Die Entwicklung von SOAP-Webservices mit Java 6

Aktuelle Technologien zur Entwicklung verteilter Java Anwendungen

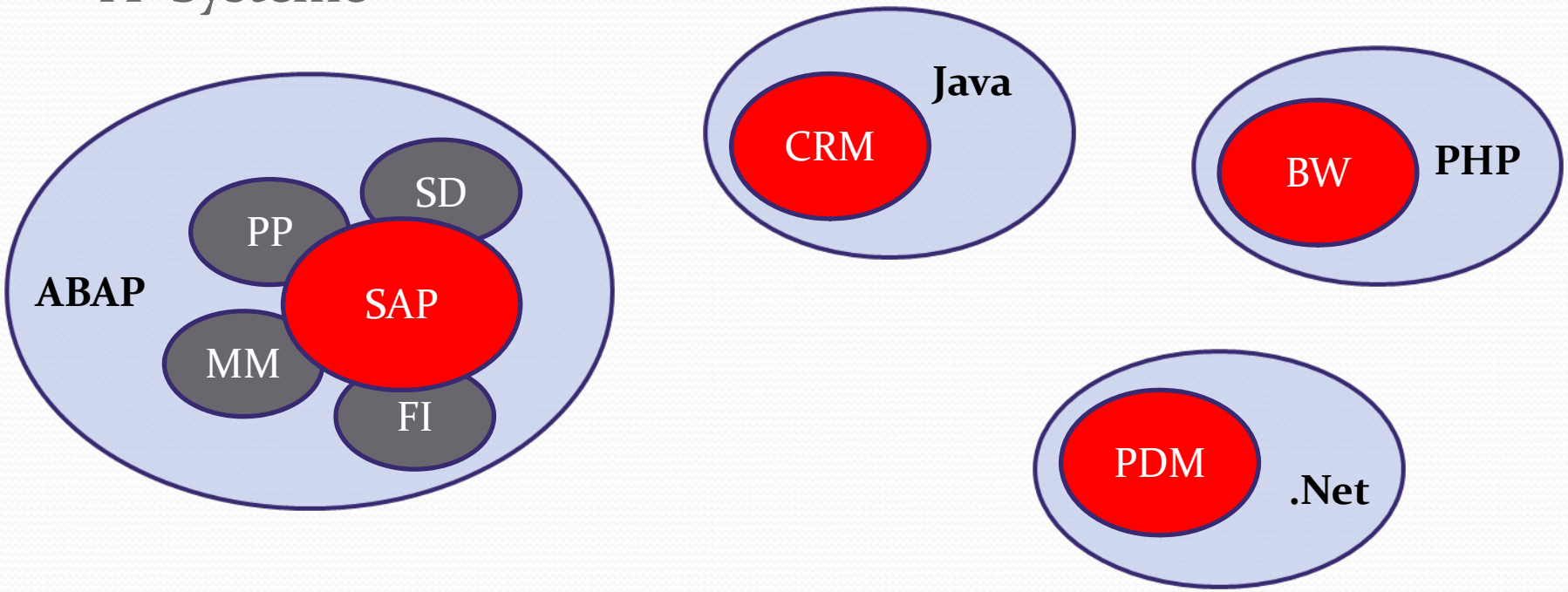
Tobias Miller

Gliederung

1. Ausgangslage
2. SOAP-Webservices
3. Java-Webservice-Technologie
4. Entwicklung eines Webservice
5. Webservice Interoperability Technologies
6. WS-ReliableMessaging
7. Fazit

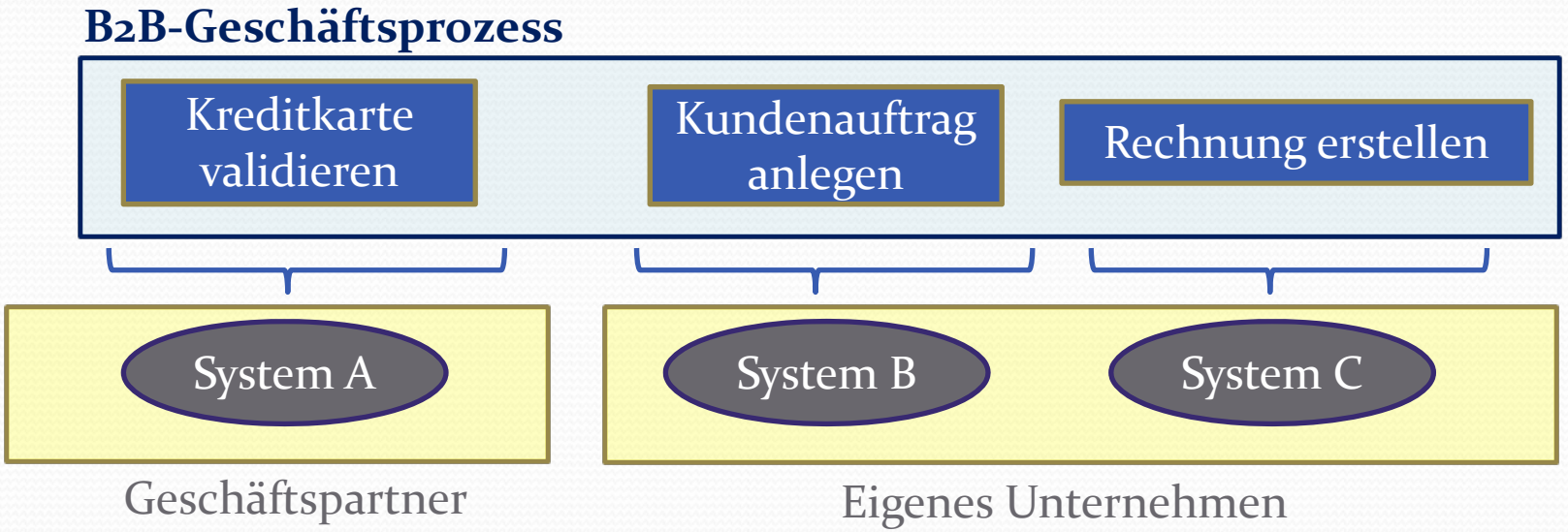
Ausgangslage

- Heterogene Systemlandschaften in großen Unternehmen
  - Unterschiedliche Plattformen und Technologien
- Verteilung der Geschäftslogik auf fachspezifische IT-Systeme



Ausgangslage

- Geschäftsprozesse laufen fachübergreifend ab

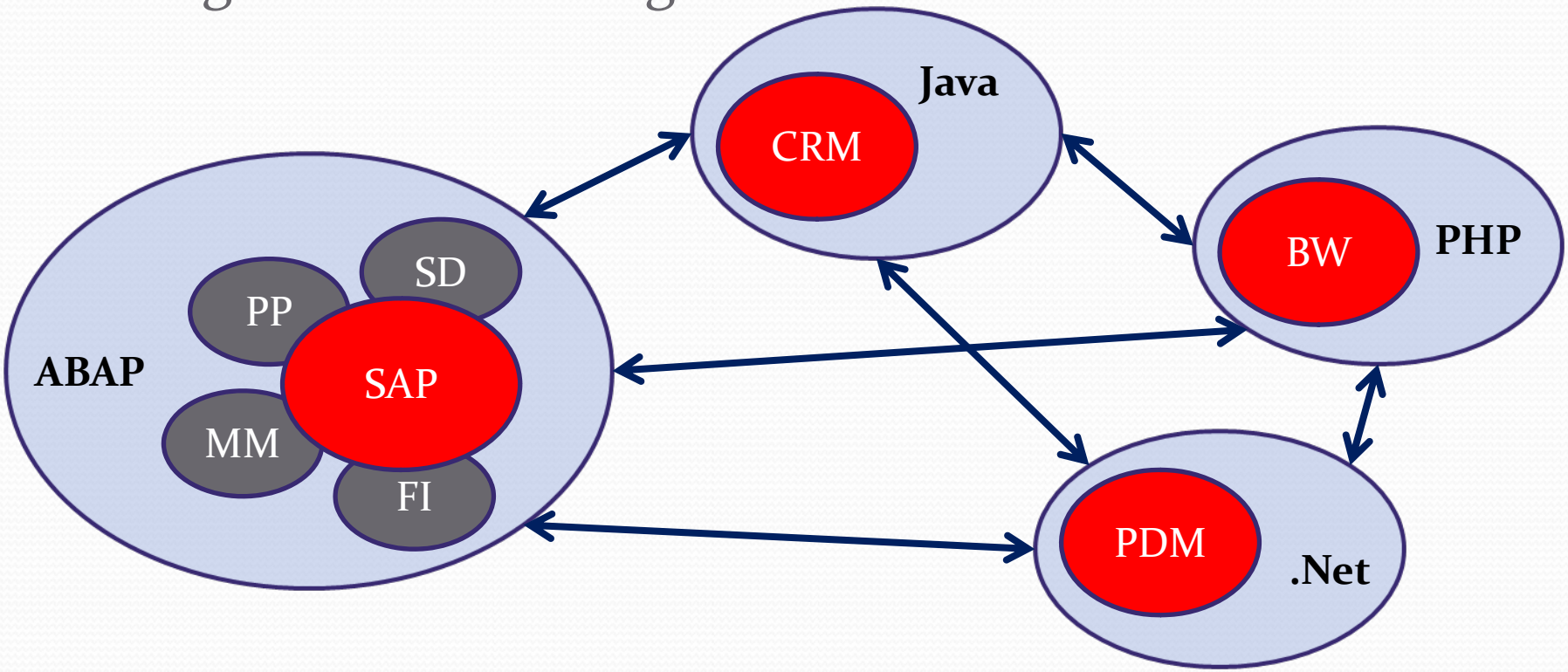


Aufgabe der IT:

Gewährleistung eines integrierten Zusammenspiels der fachspezifischen IT-Systeme

Ausgangslage

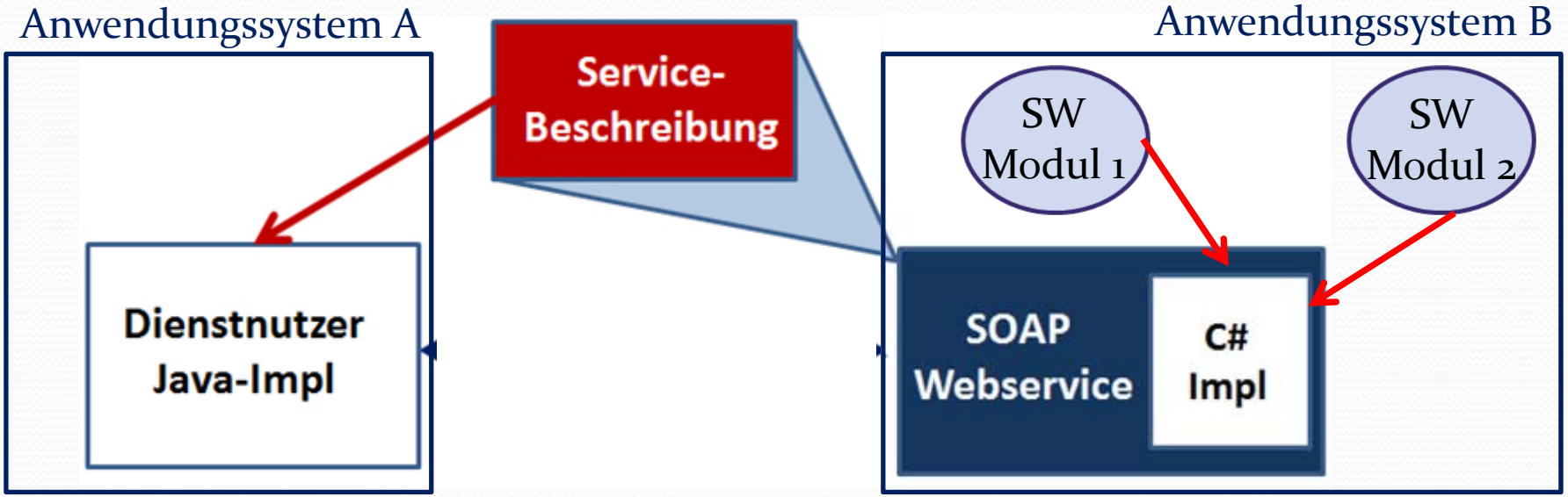
- Austausch von Geschäftsdaten über Schnittstellen
- SOAP-Webservices als geeignete Integrationstechnologie



Gliederung

1. Ausgangslage
2. SOAP-Webservices
3. Java-Webservice-Technologie
4. Entwicklung eines Webservice
5. Webservice Interoperability Technologies
6. WS-ReliableMessaging
7. Fazit

SOAP-Webservices



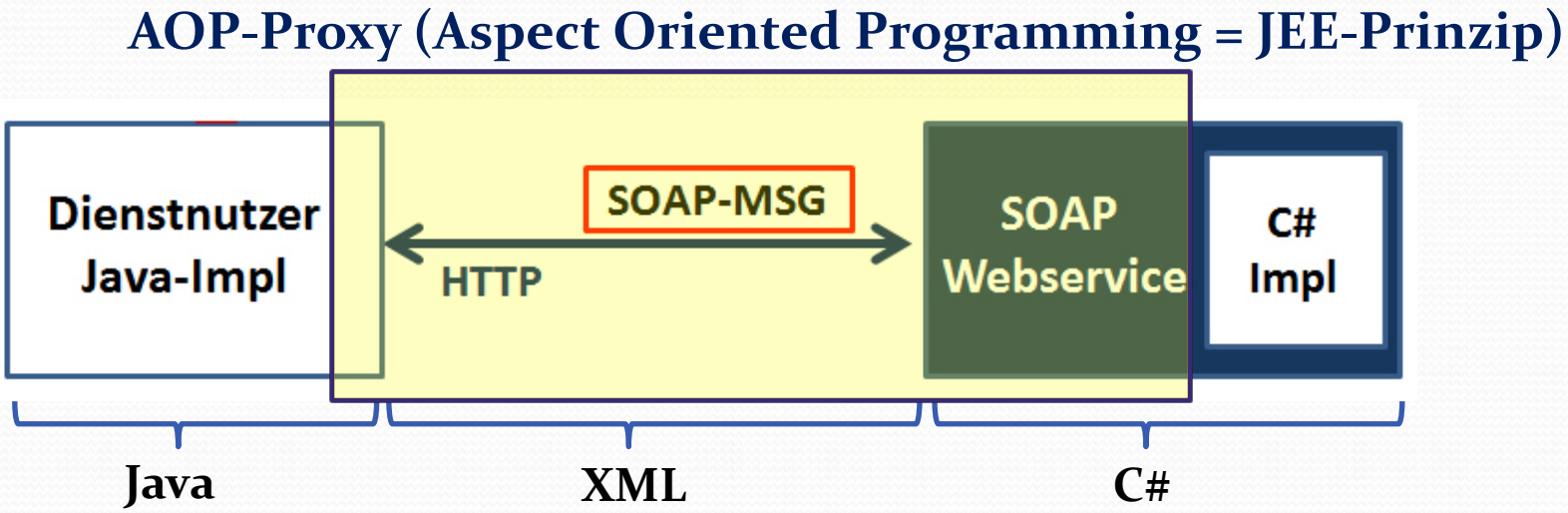
- WS-Provider:

  Stellt Geschäftsfunktionalität via Webservice zur Verfügung
- WS-Client:

  Einbindung der Servicebeschreibung in seine Systemumgebung

SOAP-Webservices

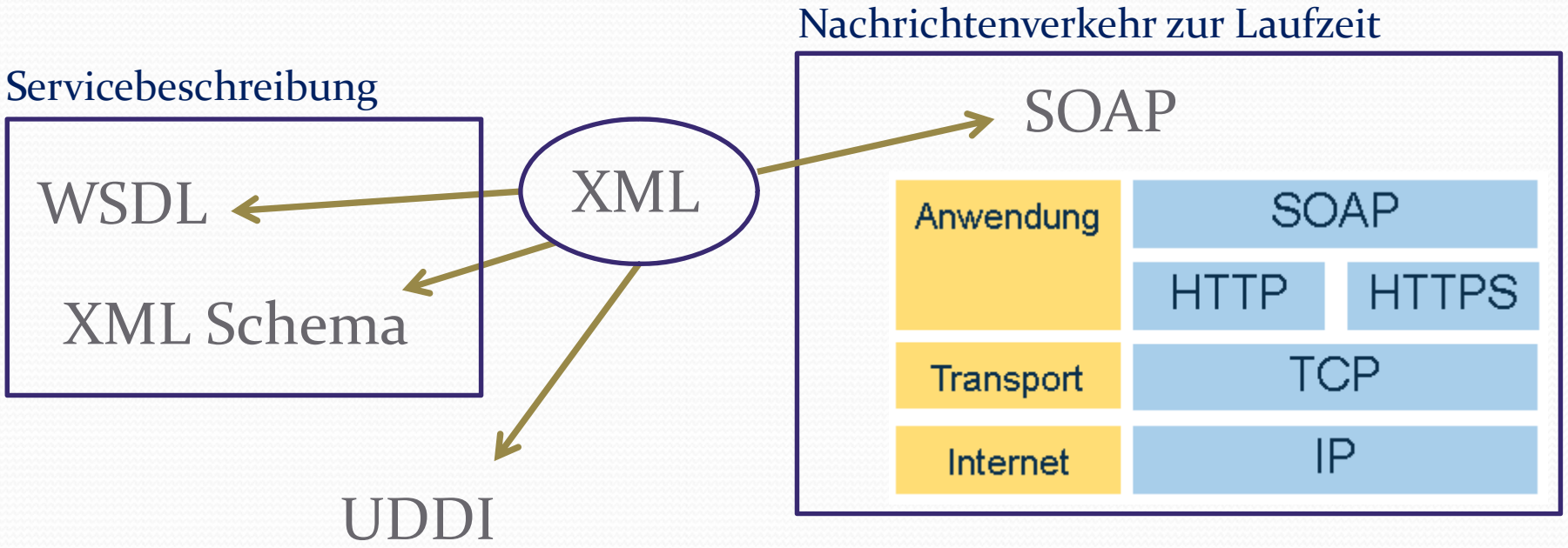
Nachrichtenverkehr über SOAP-Nachrichten
(= XML Dokumente mit spezieller Struktur)



Zwischengelagerter Proxy kapselt Nachrichten-Mapping
(Programmiersprache <-> XML)

SOAP-Webservices

SOAP-WS-Technologie inkludiert folgende Basis-Standards:



Gliederung

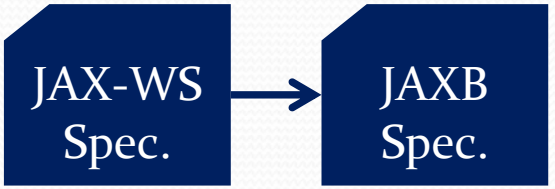
1. Ausgangslage
2. SOAP-Webservices
3. Java-Webservice-Technologie
4. Entwicklung eines Webservice
5. Webservice Interoperability Technologies
6. WS-ReliableMessaging
7. Fazit

Java Webservice Technologie

1. Java API for XML Webservices (JAX-WS)
   - Bestandteil von JRE6 **(->JDK6->Java SE6 -> JEE6)**
   - Kapselung der komplexen SOAP-WS-Technologie vor dem WS-Entwickler
     1. Entwickler verwendet Webservice-spezifische Annotationen in der Java-Implementierung
     2. Die Aufgaben der Webservice-Erzeugung und Nachrichtenprozessierung zur Laufzeit übernimmt JAX-WS

Java Webservice Technologie

1. Java API for XML Webservices (JAX-WS)
   - Definiert Mapping zwischen Java und WSDL/XSD
   - JAX-WS ist stark in JAXB (Java Architecture for XML-Binding) integriert



Bsp 1:

Delegation des Laufzeit-Mappings an JAXB (Java <-> SOAP)

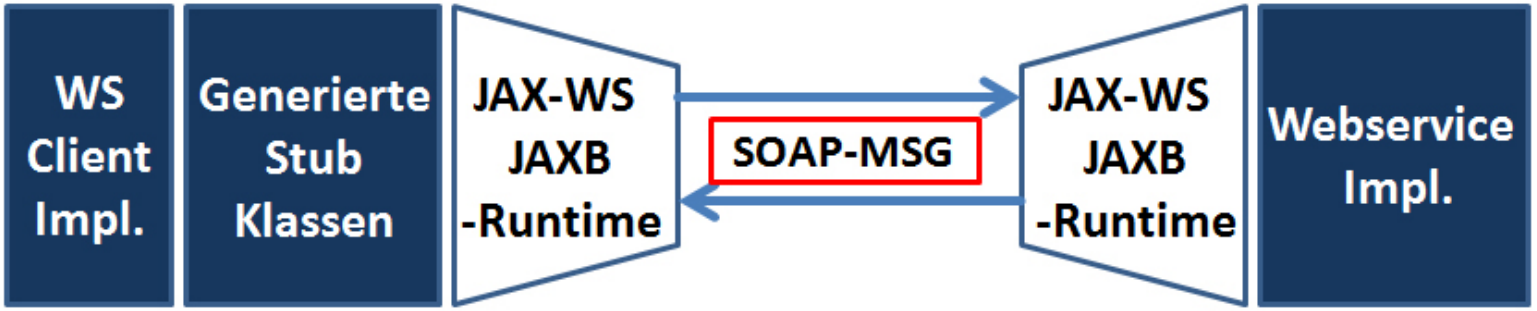
Bsp 2:

Delegation der Erzeugung der Servicebeschreibung an JAXB (Java <-> WSDL)

Java Webservice Technologie

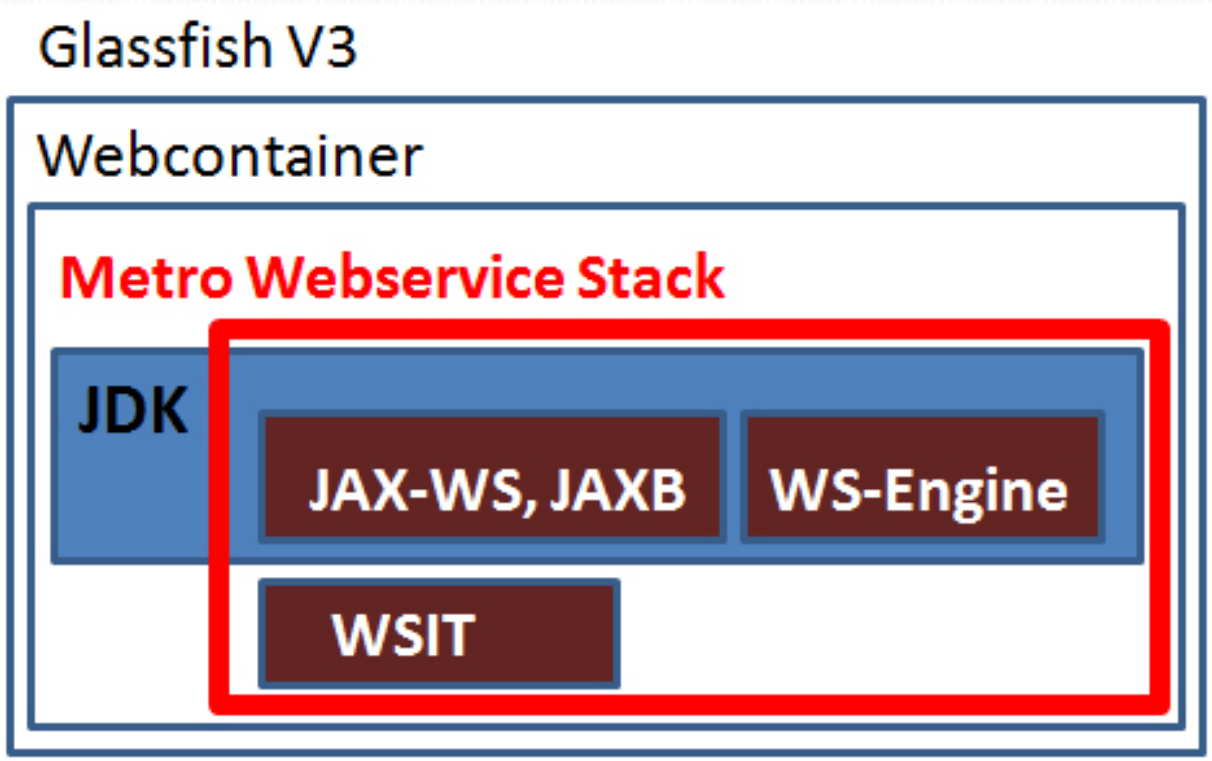
2. Java Architecture for XML-Binding (JAXB)
   - Beschreibt die Bindung einer Java-Klasse an ein XML-Schema
   - Bietet Funktionalitäten für alle Anwendungsbereiche rund um die XML-Verarbeitung mittels Java

Interaktion von JAX-WS mit JAXB:



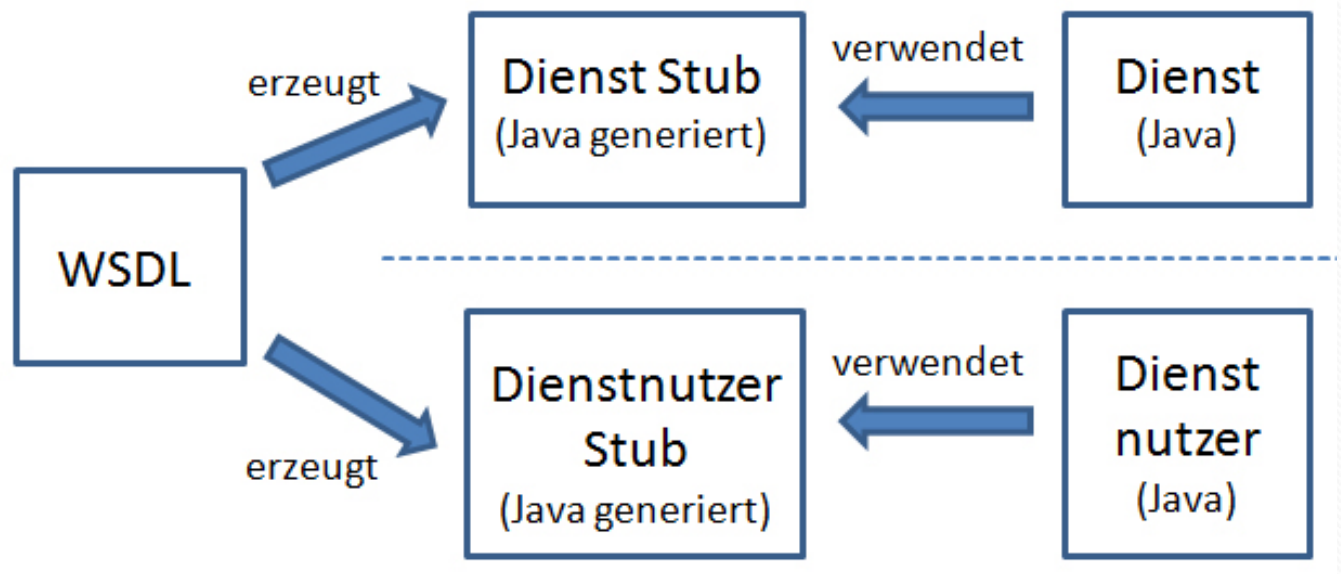
Java Webservice Technologie

- **Metro-Webservice-Stack** als Laufzeitumgebung für Webservices im Applikationsserver Glassfish V3+



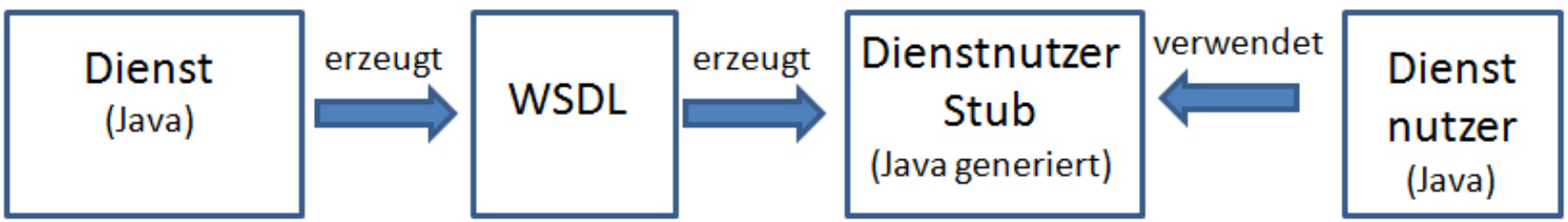
Entwicklungsansätze

1. Contract-First-Ansatz



Entwicklungsansätze

2. Code-First-Ansatz



Gängige Praxis:

- Initiale WSDL-Erstellung via Code-First
- Weiterentwicklungen mit Contract-First

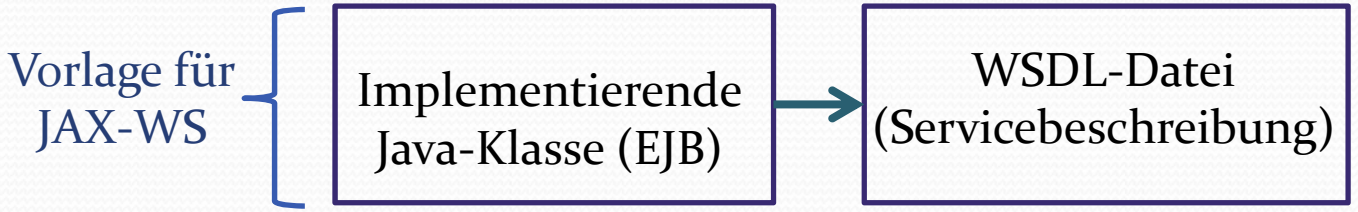
Gliederung

1. Ausgangslage
2. SOAP-Webservices
3. Java-Webservice-Technologie
4. Entwicklung eines Webservice
5. Webservice Interoperability Technologies
6. WS-ReliableMessaging
7. Fazit

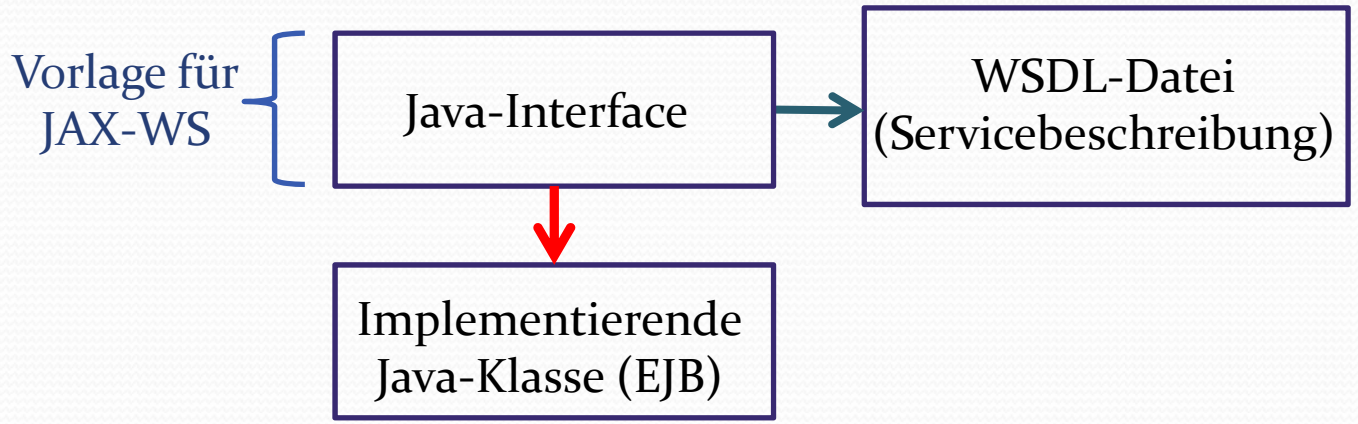
Entwicklung eines Webservice

Code-First (1) Dienstimplementierung:

1. Implizite Service-Endpunkt-Schnittstelle



2. Explizite Service-Endpunkt-Schnittstelle



Entwicklung eines Webservice

- Java-Interface: → @SOAPBinding(style=Style.DOCUMENT) RPC

```
@WebService(name="DateValidationService")
public interface DateValidate {

    @WebMethod(operationName="validateSQLFormatDate")
    @WebResult(name="realDate")
    public boolean validateDate(@WebParam(name="sqlDate") String sqlDate)
    throws DateValidateException;
}
```

- Java-Dienstimplementierung:

```
@WebService(endpointInterface = "validation.DateValidate")
@Stateless
public class DateValidateImpl implements DateValidate{

    @Override
    public boolean validateDate(String sqlDate) throws DateValidateException{
    ...
    }}
```

Entwicklung eines Webservice

Code-First (1) Dienstimplementierung:

```
@WebFault
public class DateValidateException extends Exception{
    private DateValidateFault dateValidateFault;

    public DateValidateException(String msg, DateValidateFault dateValidateFault)
    {
    super(msg);
    this.dateValidateFault = dateValidateFault;
    }
    public DateValidateFault getFaultInfo(){
        return this.dateValidateFault;
    } }
```

Entwicklung eines Webservice

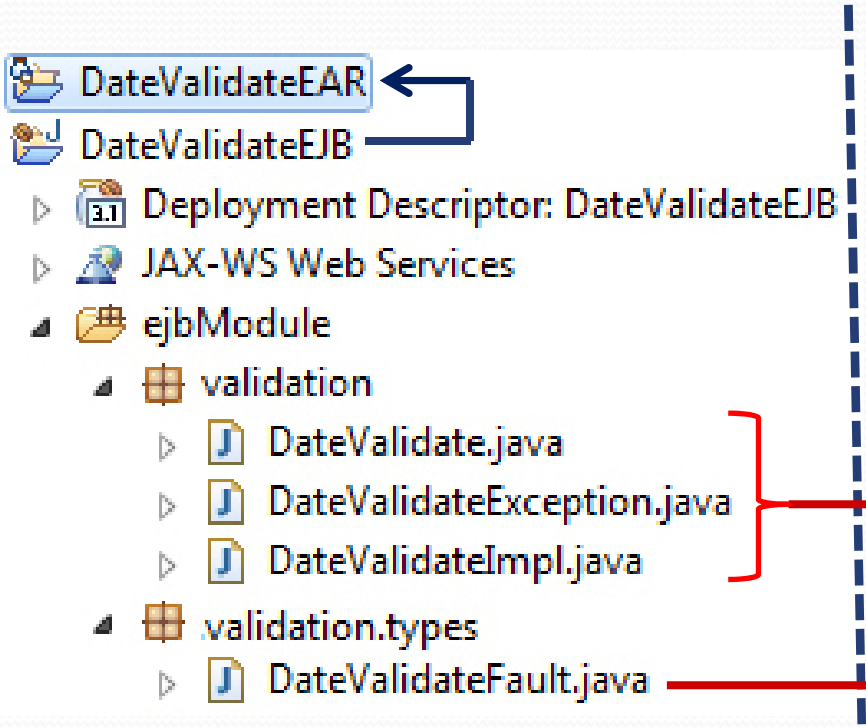
Code-First (1) Dienstimplementierung:

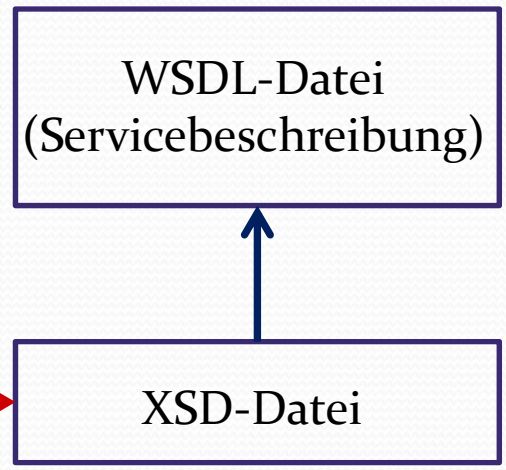
```
@XmlRootElement(name = "DateValidateFault")
public class DateValidateFault {

        private String date;
        private int errorcode;


        public String getdate() {
            return this.date;
        }
        public void setdate(String date) {
            this.date = date;
        }


        public int getErrorCode() {
            return errorcode;
        }
        public void setErrorCode(int errorcode) {
            this.errorcode = errorcode;
        }
}
```

- PUBLIC_MEMBER
- PROPERTY
- FIELD
- NONE

Entwicklung eines Webservice

Code-First (2) Deployment/WSDL-Erzeugung:





Deployment-Vorgang auf Glassfish:

JAX-WS und JAXB nehmen automatisches Mapping vor (Java->XML)

Entwicklung eines Webservice

Code-First (2) Deployment/WSDL-Erzeugung:

```
- <xs:complexType name="dateValidateFault">
  - <xs:sequence>
      <xs:element name="error-description" type="xs:int"/>
      <xs:element name="occure-date" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

Learning by Doing

Java-Deklaration	XSD-Schema
@XmlElement(**required=false**) private int test;	<xs:element name="test" type="xs:int"/>
@XmlElement(required=false) private Integer test2;	<xs:element name="test2" type="xs:int" minOccurs="0"/>

Entwicklung eines Webservice

Code-First (2) Deployment/WSDL-Erzeugung:

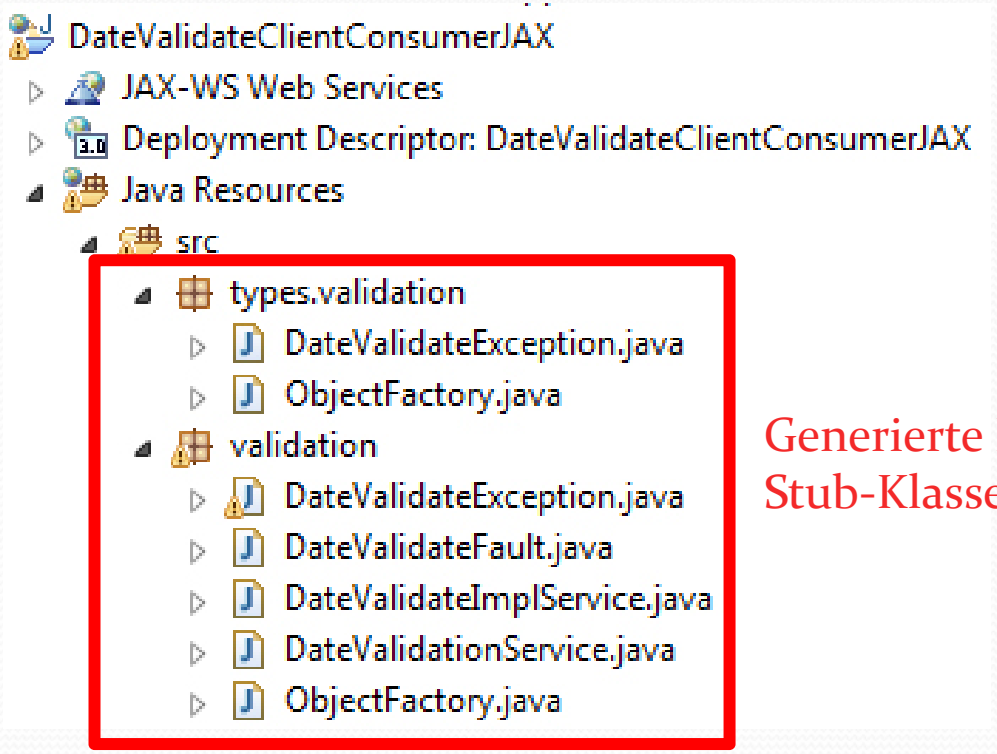
Live-Vorführung

WSDL-Erzeugung
WS-Aufruf mit SOAP-UI

Entwicklung eines Webservice

Code-First (3) Dienstnutzer-Stub-Generierung:

1. Mit dem Eclipse-Wizard (AXIS)
2. Mit „wsimport“-Tool (JAX-WS)



Generierte Dienstnutzer-Stub-Klassen

Entwicklung eines Webservice

Code-First (4) Dienstnutzer-Implementierung:

```
try {

    DateValidateImplService service = new DateValidateImplService();
    DateValidationService port = service.getDateValidateImplPort();

    port.validateSQLFormatDate(date)


} catch (DateValidateException e) {
    out.println("Fehlernachricht: "+ e.getMessage());
}
```

Entwicklung eines Webservice

Code-First (4) Dienstnutzer-Implementierung:

Live-Vorführung

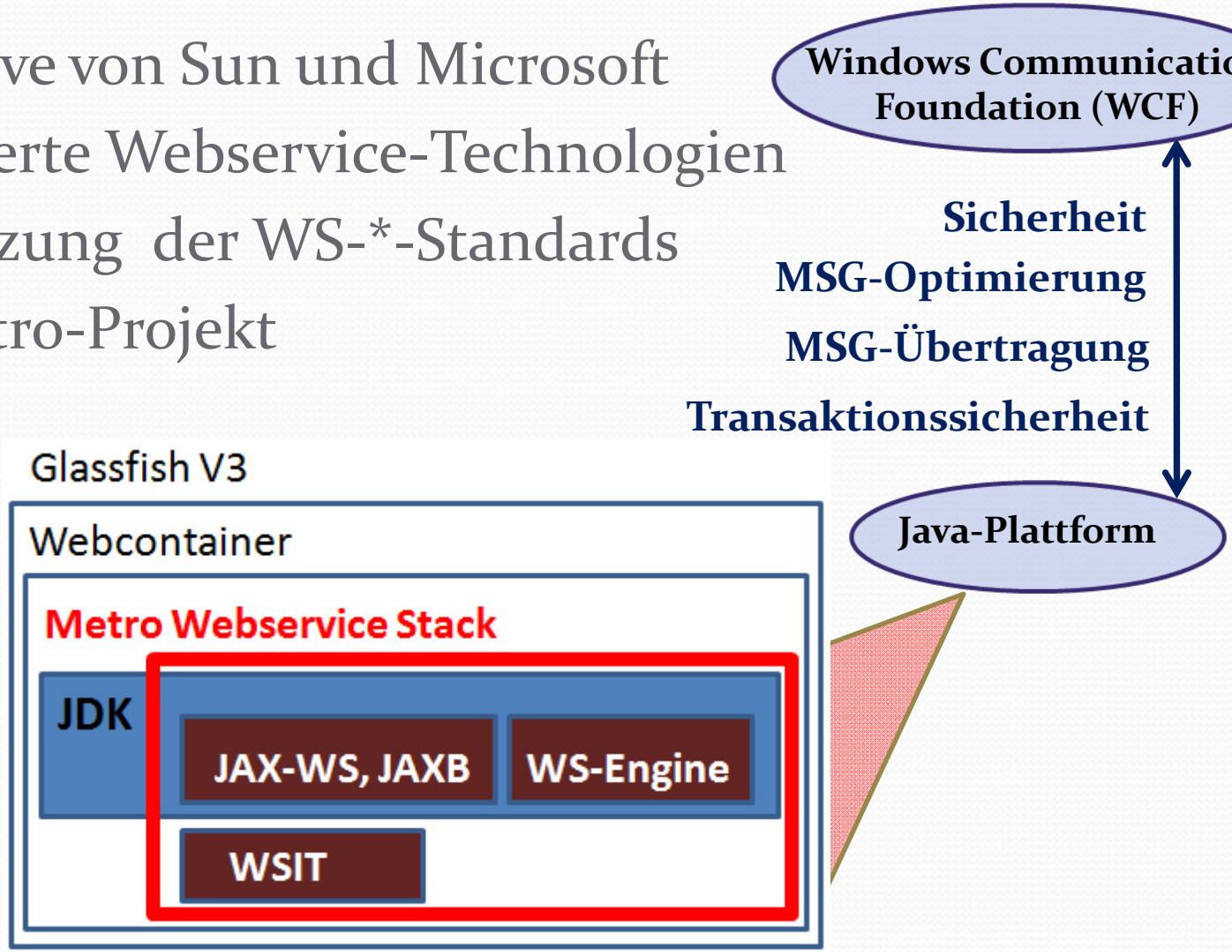
WS-Aufruf mit Clientimplementierung

Gliederung

1. Ausgangslage
2. SOAP-Webservices
3. Java-Webservice-Technologie
4. Entwicklung eines Webservice
5. Webservice Interoperability Technologies
6. WS-ReliableMessaging
7. Fazit

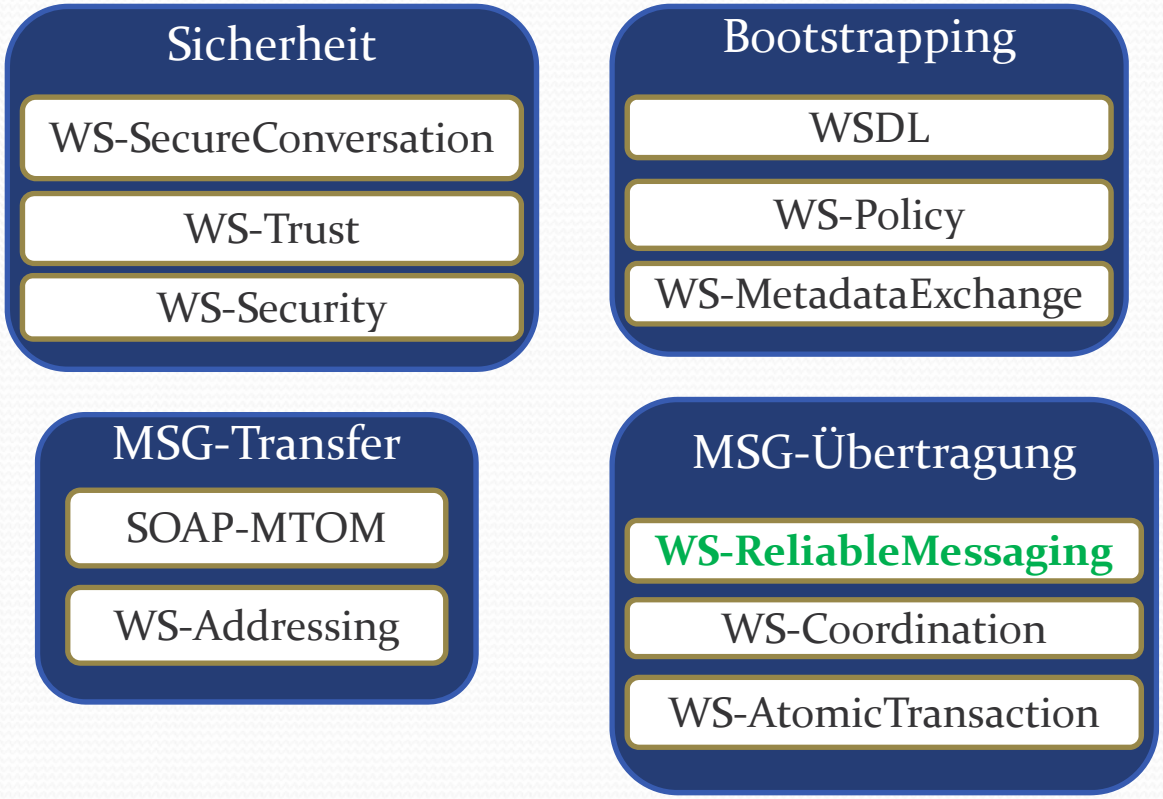
WSIT

- Initiative von Sun und Microsoft
- Erweiterte Webservice-Technologien
- Umsetzung der WS-*-Standards im Metro-Projekt



WSIT

In Metro umgesetzte WS-*-Standards:



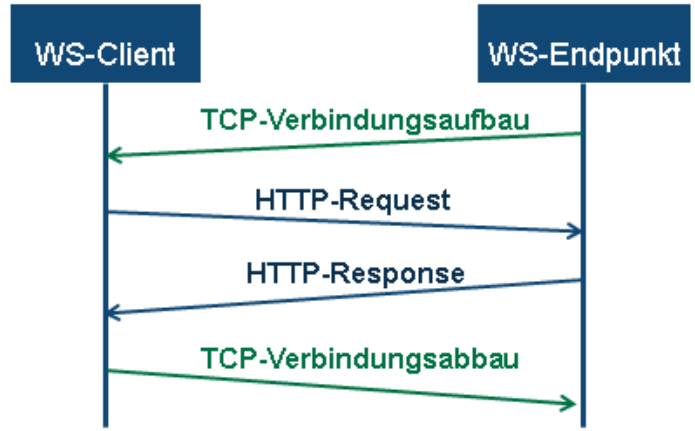
Gliederung

1. Ausgangslage
2. SOAP-Webservices
3. Java-Webservice-Technologie
4. Entwicklung eines Webservice
5. Webservice Interoperability Technologies
6. WS-ReliableMessaging
7. Fazit

WS-ReliableMessaging

HTTP kann keine garantierte Übertragung von SOAP-Nachrichten sicherstellen

- Zustandslos
- Verbindungsorientiert



- Eine garantierte Übertragung impliziert folgende benötigte Mechanismen:
  - Nachrichten-Quittierungsverfahren (Empfänger)
  - Nachrichten-Übertragungswiederholung (Sender)

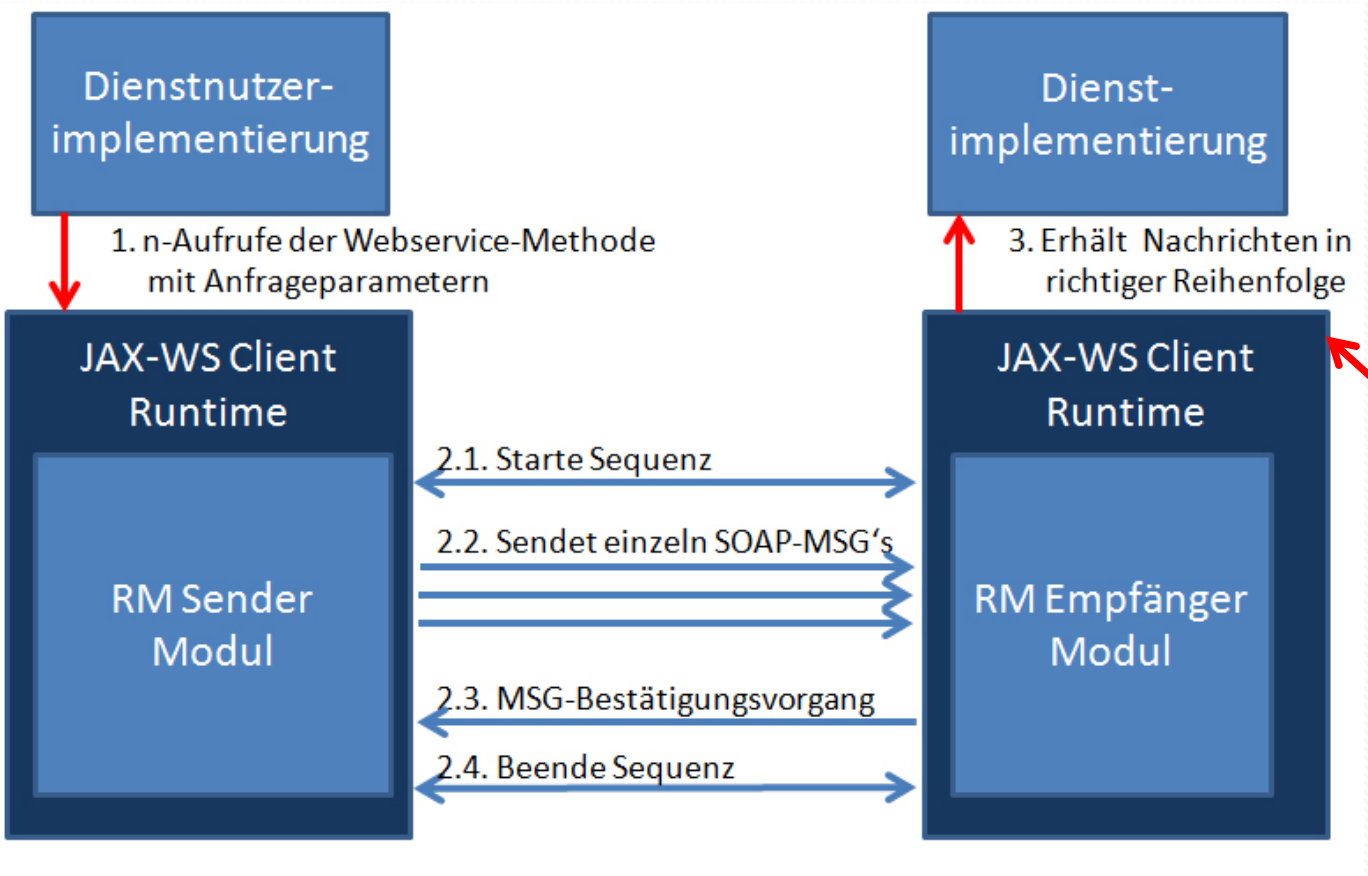
WS-ReliableMessaging

Stellt die zuverlässige Übertragung von SOAP Nachrichten zu einem Webservice-Endpunkt sicher

Zustellgarantien:

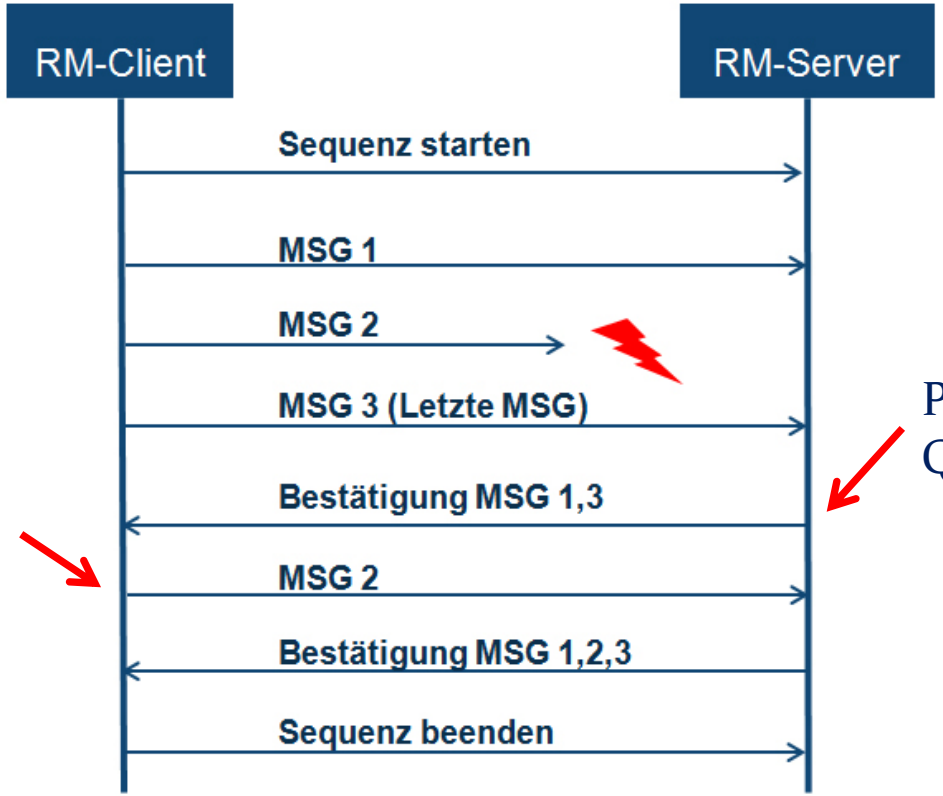
- At Least Once
  SOAP-MSG‘s werden mindestens einmal zugestellt
- Exactly Once
  SOAP-MSG‘s werden genau einmal zugestellt
- Exactly Once In Order
  Zustellung in ursprünglicher Übertragungsreihenfolge

WS-ReliableMessaging



Filterung von Nachrichten-Duplikaten bei EOIO

WS-ReliableMessaging



Übertragungs-
wiederholung

Positiv kumulatives
Quittierungsverfahren

WS-ReliableMessaging

```
<!--
    Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is M
  -->
<!--
    Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is M
  -->
<definitions targetNamespace="http://calculator.me.org/" name="CalculatorWSService">
  <wsp:Policy wsu:Id="CalculatorWSPortBindingPolicy">
    <wsrmp:RMAssertion>
      <wsp:Policy/>
    </wsrmp:RMAssertion>
    <wsam:Addressing/>
  </wsp:Policy>
  <types>
```

WS-ReliableMessaging

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header>
      <Action S:mustUnderstand="1" xmlns="http://www.w3.org/2005/08/addressing">http://calculator.me.org/CalculatorWS/
      <MessageID xmlns="http://www.w3.org/2005/08/addressing">uuid:d9c8bd92-730a-4823-a33a-cb8186fa388f</MessageID>
      <RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:7903c9d0-902b-4a92-9d82-53dce3f68fd6</RelatesTo>
      <To xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing/anonymous</To>
      <ns2:Sequence ns8:mustUnderstand="true" xmlns="http://www.w3.org/2005/08/addressing" xmlns:ns2="http://docs.oasi
         <ns2:Identifier>urn:soapui:e140fc10-fc71-4218-a40b-0994168lfla3</ns2:Identifier>
         <ns2:MessageNumber>1</ns2:MessageNumber>
      </ns2:Sequence>
      <ns2:AckRequested xmlns="http://www.w3.org/2005/08/addressing" xmlns:ns2="http://docs.oasis-open.org/ws-rx/wsrm/
         <ns2:Identifier>urn:soapui:e140fc10-fc71-4218-a40b-0994168lfla3</ns2:Identifier>
      </ns2:AckRequested>
      <ns2:SequenceAcknowledgement xmlns="http://www.w3.org/2005/08/addressing" xmlns:ns2="http://docs.oasis-open.org/
         <ns2:Identifier>uuid:e06837ad-6ed2-490c-9012-f04c37e2145d</ns2:Identifier>
         <ns2:AcknowledgementRange Upper="1" Lower="1"/>
      </ns2:SequenceAcknowledgement>
   </S:Header>
   <S:Body>
      <ns2:addResponse xmlns:ns2="http://calculator.me.org/">
         <return>8</return>
      </ns2:addResponse>
   </S:Body>
</S:Envelope>
```

Gliederung

1. Ausgangslage
2. SOAP-Webservices
3. Java-Webservice-Technologie
4. Entwicklung eines Webservice
5. Webservice Interoperability Technologies
6. WS-ReliableMessaging
7. Fazit

Fazit

1. Java-Webservice-Technologie (JAX-WS i.V.m JAXB):

   Starke Kapselung der Komplexität von SOAP-Webservices vor dem Entwickler

   ➡ Zweischneidige Angelegenheit:

   \+ Entwickler kann sich auf die Anwendungslogik fokussieren

   \- Unzureichendes Debugging in gängigen IDE‘s: JAX-WS-bezogene Fehler können oftmals sehr schwer nachvollzogen werden

Fazit

2. WSIT:

+ Gesamter WSIT-Funktionsumfang ist in Glassfish V3+ verbaut (Metro-Webservice-Stack) und funktionsfähig

- Metro-Entwicklung und Support (inkl. Dokumentationen) wurde in den letzten Jahren stark vernachlässigt.

Eigene Vermutung:

➡ Strategische Neuausrichtung von Oracle

- Stop der WSIT-Weiterentwicklungen im Metro-Projekt

- Verlagerung in den kommerziellen Bereich

Vielen Dank für Ihre Aufmerksamkeit