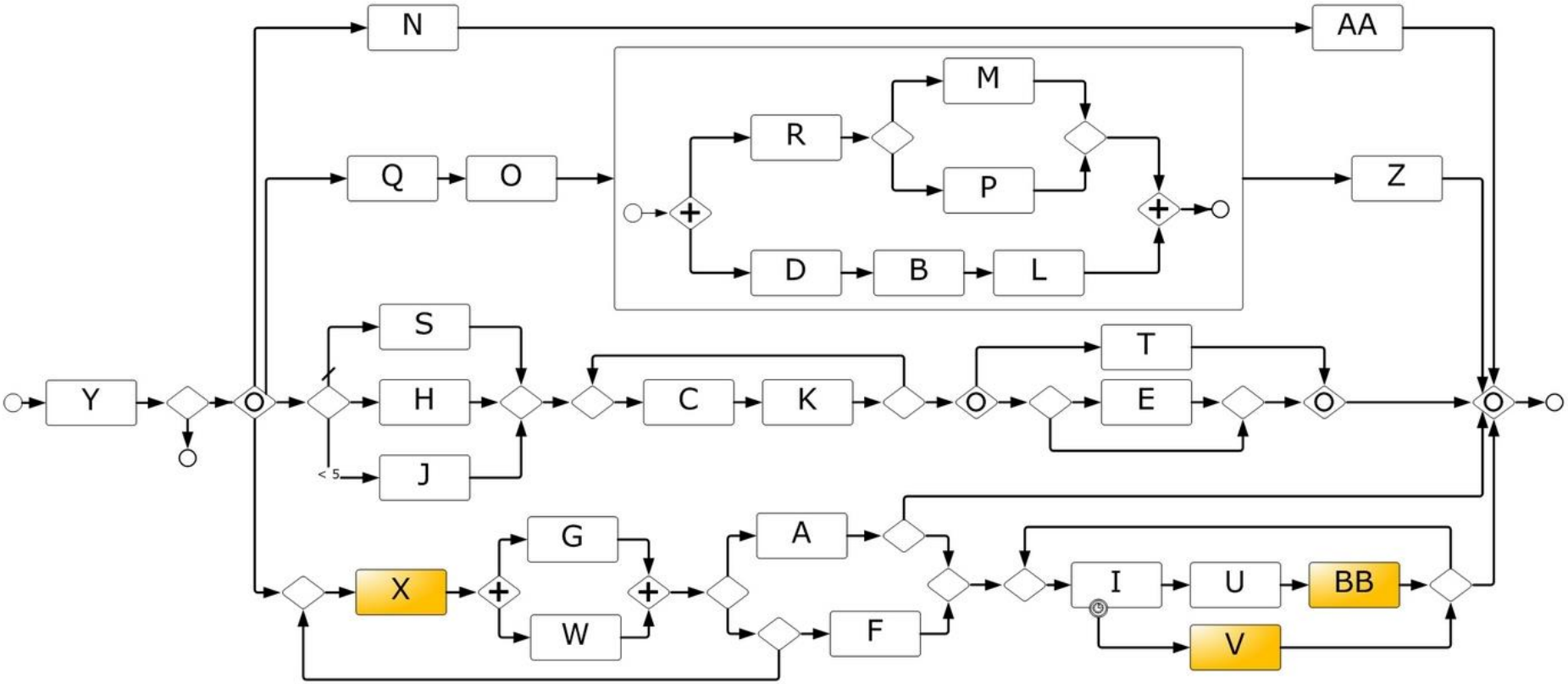
A simple BPMN workflow… simple?



Could you answer this one?



Could you answer this one?



If X is executed for a case, then both BB and V can be executed for the same case.

Wahr oder Falsch?

Some typical BPMN Editors

- http://www.ariscommunity.com/aris-express/download
- https://camunda.com/products/modeler/