



# SOAP und WSDL in der Praxis



## Heutige Vorlesung

- ☑ Aufbau von WSDL-Beschreibungen
- ☑ Protokoll-Bindungen in WSDL
- ☑ Google-WSDL lesen und erweitern können
- ☑ Vor- und Nachteile von WSDL

### heute

- Wie wird SOAP/WSDL verwendet?
  - .net, Apache Axis, XMLSpy
  - dynamische Einbindung zur Laufzeit
  - Einbindung zur Entwicklungszeit
- Anforderungen der Praxis
- Erweiterungen von SOAP/WSDL



# Wie wird SOAP/WSDL verwendet?



## .net und Apache Axis



### zwei Möglichkeiten

- Web-Dienst aufrufen
- Web-Dienst anbieten

## Web-Dienst aufrufen



### WSDL → Stubs

- Aus WSDL-Beschreibung *automatisch* Schnittstelle zur Anwendung (Stubs) generieren
- Stubs abstrahieren von SOAP und vom konkreten Übertragungsprotokoll.
- Web-Dienst erscheint als *lokale* Bibliothek.



## Web-Dienst anbieten



### Anwendungsprogramm → WSDL

- Anwendungslogik implementieren
- Anwendung → Server
- aus Anwendungsprogramm automatisch WSDL-Beschreibung erzeugen
- Übersetzung
  - Datenstruktur → Datentypen eines XML-Schemas nicht immer automatisch möglich

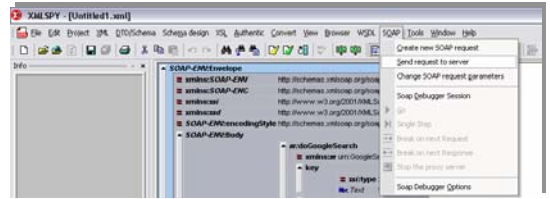
# XMLSpy



## Web-Dienste austesten

- externen Web-Dienst austesten:  
Funktionalität, Reaktionszeit
- eigenen Web-Dienst austesten:  
Testen, ob eigener Web-Dienst auch über WSDL-Beschreibung abrufbar

# Web-Dienste austesten mit XMLSpy



## WSDL → SOAP-Anfrage → SOAP-Antwort

- aus WSDL-Beschreibung SOAP-Anfrage generieren
- Parameter der SOAP-Anfrage anpassen
- SOAP-Anfrage per Knopfdruck an Web-Dienst senden
- Antwort als SOAP

# Was nun verwenden?

## XMLSpy

- + zum schnellen austesten eines Web-Dienstes geeignet
- + SOAP-Anfrage und -Antwort sichtbar
- + keine aufwendige Installation notwendig
- keine Anwendungsentwicklung möglich



## .net

- + Anwendungsentwicklung möglich
- aufwendige Installation



## Apache Axis

- + Anwendungsentwicklung unter Java möglich
- aufwendige Installation



# Web-Dienste: Die Vision

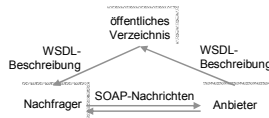
## Beispiel

- personalisierte Informations-Webseite
- Benutzer möchte Echtzeitkurse von bestimmter Aktie einer bestimmten Börse
- Informations-Webseite sucht entsprechenden Web-Dienst für Echtzeitkurse
- Informations-Webseite bindet den Web-Dienst automatisch ein



dynamische Einbindung zur Laufzeit

# Dynamische Einbindung zur Laufzeit

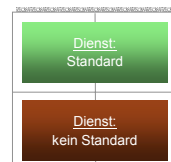


1. Anwendung sucht in Verzeichnis einen Web-Dienst, z.B. für Echtzeitkurse von Aktien
2. Suchergebnis: WSDL-Beschreibung
3. WSDL-Beschreibung → Stubs
4. Anwendung ruft Web-Dienst als lokale Bibliothek auf

Dienst muss automatisch gefunden und aufgerufen werden.

# Aufruf des Dienstes

Dienst aufrufen  
manuell automatisch



- geg.: WSDL-Beschreibung
- Um Web-Dienst *automatisch* abzurufen, muss er **standardisiert** sein:
- WSDL beschreibt zwar *Syntax* der Schnittstelle, nicht aber die *Bedeutung* der Prozedur/Parameter.
- Bedeutung muss außerhalb von WSDL festgelegt (standardisiert) werden

## Beispiel



### Float Aktienkurs(Integer WKN, String Boersenplatz)

- solcher Stub kann automatisch aus WSDL-Beschreibung generiert werden

aber:

- Was bedeutet WKN?
- Was bedeutet Boersenplatz?  
Wie wird der Xetra-Handel in Frankfurt kodiert?
- Was bedeutet Aktienkurs?  
Echtzeit oder verzögert? Wie lange verzögert?
- Was bedeutet Ergebnistyp Float?  
Welche Währung?

## Beispiel



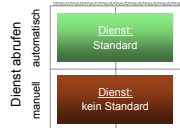
### Float Aktienkurs(Integer WKN, String Boersenplatz)

- solcher Stub kann automatisch aus WSDL-Beschreibung generiert werden

aber:

- Was bedeutet WKN?
- Was bedeutet Boersenplatz?  
Wie wird der Xetra-Handel in Frankfurt kodiert?
- Was bedeutet Aktienkurs?  
Echtzeit oder verzögert? Wie lange verzögert?
- Was bedeutet Ergebnistyp Float?  
Welche Währung?

## Aufruf des Dienstes



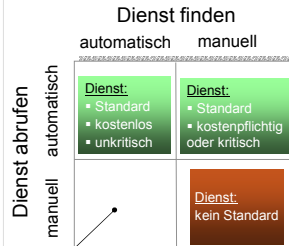
**automatisch bedeutet:**

- Übersetzung WSDL → Stubs automatisch
- Integration von Stubs in Anwendungslogik automatisch

**manuell bedeutet:**

- Übersetzung WSDL → Stubs automatisch
- Integration von Stubs in Anwendungslogik programmiert Entwickler

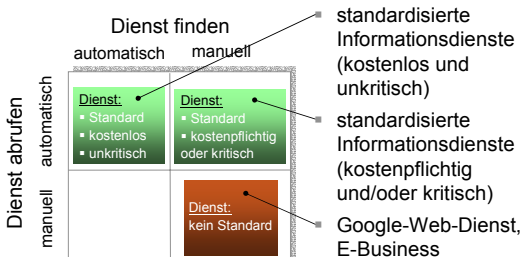
## Dienst finden



- geg.: standardisierter Web-Dienst
- ges.: Anbieter des Web-Dienstes, einschl. WSDL-Beschreibung
- Um Anbieter *automatisch* zu finden, muss Web-Dienst **kostenlos** und **unkritisch** sein.
- andernfalls Rücksprache und Verhandlungen nötig

↳ macht keinen Sinn, da standardisierter Web-Dienst auch automatisch aufgerufen werden kann

## Web-Dienste: Die Realität



## Fazit



zwei Möglichkeiten, Web-Dienst einzubinden:

### zur Laufzeit

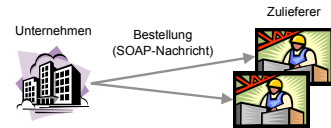
- Vollautomatisches Einbinden prinzipiell mit WSDL möglich, wird aber *Ausnahme* bleiben.
- Grund: nur für standardisierte, kostenlose und unkritische Dienste möglich

### zur Entwicklungszeit

- kein vollautomatisches Einbinden, aber WSDL *erleichtert* das Einbinden ganz erheblich
- Grund: aus WSDL-Beschreibung können automatisch Stubs generiert werden

# Anforderungen der Praxis

## Beispiel



- Absender eindeutig identifizierbar?
- Nachricht unversehrt?
- Nachricht *genau einmal* übermittelt?
- Antwort nach bestimmter Zeit garantiert?
- Passt die Nachricht in den Geschäftsprozess?  
z.B. keine Bestellung ohne vorherige Bestellanfrage

## Realisierung mit SOAP

- Absender eindeutig identifizierbar?
- Nachricht unversehrt?
- Nachricht *genau einmal* übermittelt?
- Antwort nach bestimmter Zeit garantiert?
- Passt die Nachricht in den Geschäftsprozess?

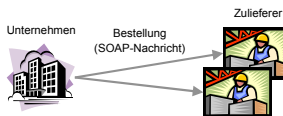
- *keine* der Anforderungen von SOAP direkt unterstützt
- können aber als Zusatzinformationen im Briefkopf der SOAP-Nachricht realisiert werden

## Zusatzinformationen im SOAP-Briefkopf

- Absender eindeutig identifizierbar?
- Nachricht unversehrt?
- Nachricht *genau einmal* übermittelt?
- Antwort nach bestimmter Zeit garantiert?
- Passt die Nachricht in den Geschäftsprozess?

- digitale Signatur
  - Identifikation des Absenders
  - Unversehrtheit der Nachricht
- Anforderung einer Empfangsbestätigung
  - Nachricht mindestens einmal zugestellt
- eindeutige Nachrichtenreferenz
  - Erkennung von Duplikaten
- Verweis auf vorherige Nachrichten, z.B. Bestellanfrage
  - Berücksichtigung des Geschäftsprozesses (Workflows)

## So einfach ist es dann doch nicht...



Problem der Interoperabilität (Zusammenarbeitfähigkeit):

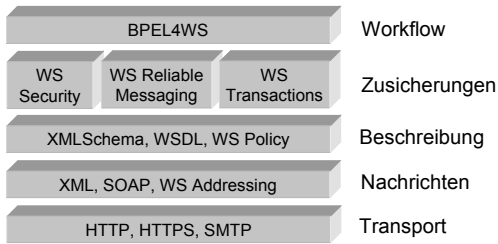
- Zulieferer soll ersetzt werden
- Neuer Zulieferer bekommt WSDL-Beschreibung.
- WSDL beschreibt zwar *Syntax* der Schnittstelle, einschl. der Zusatzinformationen im Briefkopf.
- Was die Zusatzinformationen *bedeuten*, muss aber zwischen Unternehmen und Zulieferer geklärt werden.

## Beispiel für Interoperabilität



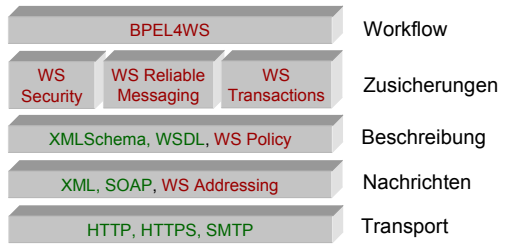
- Was interoperabel bedeutet, kann am Beispiel Fax verdeutlicht werden.
- Fax tatsächlich uneingeschränkt interoperabel:
- Ein Faxgerät kann durch ein anderes ersetzt werden, *ohne* mit potentiellen Sendern Protokolle auszuhandeln.
- Z.B. muss *nicht* geklärt werden, wie Empfang eines Fax bestätigt wird.
- Grund: Es gibt einen etablierten internationalen Fax-Standard.

## Standardisierung von Erweiterungen



- sehr gute Übersicht:  
<http://www-306.ibm.com/software/solutions/webservices/pdf/SecureReliableTransactedWSAction.pdf>

## Etablierte Standards



- etablierte Standards in grün
- (noch) nicht etablierte Standards in rot

## Beispiel: erweiterte SOAP-Nachricht



```

<?xml version="1.0" encoding="UTF-8" ?>
<S:Envelope ...>
  <S:Header>
    <wsa:ReplyTo>
      <wsa:Address>http://business456.com/User12</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>http://fabrikam123.com/Traffic</wsa:To>
    <wsa:Action>http://fabrikam123.com/Traffic/Status</wsa:Action>
  </S:Header>
  <S:Body>
    <wssec:BinarySecurityToken
      Value="X509v3"
      Encoding="Base64Binary"
      Id="BinarySecurityToken"
      Type="http://www.w3.org/2001/XMLSchema#base64Binary"
    </wssec:BinarySecurityToken>
    <wsrm:Sequence
      MessageId="http://fabrikam123.com/seq1234"
      MessageNumber="1"
    </wsrm:Sequence>
    <app:TrafficStatus
      xmlns:app="http://highway.mn.org/payloads"
      Road="520W</road>"
      Speed="3MPH</speed>"
    </app:TrafficStatus>
  </S:Body>
</S:Envelope>
  
```

WS Reliable Messaging wird exemplarisch betrachtet

## Zuverlässige Kommunikation

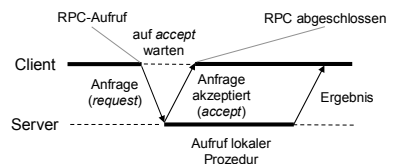


## Beispiel: Buchungsanfrage



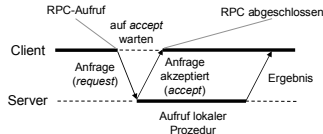
- Unternehmen ruft Buchungsprozedur eines Reiseunternehmens auf
- Buchung nimmt häufig längere Zeit in Anspruch.
- deshalb *asynchroner* Aufruf der Buchungsprozedur

## Asynchrone Buchungsprozedur



- Buchungsanforderung wird vom Client aufgerufen
- Client wartet bis er vom Server Bestätigung der Annahme bekommen hat.
- Server schickt später dann das endgültige Ergebnis.

## Zuverlässige Buchungsanfrage



- Anfrage (*request*): *genau einmal* zustellen
  - Annahme (*accept*): *mindestens einmal* zustellen
  - Ergebnis: *genau einmal* zustellen
  - Annahme immer *vor* Ergebnis, ansonsten wird evtl. zweite Buchungsanfrage gestellt.
- Annahme und Ergebnis bilden eine Sequenz

© Klaus Schild, 2004

31

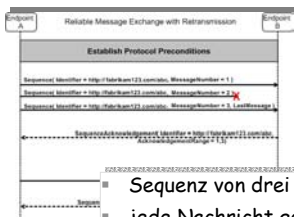
## WS Reliable Messaging

- XML-Standard für zuverlässige Kommunikation
  - kann im Briefkopf von SOAP-Nachrichten oder in WSDL-Beschreibungen verwendet werden
  - gemeinsamer Vorschlag von BEA, IBM, Microsoft und TIBCO
  - von März 2004
  - noch *kein* etablierter Standard
- <http://www-106.ibm.com/developerworks/library/ws-rm/>

© Klaus Schild, 2004

32

## Beispiel

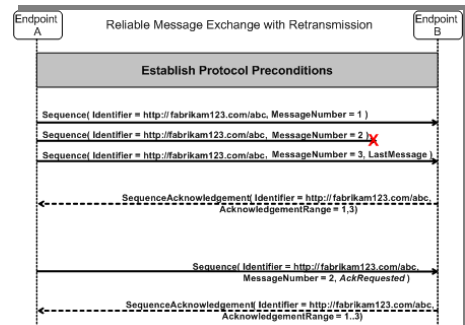


- Sequenz von drei Nachrichten
- jede Nachricht genau einmal zustellen
- Reihenfolge beachten
- Empfang der Sequenz bestätigen

© Klaus Schild, 2004

33

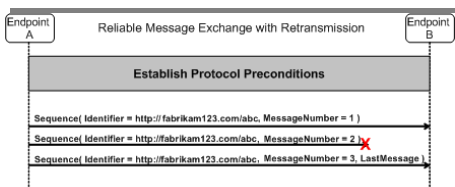
## Beispiel



© Klaus Schild, 2004

34

## Sequenz von SOAP-Nachrichten senden



© Klaus Schild, 2004

35

## Erste SOAP-Nachricht

```
<S:Envelope xmlns:wsm="http://schemas.xmlsoap.org/ws/2004/03/rm"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/71e0654e-5ce8-477b-bb9d-34f05cfc9c
    </wsa:MessageID>
    <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:ReplyTo>
    <wsm:Sequence>
      <wsu:Identifier>http://Business456.com/abc</wsu:Identifier>
      <wsm:MessageNumber>1</wsm:MessageNumber>
    </wsm:Sequence>
  </S:Header> ...
```

- WS RM benutzt WS Addressing
- Nachricht hat eindeutige Referenz (MessageID)

© Klaus Schild, 2004

36

## Erste SOAP-Nachricht

```
<S:Envelope xmlns:wsm="http://schemas.xmlsoap.org/ws/2004/03/rm"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
    </wsa:MessageID>
    <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:ReplyTo>
    <wsm:Sequence>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
      <wsm:MessageNumber>1</wsm:MessageNumber>
    </wsm:Sequence>
  </S:Header> ...
```

WS RM benutzt WS Utilities  
Sequenz hat eindeutige Referenz (Identifier)  
erste Nachricht der Sequenz (MessageNumber)

© Klaus Schild, 2004

37

## Zweite SOAP-Nachricht

```
<S:Envelope ...>
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/daa7d0b2-c8e0-476e-a9a4-d164154e38de
    </wsa:MessageID>
    <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:ReplyTo>
    <wsm:Sequence>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
      <wsm:MessageNumber>2</wsm:MessageNumber>
    </wsm:Sequence>
  </S:Header> ...
```

neue Nachrichten-Referenz  
alte Sequenz-Referenz

© Klaus Schild, 2004

38

## Dritte und letzte SOAP-Nachricht

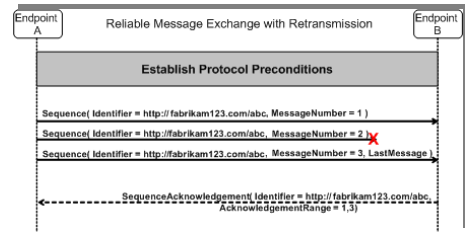
```
<S:Envelope ...>
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
    </wsa:MessageID>
    <wsa:To>http://fabrikam123.com/serviceB/123</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
    </wsa:ReplyTo>
    <wsm:Sequence>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
      <wsm:MessageNumber>3</wsm:MessageNumber>
      <wsm>LastMessage/>
    </wsm:Sequence>
  </S:Header> ...
```

neue Nachrichten-Referenz  
alte Sequenz-Referenz

© Klaus Schild, 2004

39

## Bestätigung



© Klaus Schild, 2004

40

## Bestätigung

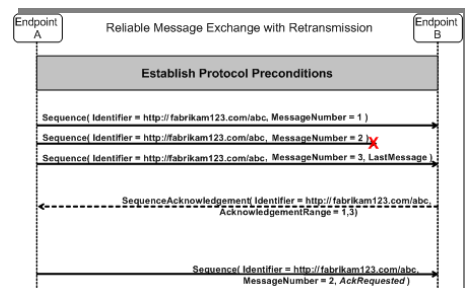
```
<S:Envelope ...>
  <S:Header>
    <wsa:MessageID>
      http://fabrikam123.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
    </wsa:MessageID>
    <wsa:To>http://Business456.com/serviceA/789</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://fabrikam123.com/serviceB/123</wsa:Address>
    </wsa:ReplyTo>
    <wsm:SequenceAcknowledgment>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
      <wsm:AcknowledgmentRange Upper="1" Lower="1"/>
      <wsm:AcknowledgmentRange Upper="3" Lower="3"/>
    </wsm:SequenceAcknowledgment>
  </S:Header> ...
```

erste und dritte Nachricht angekommen

© Klaus Schild, 2004

41

## Zweite Nachricht nochmals senden



© Klaus Schild, 2004

42

## Zweite Nachricht nochmals senden

```
<S:Envelope ...>
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
    </wsa:MessageID>
    ...
    <wsrm:Sequence>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
      <wsrm:MessageNumber>2</wsrm:MessageNumber>
    </wsrm:Sequence>
    <wsrm:AckRequested>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
    </wsrm:AckRequested>
  </S:Header> ...
```

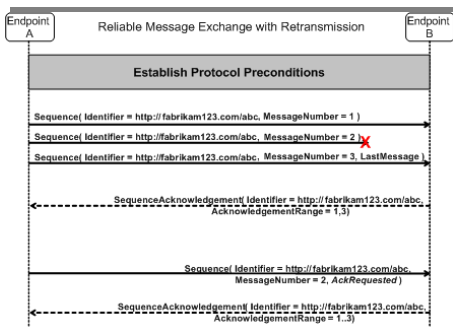
- zweite Nachricht der Sequenz
- neue Nachrichten-Referenz
- alte Sequenz-Referenz

## Zweite Nachricht nochmals senden

```
<S:Envelope ...>
  <S:Header>
    <wsa:MessageID>
      http://Business456.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
    </wsa:MessageID>
    ...
    <wsrm:Sequence>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
      <wsrm:MessageNumber>2</wsrm:MessageNumber>
    </wsrm:Sequence>
    <wsrm:AckRequested>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
    </wsrm:AckRequested>
  </S:Header> ...
```

- Empfangsbestätigung für Sequenz verlangt

## Endgültige Bestätigung



## Endgültige Bestätigung

```
<S:Envelope ...>
  <S:Header>
    <wsa:MessageID>
      http://fabrikam123.com/guid
    </wsa:MessageID>
    <wsa:To>http://Business456
    <wsa:ReplyTo>
      <wsa:Address>http://fabrikam123.com/serviceB/123</wsa:Address>
    </wsa:ReplyTo>
    <wsrm:SequenceAcknowledgement>
      <wsu:Identifier>http://Business456.com/RM/ABC</wsu:Identifier>
      <wsrm:AcknowledgementRange Upper="3" Lower="1"/>
    </wsrm:SequenceAcknowledgement>
  </S:Header> ...
```

- alle 3 Nachrichten empfangen
- neue Nachrichten-Referenz
- alte Sequenz-Referenz

## Was gibt es noch?

- **wsu:Expires**: Verfallsdatum und -zeit
- **wsrm:InactivityTimeout**: wie lange Empfänger höchstens auf nächste Nachricht warten muss
- **wsrm:SequenceFault**: Fehlermeldung bzgl. Sequenzen, z.B.:
  - **wsrm:SequenceTerminated**
  - **wsrm:InvalidAcknowledgement**
- **wsrm:DeliveryAssurance**: verlangte Zuverlässigkeit, z.B.:
  - **AtLeastOnce**
  - **InOrder**

## Integration in SOAP oder WSDL?

- WS-RM-Elemente können im Briefkopf einer SOAP-Nachricht erscheinen.
- Elemente wie z.B.

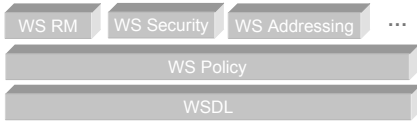
```
<wsrm:DeliveryAssurance Value="wsrm:AtLeastOnce"/>
```

machen in SOAP-Nachrichten allerdings keinen Sinn.
- Solche Elemente werden in einer WSDL-Beschreibung integriert.

Wie werden WS-RM-Elemente in WSDL integriert?

## Zweistufige Integration in WSDL

1. WS-RM-Elemente werden als Kommunikationsregeln (*policies*) formuliert.  
Formulierung mit Hilfe von WS Policy
2. Diese Kommunikationsregeln werden dann in WSDL integriert.
  - Gleiches Vorgehen auch bei anderen Erweiterungen:



© Klaus Schild, 2004

49

## Beispiel

```
<wsdl:definitions name="PurchaseOrder" ...>
  <wsdl:message name="OrderRequest">
    <wsp:Policy>
      <wsrm:DeliveryAssurance Value="wsrm:ExactlyOnce"
        wsp:Usage="wsp:Required"/>
    </wsp:Policy>
    ...
  </wsdl:message>
  <wsdl:message name="OrderResponse">
    ...
  </wsdl:message>
  ...
</wsdl:definitions>
```

hier *wsp:Policy* als Erweiterungselement  
tatsächlich erlaubt aber WSDL für *message*  
und *portType* *keine* Erweiterungselemente  
daher etwas kompliziertere Kodierung mit  
Attributen

© Klaus Schild, 2004

50

## Bewertung der Erweiterungen

© Klaus Schild, 2004

51

## Bewertung der Erweiterungen

- + einzelne Erweiterungen *unabhängig* voneinander
- + meist gemeinsame Vorschläge von Microsoft und IBM
- noch *keine* etablierten Standards
- WS Policy Grammatik zur Festlegung von qualitativen Aspekten, einzelne qualitative Aspekte müssen noch standardisiert werden

*Keine prinzipiellen Hürden, jedoch noch  
ein langer Standardisierungsweg zu gehen!*

© Klaus Schild, 2004

52

## Wie geht es weiter?

- ☑ Wie wird WSDL verwendet?
  - .net, Apache Axis, XMLSpy
  - dynamische Einbindung zur Laufzeit
  - Einbindung zur Entwicklungszeit
- ☑ Anforderungen der Praxis
- ☑ Erweiterungen von SOAP/WSDL

### heute

- 16:15 betreute Rechnerübung

### nächste Woche (7.7.)

- Ausblick, Rückblick, Hinweise für Klausur

### Klausur am 21.7.

© Klaus Schild, 2004

53