
Martin Renner

01. Juli 2004



ex|Xcellent
solutions

WebServices in der Praxis

Messdaten-Integration bei DaimlerChrysler



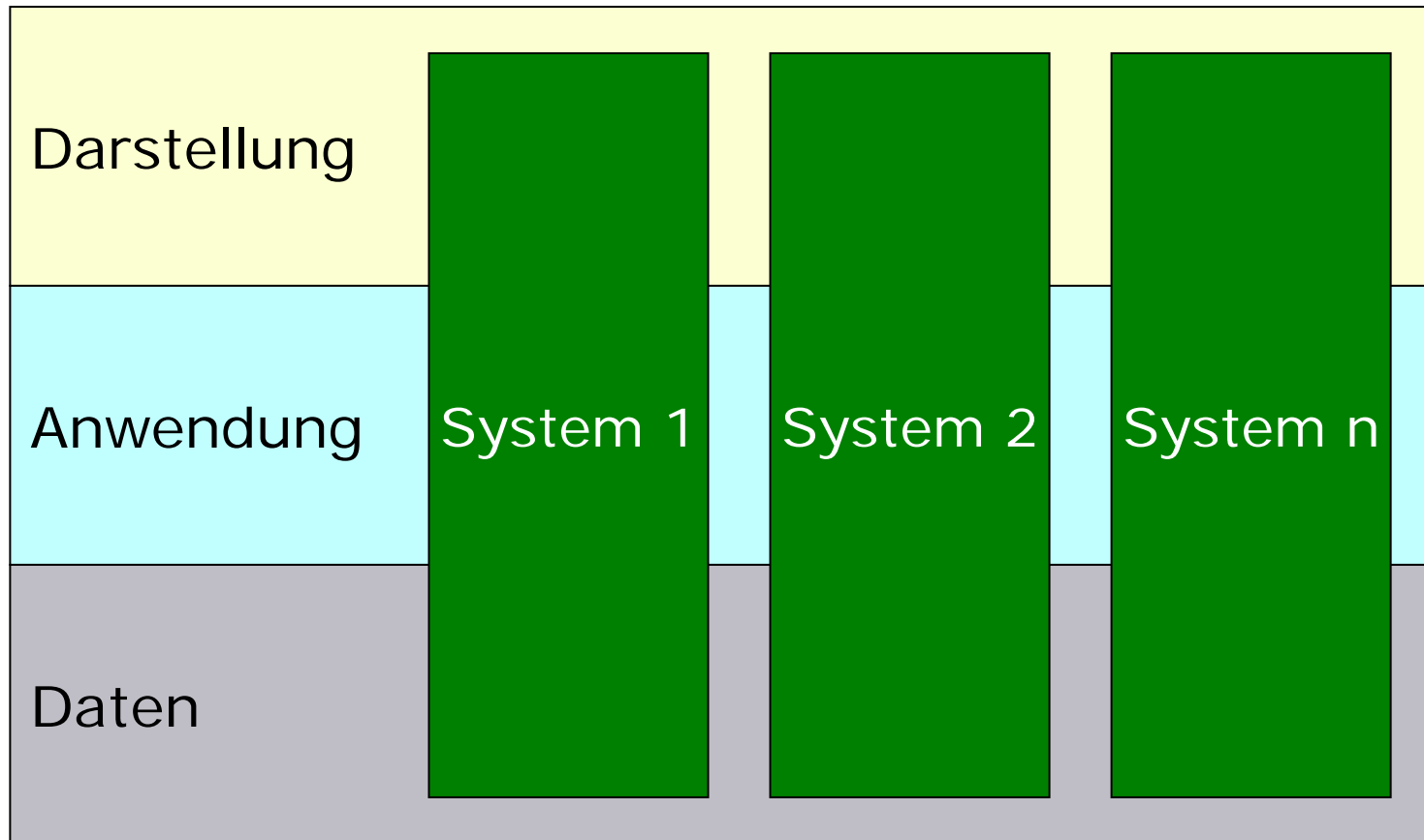
- x|Projektvorstellung
- x|Voruntersuchungen zur Implementation
- x|Architekturkonzepte
- x|Details zur Umsetzung
- x|Probleme/Schwachstellen
- x|Performance- und Volumenbetrachtung
- x|Resümee
- x|Diskussion

Projektvorstellung

Qualitätsdatenbank für Messdaten

- x| wird in der Fahrzeugproduktion in den unterschiedlichsten Bereichen eingesetzt (Motorbau, Montage, Lackierung etc.)
- x| Messmaschinen liefern je nach Bereich sehr unterschiedliche Messdaten an eine Datenbank
- x| je nach Bereich teilweise sehr hohes Datenaufkommen (z.B. Schichtdicken in der Lackierung)
- x| Datenbasis für Auswertungswerkzeuge zur Qualitätsbestimmung

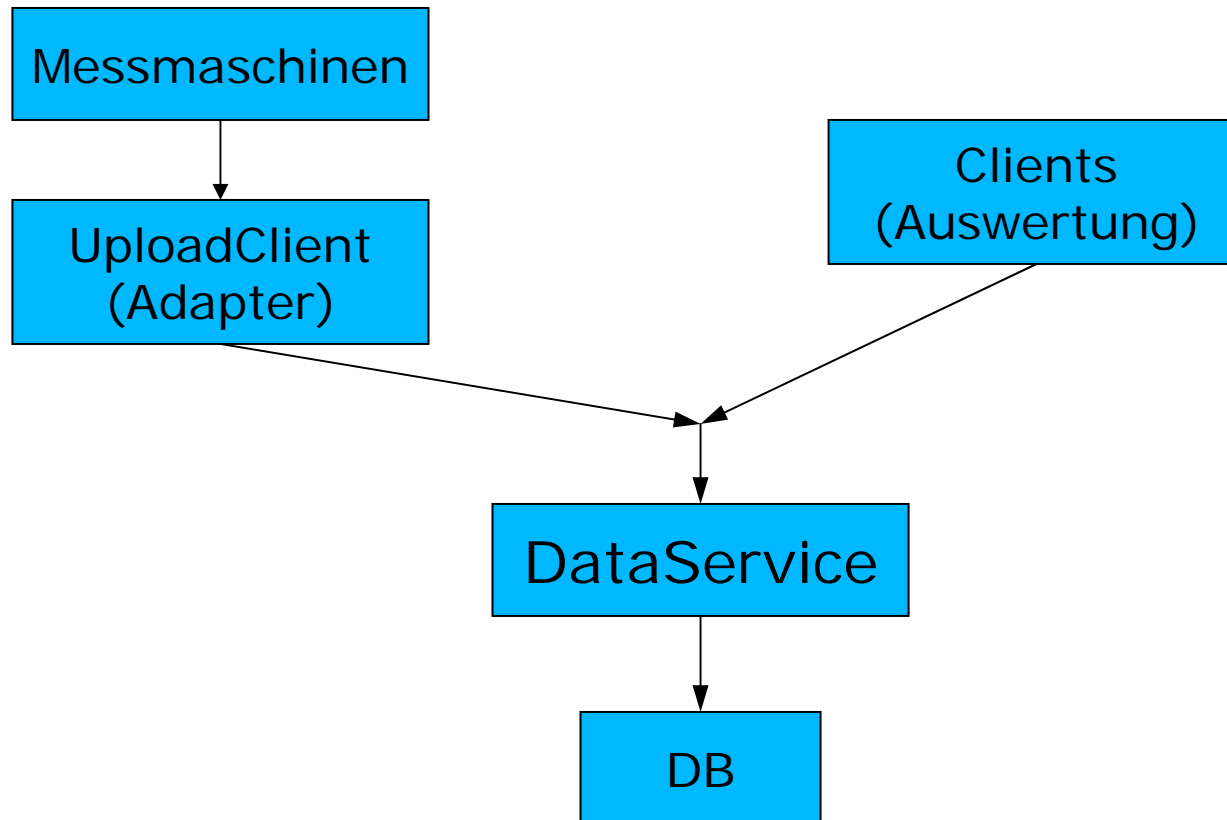
Bisher



Ziel

- x| Einheitliche Datenbank für alle Bereiche
- x| Einheitliche Bewertung des Qualitätsstandes über Werksgrenzen
- x| Schaffung standardisierter Auswertungswerkzeuge

Ziel



Anforderungen

x| Heterogenes Client-Umfeld (Java und .NET)
→ Implementation als Webservice

x| Besondere Anforderungen (↔ Webservice ?)
| akzeptable Antwortzeiten
| auch in Bezug auf Datenvolumen

Ausgangssituation

Prototyp von DaimlerChrysler in C# (.NET)

Zielsetzung

Produktionstaugliche Lösung in Java

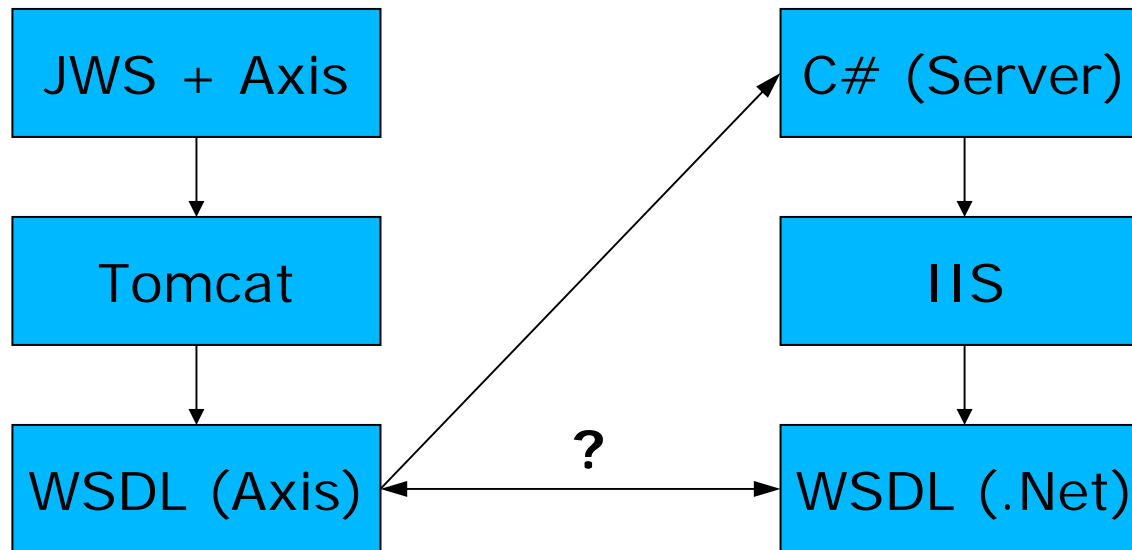
Voruntersuchungen

x| von .NET generierte WSDL konnte von Axis nicht verarbeitet werden

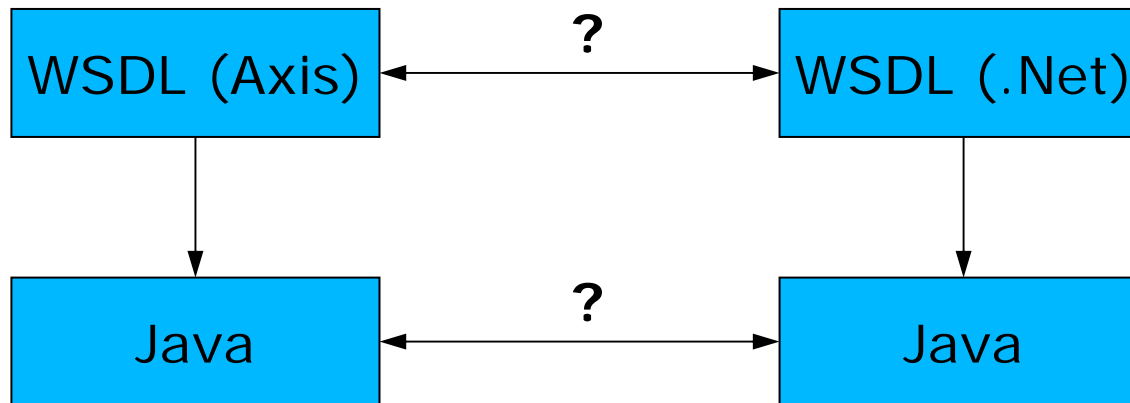
```
<types>
  <s:schema elementFormDefault="qualified",
            targetNamespace="DCqdataService">
    <s:import
      namespace="http://microsoft.com/wsdl/types/"
    />
    <s:element name="Status">
      ...
    </s:element>
  </s:schema>
</types>
```

WSDL generieren lassen oder selbst erstellen?

WSDL generieren lassen oder selbst erstellen?
Ein „Round-Trip“-Beispiel in der Praxis:



WSDL generieren lassen oder selbst erstellen?



WSDL selbst erstellen!

RPC, Document oder Document Wrapped?

- x|**RPC = „Section 5 encoding“ von SOAP 1.1
- x|**RPC wurde vor XML-Schema definiert
- x|**RPC stammt aus der Zeit, als es noch gar keine „WebServices“ gab
- x|**Mit Java+RPC ist kein „unsigned“ möglich
- x|**Erst später entstand der Bedarf, Schnittstellen (SOAP-Methoden) zu beschreiben
→ WSDL

RPC, Document oder Document Wrapped?

x|Java-Frameworks:

bisher Fokus auf RPC, Document gewinnt an Bedeutung

x|.NET:

Document im Vordergrund, RPC nicht vollständig

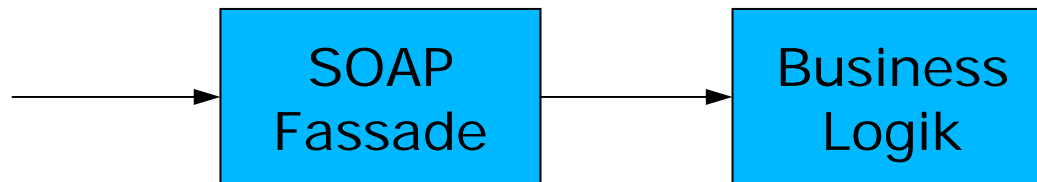
x|[msdn: The Argument against SOAP encoding](#)

x|SOAP 1.2: RPC wird optional

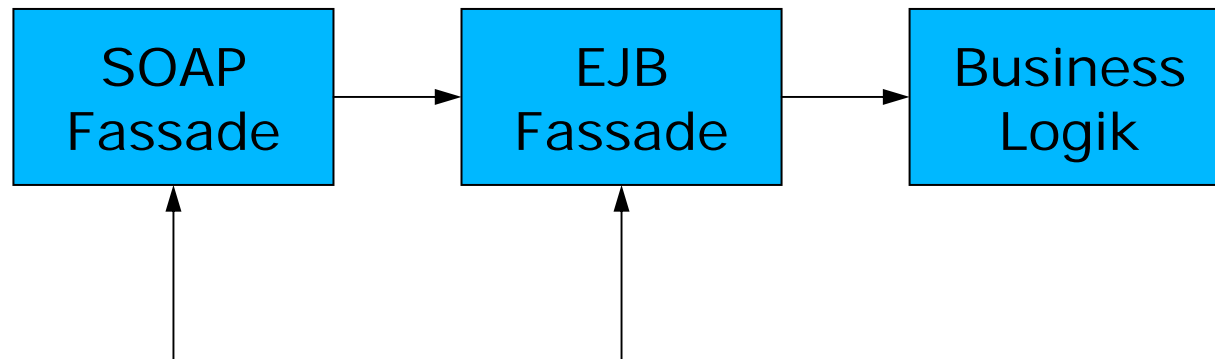
→ Document oder Document Wrapped

Architekturkonzepte

- x| Unterschiedliche Frameworks generieren unterschiedliche Klassen
- x| Trennung zwischen Framework-Klassen und Business Logik
- x| SOAP nur eine Fassade für (bestehende) Business Objekte



x|Business Logik kann somit auch als EJB angeboten werden



x|Die jeweilige Fassade ist dafür zuständig, die Business Logik mit Ressourcen zu versorgen

Umsetzung

- x|WSDL erstellen ist sehr komplex, Tools helfen hier momentan wenig
- x|WS-I Basic Profile berücksichtigen
(Web Services Interoperability Organization)
Basic Profile Version 1.0 vom 16.04.2004
- x|Für Eclipse: [Web Service Validation Tools](#)
Bindet WS-I Basic Profile in Eclipse ein
- x|Mit Java und .NET die WSDL prüfen

Tipps für die WSDL-Erstellung

- x|**komplexe Schema-Konstrukte vermeiden (z.B. choice)
- x|**komplette Schema- und WSDL-Definition in einer Datei ablegen
- x|**für jede Methode eine eindeutige SOAPAction definieren
- x|**WebService Framework dazu bringen, die selbst erstellte WSDL auszuliefern
(Axis: Parameter „wsdlFile“ in deploy.wsdd)

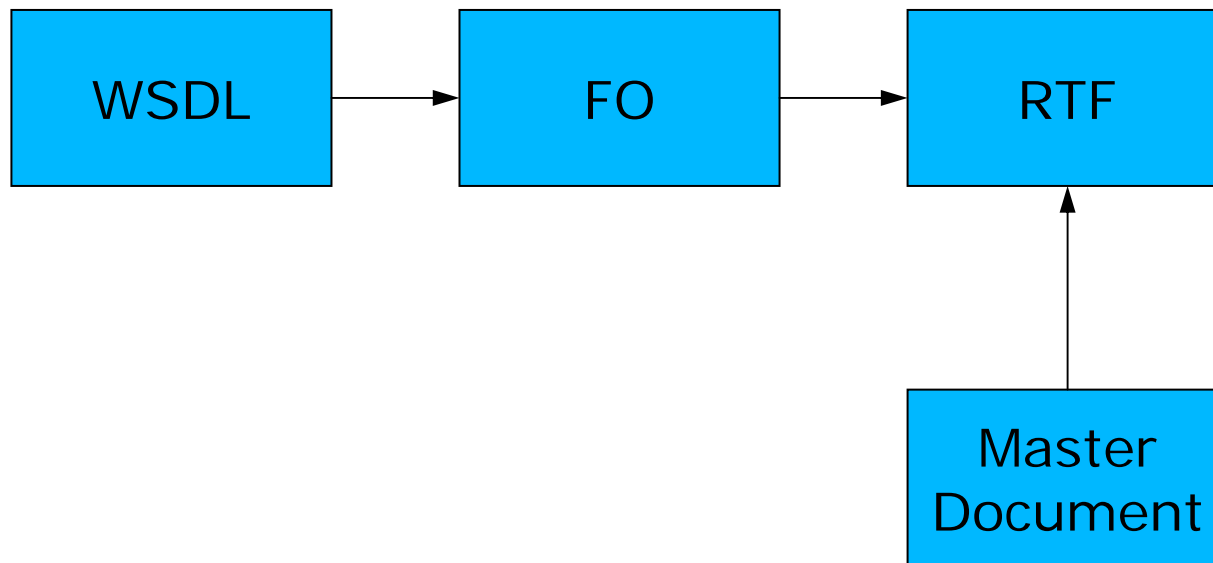
Dokumentation

- x| Dokumentation muss zur WSDL passen
→ es kommt nur ein Generator in Frage
- x| existierende Tools erfüllen unsere Wünsche
nicht ausreichend
- x| Wunsch des Kunden: Ein Word-Dokument

→ Implementation eines Doku-Generators

Doku-Generator

x|pragmatisch, wenig Aufwand, aber
arbeitserleichternd



Probleme / Schwachstellen

- x|**WebServices können nicht in Transaktionen eingebunden werden
- x|**Unterschiedliche Frameworks generieren unterschiedliche Klassen
- x|**Schema inline in WSDL erlaubt keine Wiederverwendung
- x|**kein Objektmodell (im Vergleich zu RMI)
- x|**proprietäre (Microsoft-) Erweiterungen aufgrund fehlender Spezifikationen

x|Viele Schema-Vorgaben werden nicht in Quellcode umgesetzt:

```
<complexType name="Order">  
  <attribute name="entity" type="tns:Entity"  
            use="required" />  
  
  ...  
  <attribute name="direction"  
            type="tns:OrderDirection"  
            default="Asc" />  
</complexType>
```

- x| Dilemma RPC Encoding und Document Encoding hat unterschiedliche Implementierungen der Encoding-Layer zur Folge
- x| Viele semantische Details in der WSDL führen zu unterschiedlich generierten Klassen
- x| Frameworks sind auch heute noch nicht von Trivial-Bugs befreit:
 - | Axis: Document + Void
 - | WebSphere: Document + leere Arrays, abstrakte Elemente

Performance- / Volumenbetrachtung

- x| Frameworks validieren nicht gegenüber dem Schema
→ Performancegewinn

- x| Axis-Timing für ConfigurationSearch-Aufruf:
axisServlet.doPost: ConfigurationSearchpre=0
invoke=31 post=0 send=47
DataServiceSoap.configurationSearch

Volumen der übertragenen Daten

- | Webservice: ~ 24 kb
- | Serialisiert: 12 kb

Resümee

- x|Definition eines WebServices ist komplexer als erwartet
- x|Interoperabilität nur mit Hürden
- x|Integration von Java und .NET im konkreten Projekt sehr zufriedenstellend
- x|Tool-Unterstützung zum Erstellen von WSDLs noch nicht ausreichend
- x|Einsatz-Szenarien genau abwägen
- x|WS-I weitet Arbeit aus (z.B. auf Security, Attachments, ...)

Diskussion / Fragen

Martin Renner

m.renner@exxcellent.de

+49731-55026-33