

Hochschule Darmstadt  
Fachbereich Informatik

# Daten- und Systemintegration

**SOA und Web Service Mediations Systeme - v 1.0**

Prof. Dr. Frank Bühler

Prof. Dr. Günter Turetschek

Quelle WMS/Synapse: Masterarbeit von Michael Buchholz (M.Sc.), Nico Tarsia (M.Sc.)

# Vorlesung „Daten- und Systemintegration“

## **Kapitel 6      SOA und Web Service Mediation Systeme**

- 6.1      Entwicklung eines WebService mit Oracle 10g (JDeveloper) und NetBeans (Glassfish)**
- 6.2      Begriffe und Überblick
- 6.3      Überblick WMS (Definition, Funktionalität, Anwendungsgebiete)
- 6.4      Erweitertes WMS
- 6.5      Synapse, prototypische Implementierung
- 6.6      JBI und WMS

# 6.1 JDeveloper – Entwicklung eines Web Service

## Ausgangspunkt „Java Class Diagramm“

The screenshot displays the Oracle JDeveloper 10g IDE. The main window shows a Java Class Diagram with two classes: a blue class box for «java class» mypackage WSTest with a method + printText(String p1):, and a green class box for «java web service» MyWSTestWS with a method printText(String p1): String. A dashed arrow points from the web service to the class. A yellow box with the text "Generate/Web Service" is overlaid on the diagram. The bottom of the window shows a log of generated files:

```
Generated WSDL document: file:/C:/OraBPELPM_1/integration/jdev/jdev/mywork/WebService_Demo/Project/src/mypackage/MyWSTestWS.wsdl
Generated Deployment Descriptor: {0}file:/C:/OraBPELPM_1/integration/jdev/jdev/mywork/WebService_Demo/Project/src/mypackage/MyWSTestWS.wsdl
Generated Interface: file:/C:/OraBPELPM_1/integration/jdev/jdev/mywork/WebService_Demo/Project/src/mypackage/MyWSTestWS.java
Generated Deployment Profile: file:/C:/OraBPELPM_1/integration/jdev/jdev/mywork/WebService_Demo/Project/src/mypackage/MyWSTestWS.xml
Generation complete.
```

# 6.1 JDeveloper – Entwicklung eines Web Service

## Definition Webservice – Implementierung der Java Klasse

```
package mypackage;
```

```
public class WSTest
```

```
{
```

```
/**
```

```
*
```

```
* @webmethod
```

```
*/
```

```
public String printText(String p1)
```

```
{
```

```
    String txt = new String("Hallo ");
```

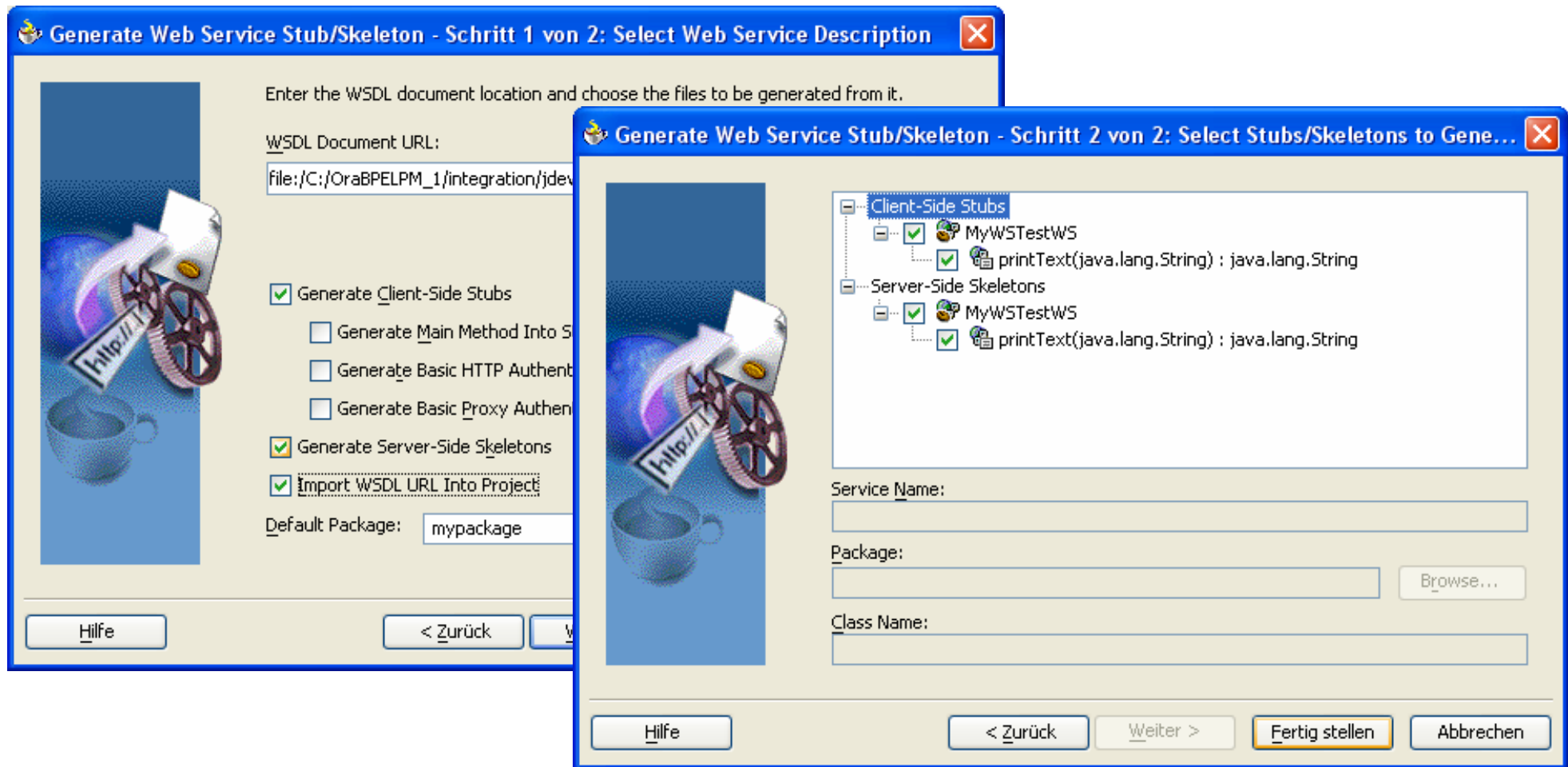
```
    return (new String(txt + p1));
```

```
}
```

```
}
```

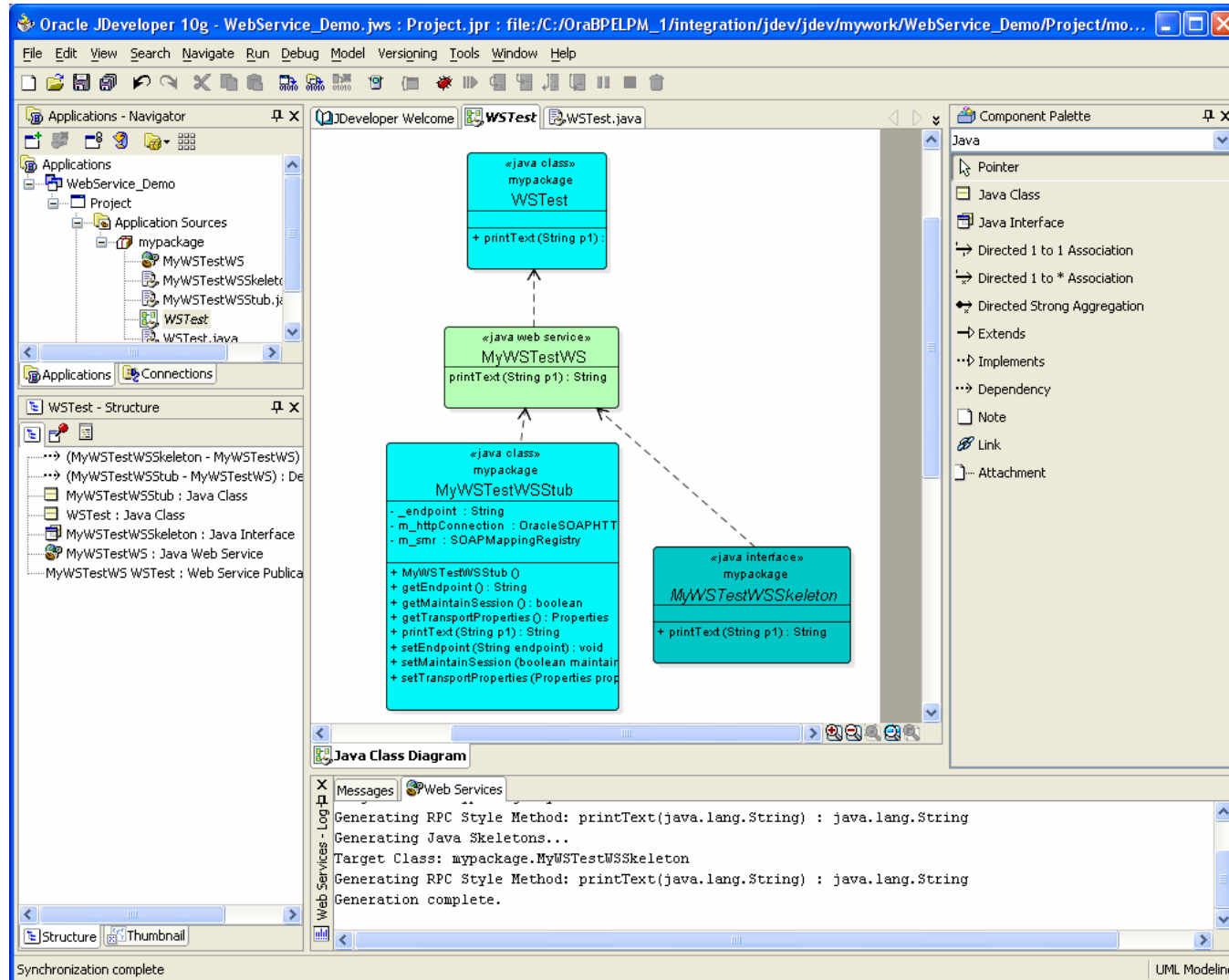
# 6.1 JDeveloper – Entwicklung eines Web Service

## Generierung des WebService



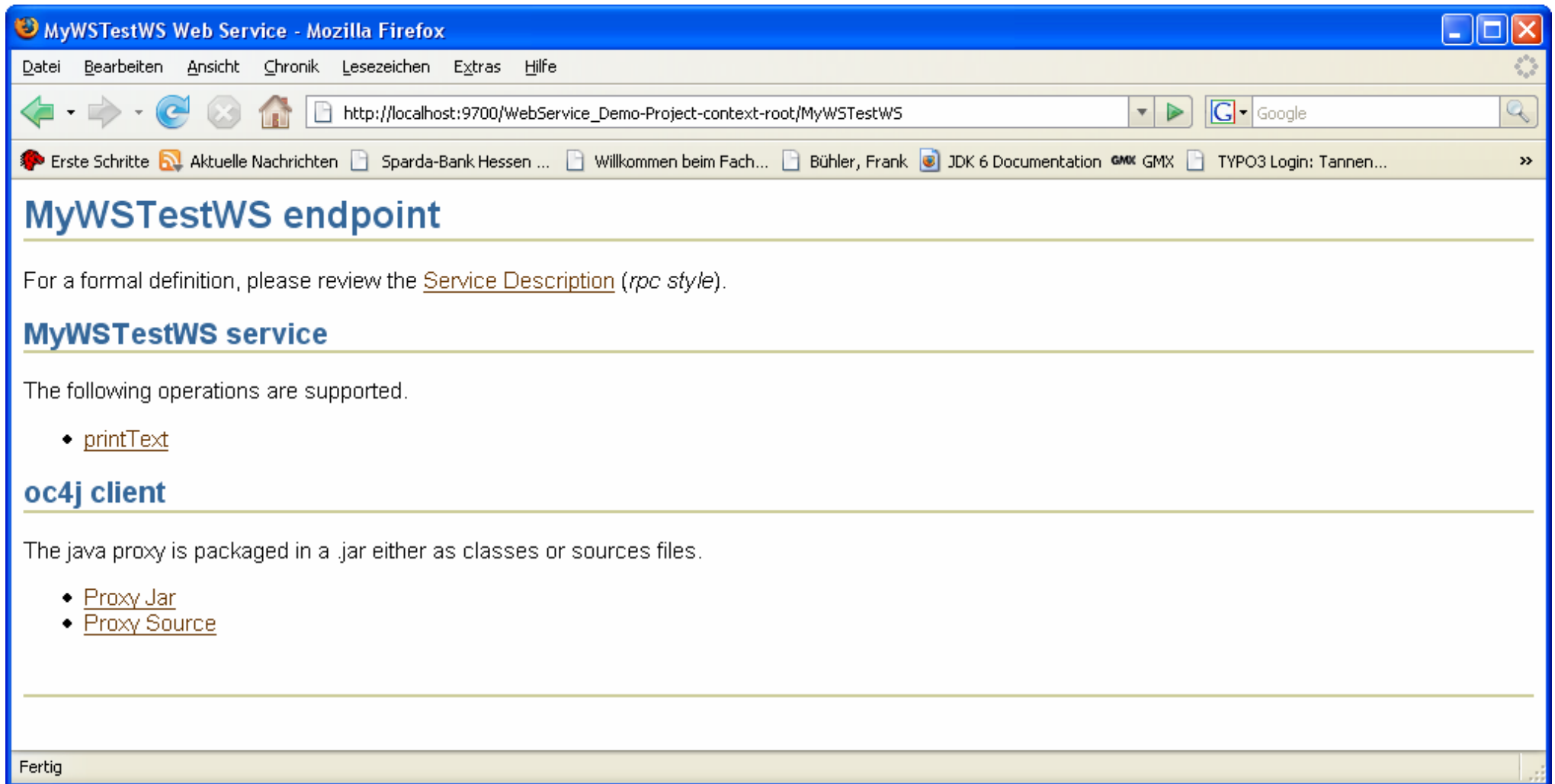
# 6.1 JDeveloper – Entwicklung eines Web Service

## Generierung und Deployment des WebService



# 6.1 JDeveloper – Entwicklung eines Web Service

## Aufruf des Webservice – 1(3)



# 6.1 JDeveloper – Entwicklung eines Web Service

## Aufruf des Webservice – 2(3)

MyWSTestWS Web Service - Mozilla Firefox

Click [here](#) for a complete list of operations.

### printText

### Test

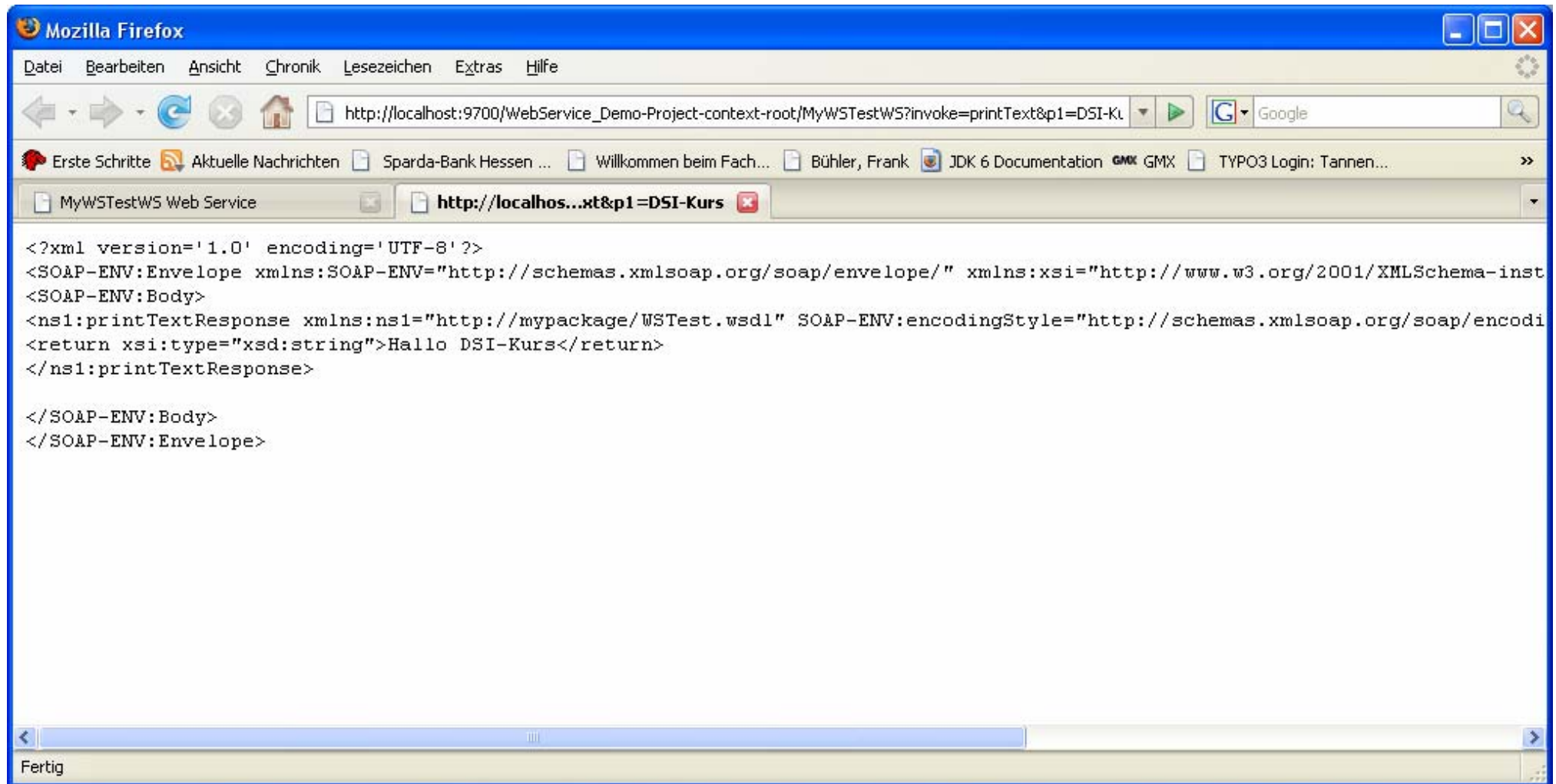
To test the operation using the HTTP GET protocol, click the 'Invoke' button.

Parameter	Type	Value
p1	string	<input type="text" value="DSI-Kurs"/>

Fertig

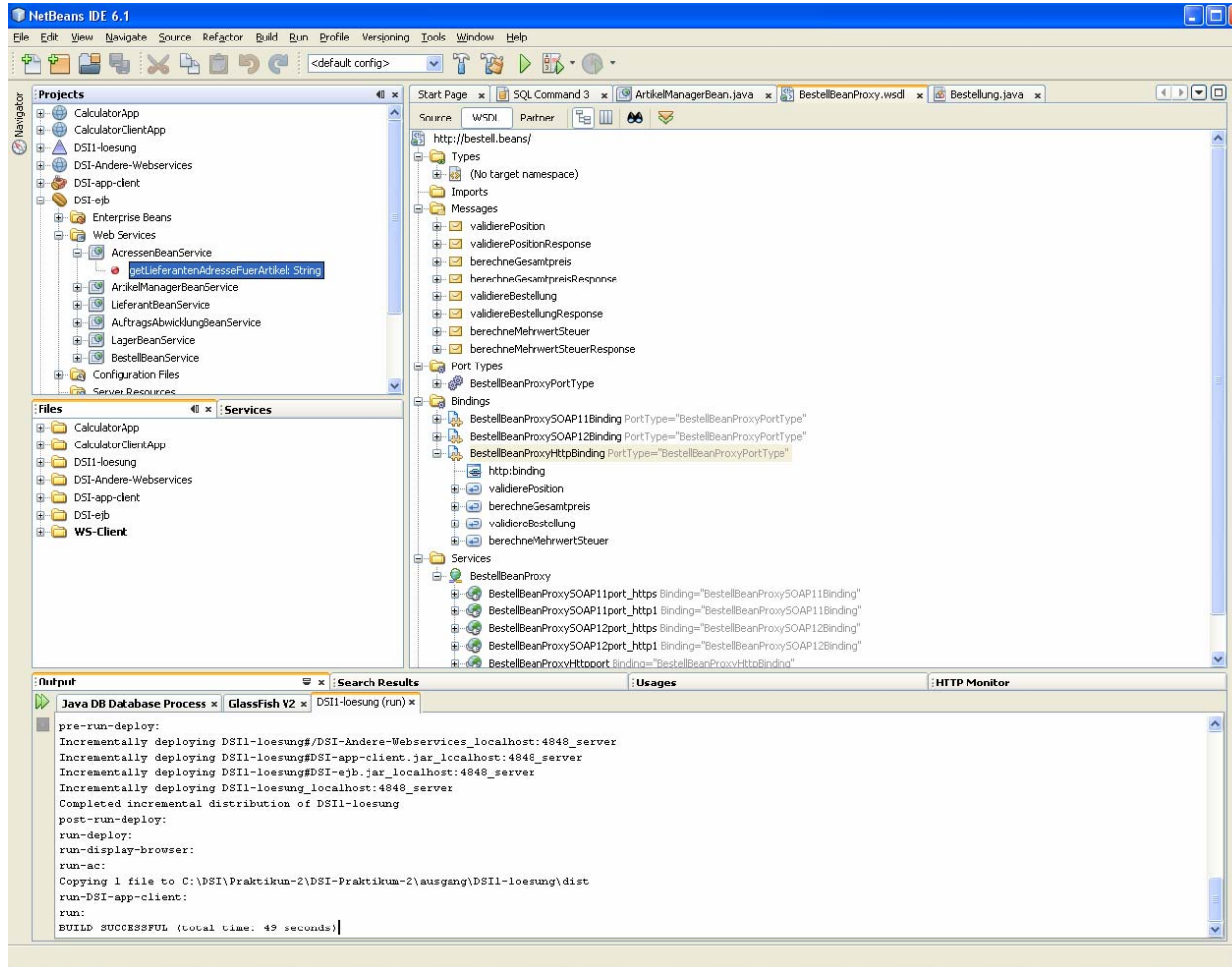
# 6.1 JDeveloper – Entwicklung eines Web Service

## Aufruf des Webservice – 3(3)



# 6.1 JDeveloper – Entwicklung eines Web Service

## Entwicklung eines Webservice mit NetBeans und Glassfish



# 6.1 JDeveloper – Entwicklung eines Web Service

## Testen des Webservice mit WebBrowser

Verfolgen des Methodenaufrufs - Mozilla Firefox

http://localhost:8080/AdressenBeanService/AdressenBean?Tester

### getLieferantenAdresseFuerArtikel Methodenaufruf

---

Method parameter(s)

Type	Value
long	1

---

Zurückgegebene Methode

java.lang.String: "Shimano, Merkelstr. 133, 12345 Dresden"

---

SOAP-Anforderung

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:getLieferantenAdresseFuerArtikel xmlns:ns2="http://adressen.beans/">
      <id>1</id>
    </ns2:getLieferantenAdresseFuerArtikel>
  </S:Body>
</S:Envelope>
```

---

SOAP-Antwort

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getLieferantenAdresseFuerArtikelResponse xmlns:ns2="http://adressen.beans/">
      <Adresse>Shimano, Merkelstr. 133, 12345 Dresden</Adresse>
    </ns2:getLieferantenAdresseFuerArtikelResponse>
  </S:Body>
</S:Envelope>
```

Fertig

# 6.1 JDeveloper – Entwicklung eines Web Service

## Testen des Webservice mit soapUI

The screenshot displays the soapUI 1.7.5 interface. On the left, a project tree shows the workspace structure, including a service named 'getLieferantenAdresseFuerArtikel' with a 'Request 1' test case selected. The main area is split into two panes: 'SOAP Request' and 'SOAP Response'. The request pane shows an XML envelope with a body containing a 'getLieferantenAdresseFuerArtikel' operation and an 'id' of '1'. The response pane shows an XML envelope with a body containing a 'getLieferantenAdresseFuerArtikelResponse' operation and an 'Adresse' of 'Shimano, Merkelstr. 133, 12345 Dresden'. Below the panes, a 'Request Properties' table is visible, and a log window at the bottom shows the execution details.

Property	Value
Name	Request 1
Description	
Message Size	307
Encoding	UTF-8
Endpoint	http://localhost:8080/AdressenB...
Bind Address	
Username	

```
SOAP Request: <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Header/><soapenv:Body><adr:getLieferantenAdresseFuerArtikel><id>1</id></adr:getLieferantenAdresseFuerArtikel></soapenv:Body></soapenv:Envelope>
```

```
SOAP Response: <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"><S:Body><ns2:getLieferantenAdresseFuerArtikelResponse xmlns:ns2="http://adresses.beans/"><Adresse>Shimano, Merkelstr. 133, 12345 Dresden</Adresse></ns2:getLieferantenAdresseFuerArtikelResponse></S:Body></S:Envelope>
```

Log output:  
Mon Sep 08 12:02:35 CEST 2008:INFO:Finding importer for {http://adresses.beans/}AdressenBeanPortBinding  
Mon Sep 08 12:02:35 CEST 2008:INFO:Importing binding {http://adresses.beans/}AdressenBeanPortBinding  
Mon Sep 08 12:02:35 CEST 2008:INFO:importing endpoint http://localhost:8080/AdressenBeanService/AdressenBean  
Mon Sep 08 12:02:35 CEST 2008:INFO:importing operation getLieferantenAdresseFuerArtikel  
Mon Sep 08 12:03:00 CEST 2008:INFO:Got response for [AdressenBeanPortBinding.getLieferantenAdresseFuerArtikel:Request 1] in 216ms (301 bytes)

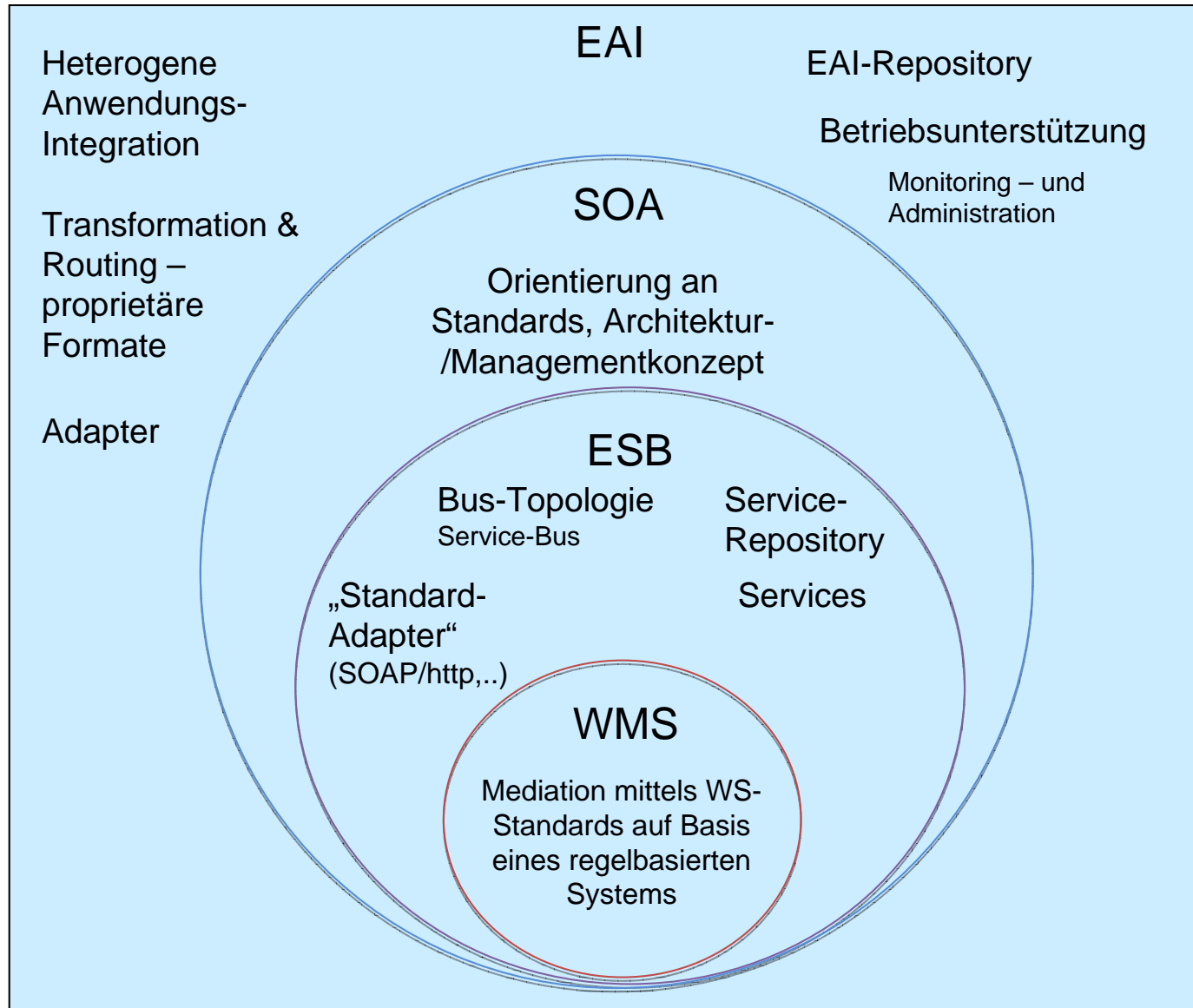
# Vorlesung „Daten- und Systemintegration“

## **Kapitel 4      SOA und Web Service Mediation Systeme**

- 6.1      Entwicklung eines Webservice mit Oracle 10g (JDeveloper)
- 6.2      **Begriffe und Überblick**
- 6.3      Überblick WMS (Definition, Funktionalität, Anwendungsgebiete)
- 6.4      Erweitertes WMS
- 6.5      Synapse, prototypische Implementierung
- 6.6      JBI und WMS

# 6.2 Begriffe und Überblick

## EAI, SOA, ESB & WMS – eine mögliche Einordnung



## 6.2 Begriffe und Überblick

### Definition ESB

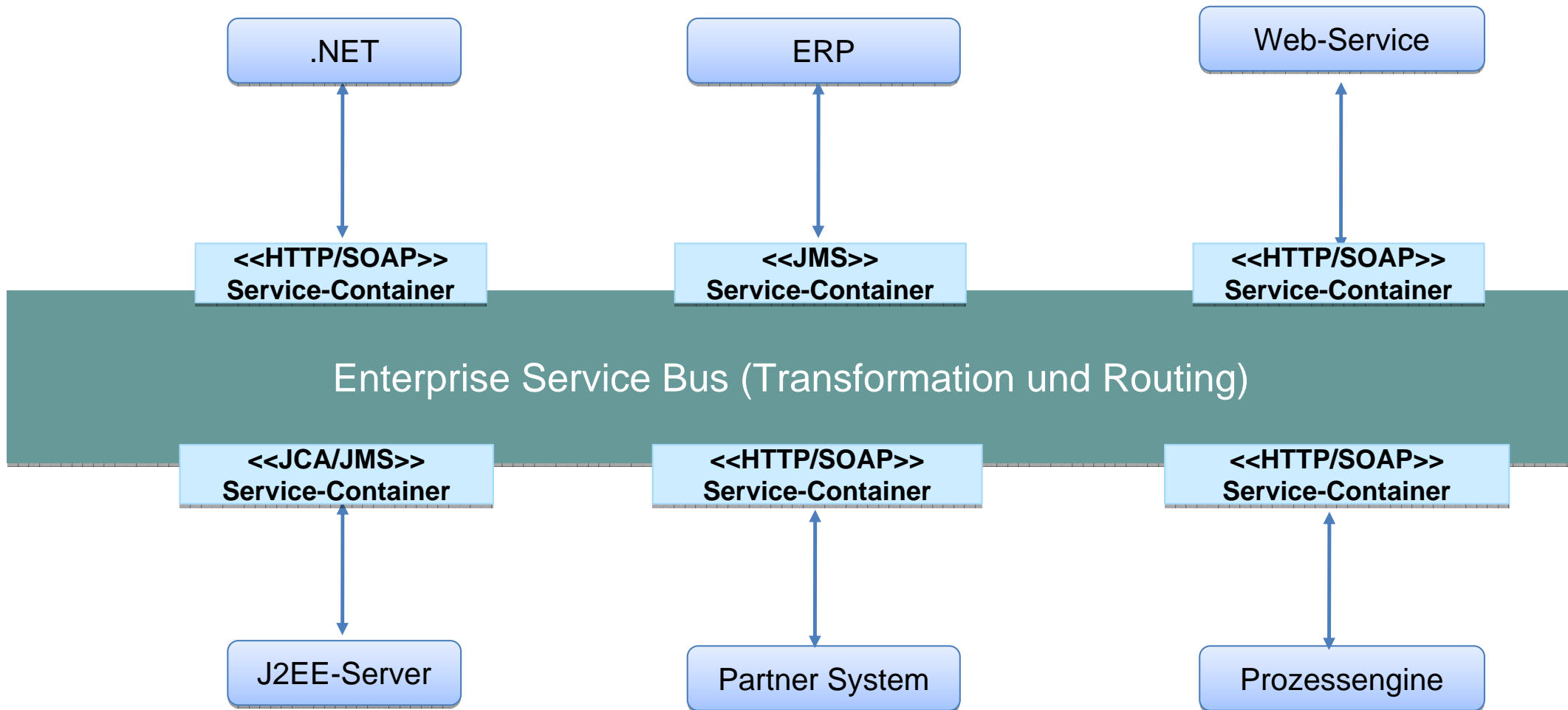
- Kommunikationsinfrastruktur für eine SOA
- Transport- und protokollunabhängiger Zugriff auf IT-Assets mittels standardbasierter Integration, Mediation von einheitlichen XML-Nachrichten auf dem Service-Bus

### ESB-Komponenten

- **Service-Container**  
Standardbasierte Integration
- **Service-Bus**  
Nachrichtentransport
- **Mediationskomponente**  
Transformation & Routing (CBR)
- **Repository/Registry**  
Für interne Services und ESB-Konfigurationen
- **Administration- und Monitoringkomponenten**  
Verwaltung und Überwachung

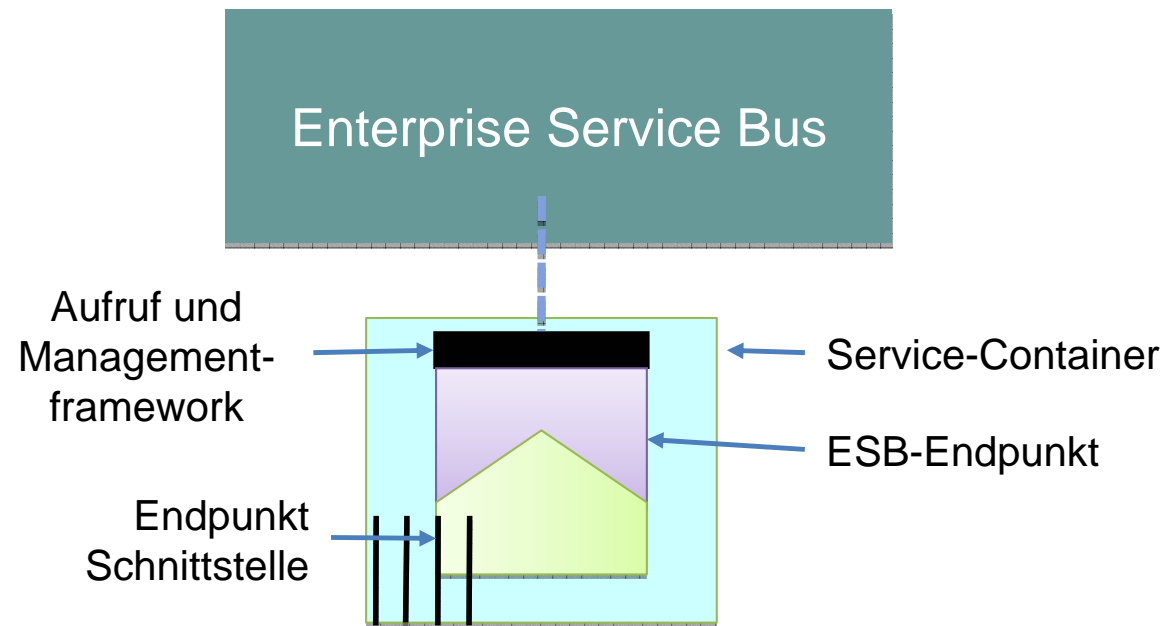
# 6.2 Begriffe und Überblick

## Enterprise Service Bus (ESB)



## 6.2 Begriffe und Überblick

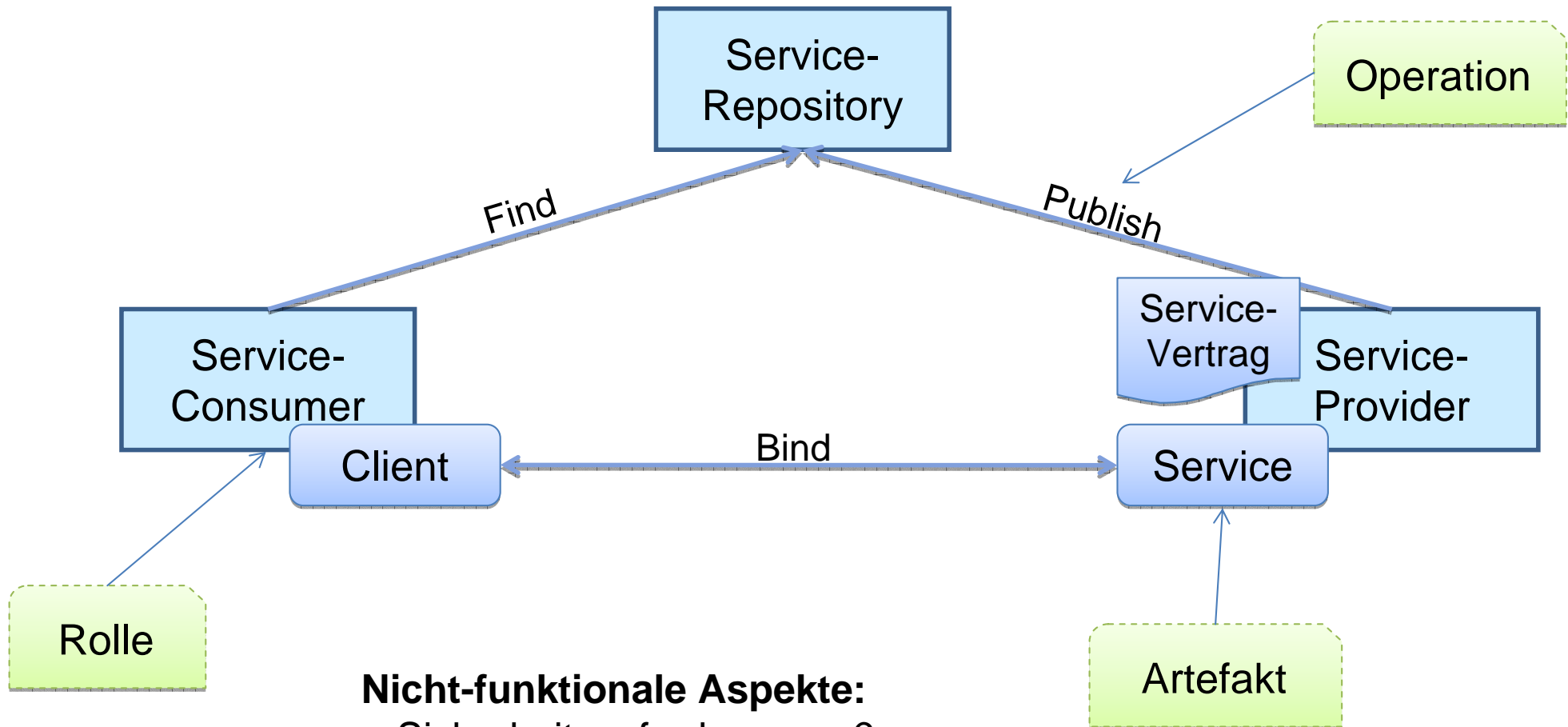
### Enterprise Service Bus (ESB)



ESB := Generischer Service-Container

# 6.2 Begriffe und Überblick

## Service-orientierte Architektur

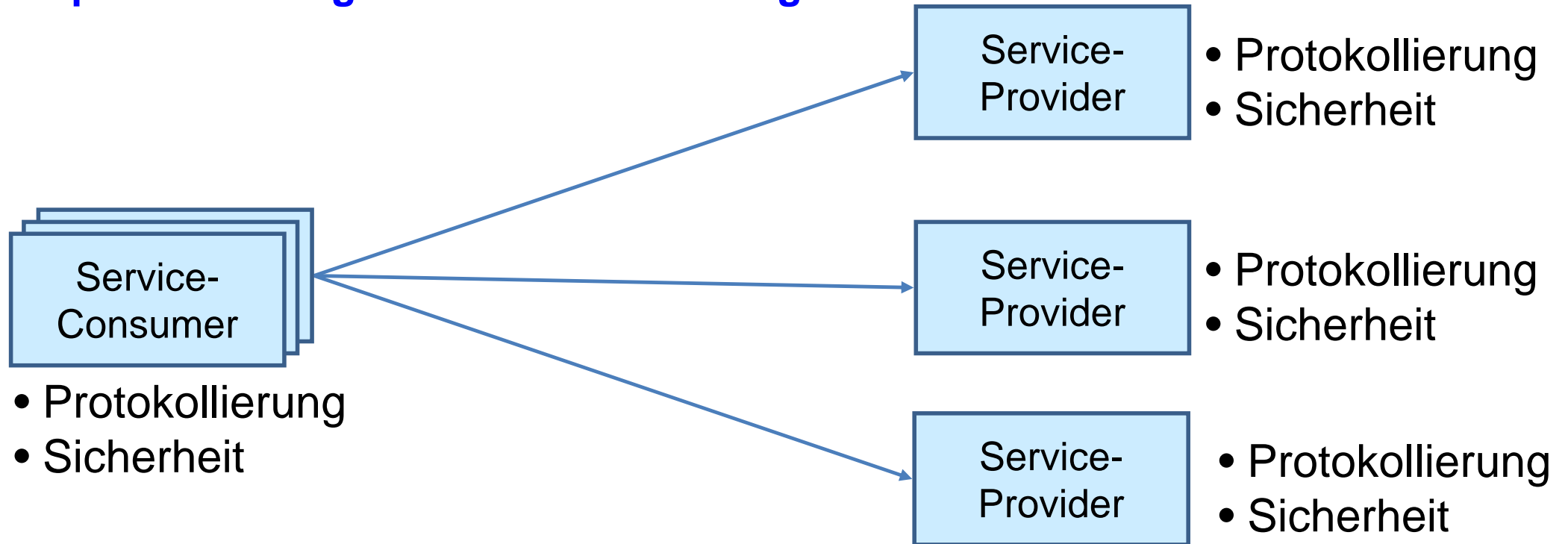


### Nicht-funktionale Aspekte:

- Sicherheitsanforderungen?
- Nachvollziehbarkeit?
- Wartbarkeit, Änderbarkeit?

## 6.2 Begriffe und Überblick

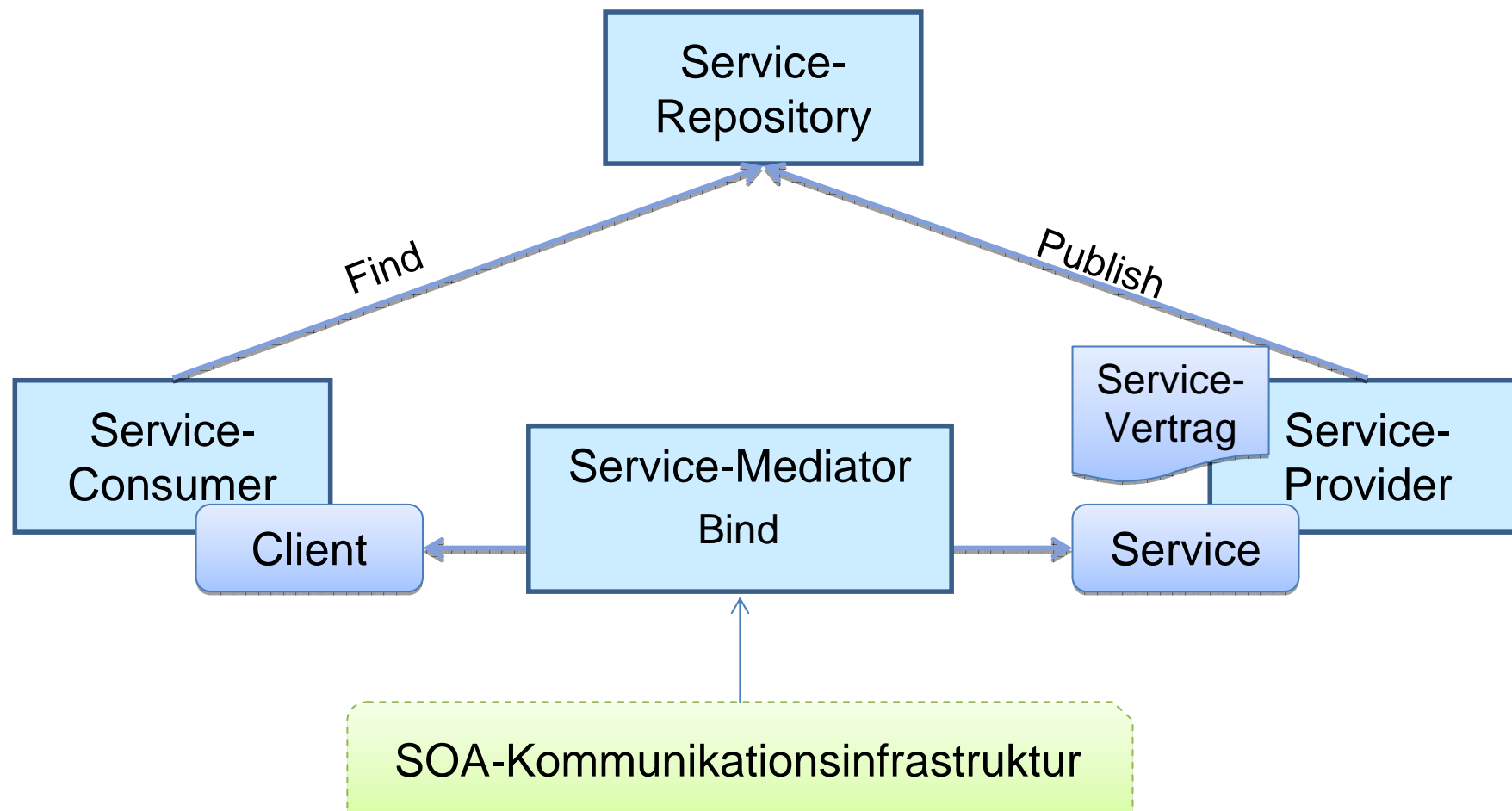
### Implementierung von QoS-Anforderungen



Probleme: Redundante Implementierungen bei vielen Clients  
Schlechte Wartbarkeit & Änderbarkeit

## 6.2 Begriffe und Überblick

### SOA mit neuer Rolle „Service-Mediator“



# Vorlesung „ Daten- und Systemintegration“

## **Kapitel 4      SOA und Web Service Mediation Systeme**

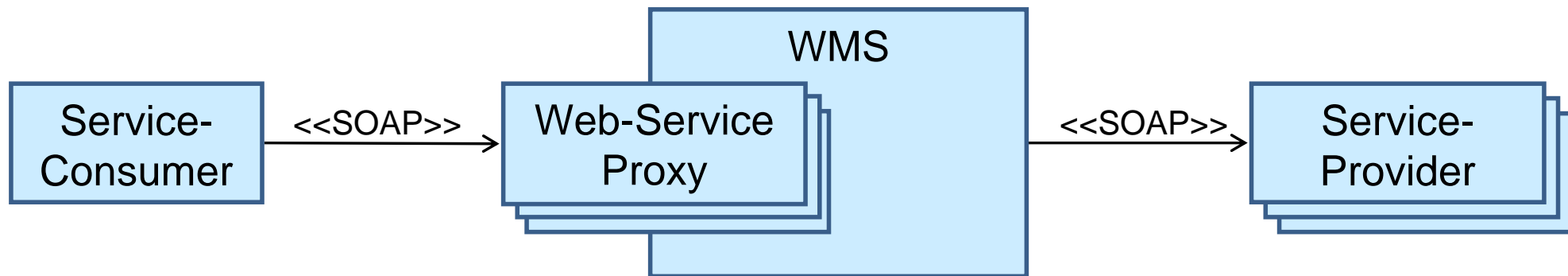
- 6.1      Entwicklung eines Webservice mit Oracle 10g (JDeveloper)
- 6.2      Begriffe und Überblick
- 6.3      **Überblick WMS (Definition, Funktionalität, Anwendungsgebiete)**
- 6.4      Erweitertes WMS
- 6.5      Synapse, prototypische Implementierung
- 6.6      JBI und WMS

## 6.3 Überblick WMS

### Definition WMS

Das WMS (WebService Mediation System) stellt die notwendige SOA-Kommunikationsinfrastruktur auf Basis von Web-Service Standards bereit. Kernfunktionalitäten ist die Mediation von SOAP-Nachrichten

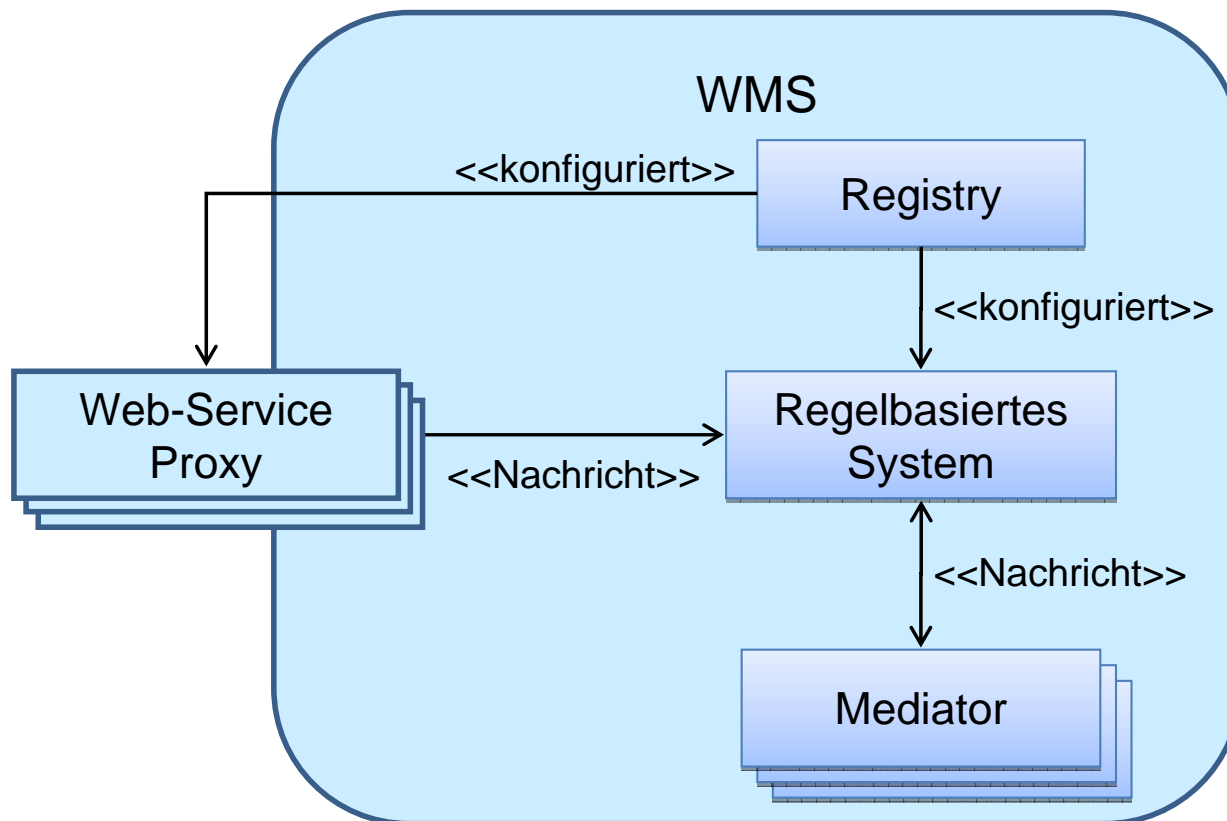
- fachliche Transformation
- explizites und implizites (CBR)



WMS: Kommunikationsinfrastruktur für SOA

## 6.3 Überblick WMS

### WMS-Aufbau



### WMS- Funktionalität

- Bereitstellung von Service-Endpunkten
- Routing von SOAP-Nachrichten
- Validierung von Nachrichten
- Fachliche Transformation
- Protokollierung
- Lastverteilung
- Fehlerbehandlung
- Dynamische Konfiguration

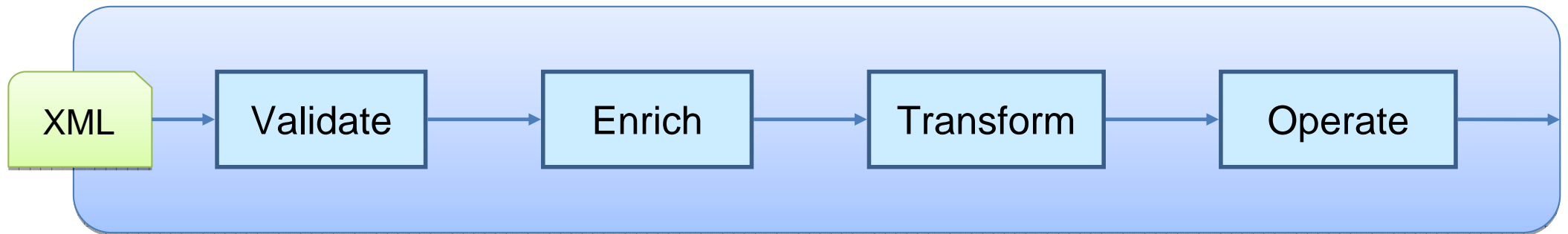
## 6.3 Überblick WMS

### Typische Mediatoren im Überblick

- **Definition von Endpunkten**  
Lastverteilung / Fail-over / WSDL
- **Transformation per XSLT**  
Sowohl Inhalt als auch Header
- **Validierung**
- **Entscheidungen per XPATH (Filter)**
- **Protokollierung**

## 6.3 Überblick WMS

### WMS: VETO-Pattern (Mediation)



### Ablauf

Explizites Routing

Validate: Schemavalidierung

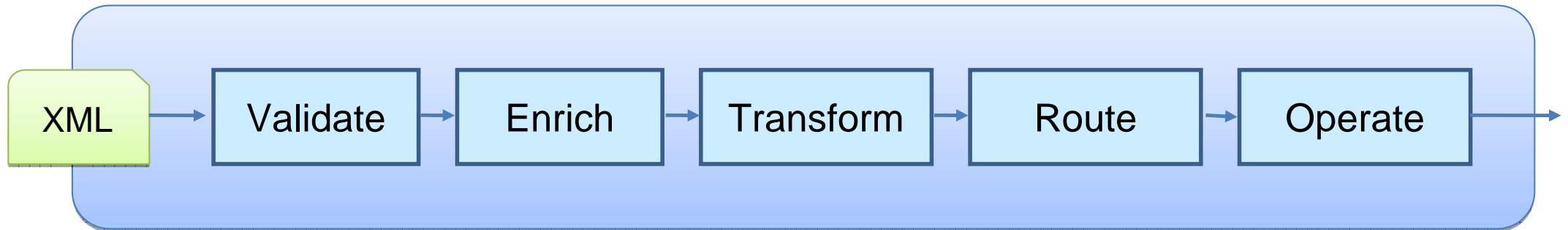
Enrich: Anreicherung mit externen Daten

Transform: In Zielformat (Adapter)

Operate: Aufruf des Zielsystems (Adapter)

## 6.3 Überblick WMS

WMS: Variation: VETRO-Pattern (Mediation)



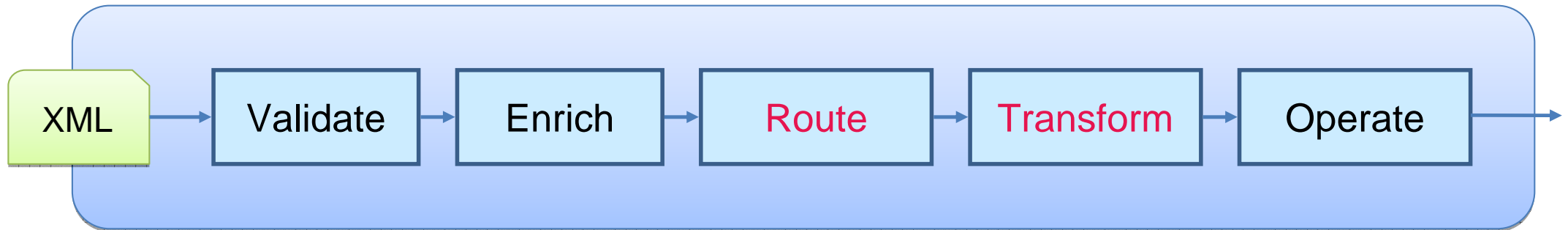
### Implizites Routing (CBR)

Kritik bei WMS-Anwendung:

- Nicht anwendbar bei mehreren Endpunkten mit verschiedenen Zielformaten
- Transformation nach Route und vor Operate nötig!

## 6.3 Überblick WMS

WMS: Variation: VERTO-Pattern (Mediation)

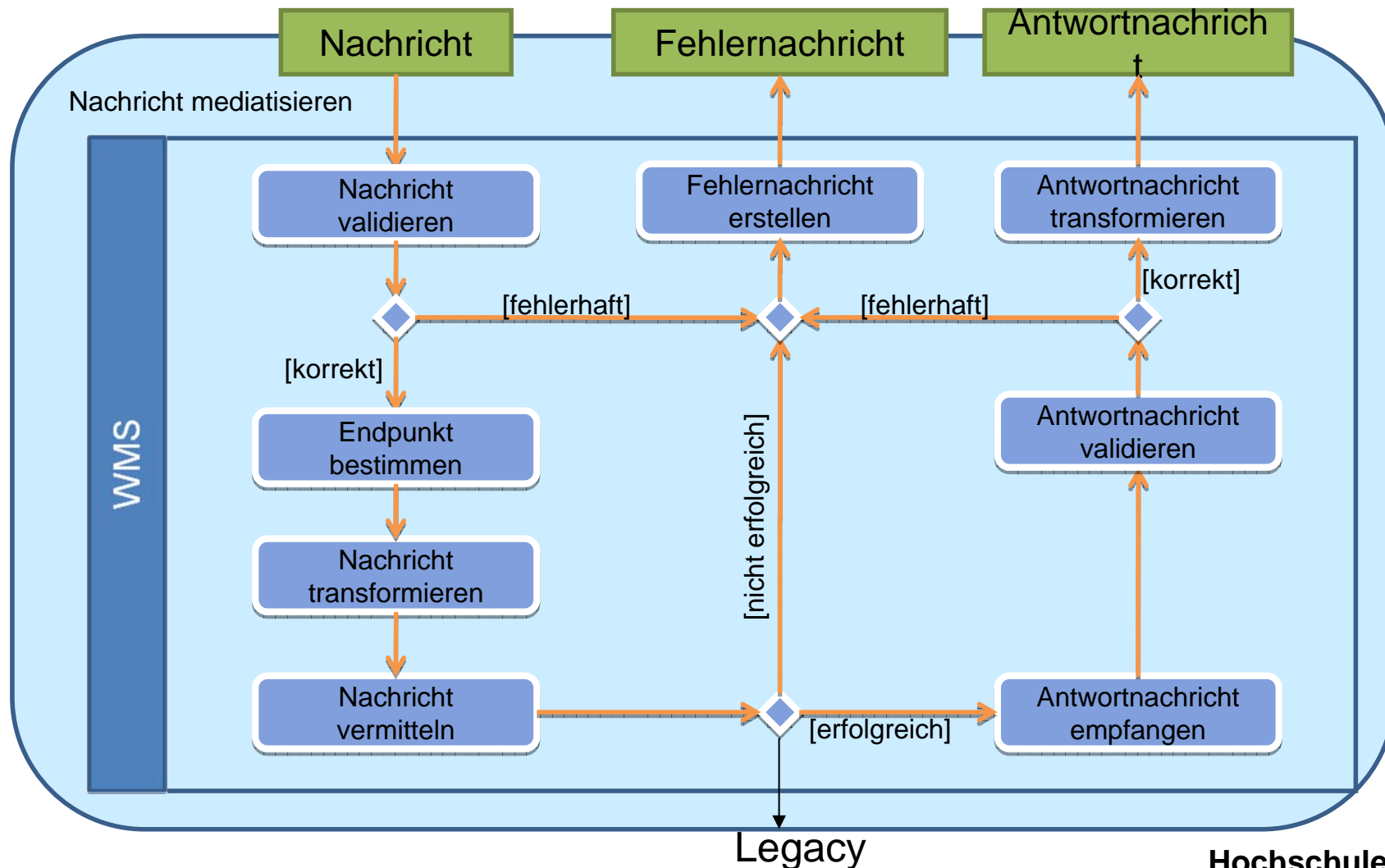


### Tausch von Route & Transform

Nach Routing-Entscheidung wird die Nachricht an das Zielformat des gewählten Endpunkts angepasst.

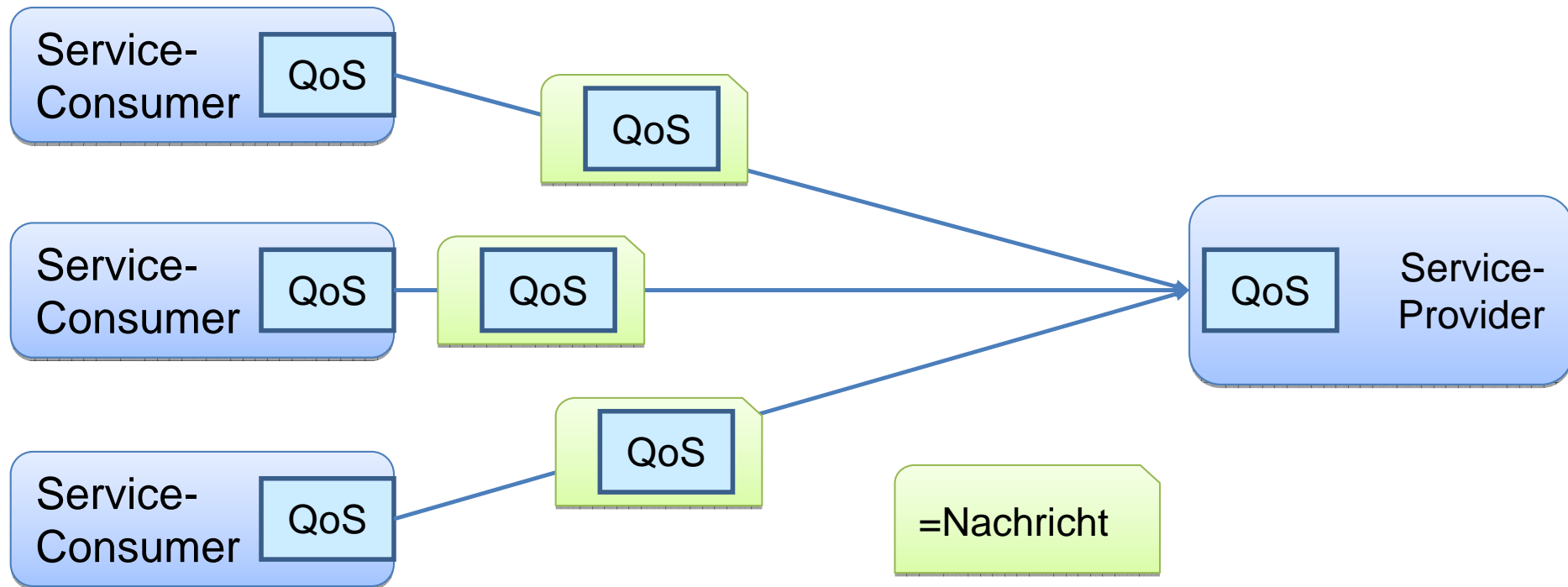
# 6.3 Überblick WMS

## Nachricht mediatisieren - Mediatoren



## 6.3 Überblick WMS

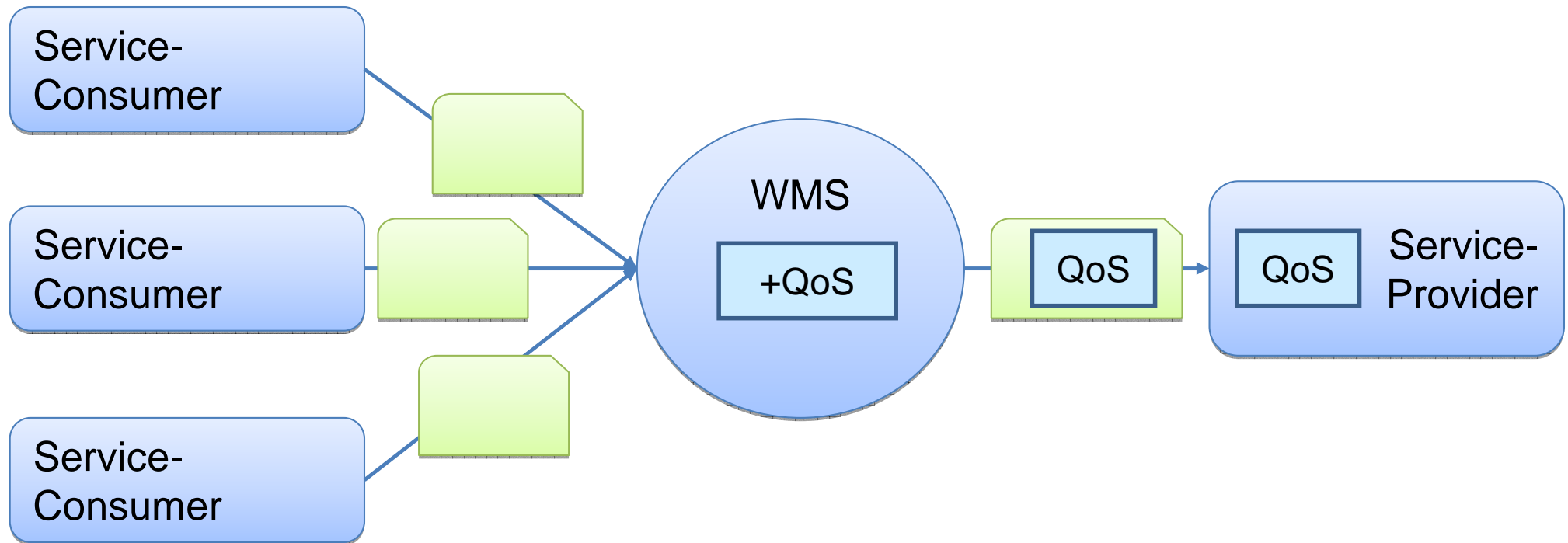
### Kommunikation mit Erfüllung bestimmter QoS-Anforderungen



## 6.3 Überblick WMS

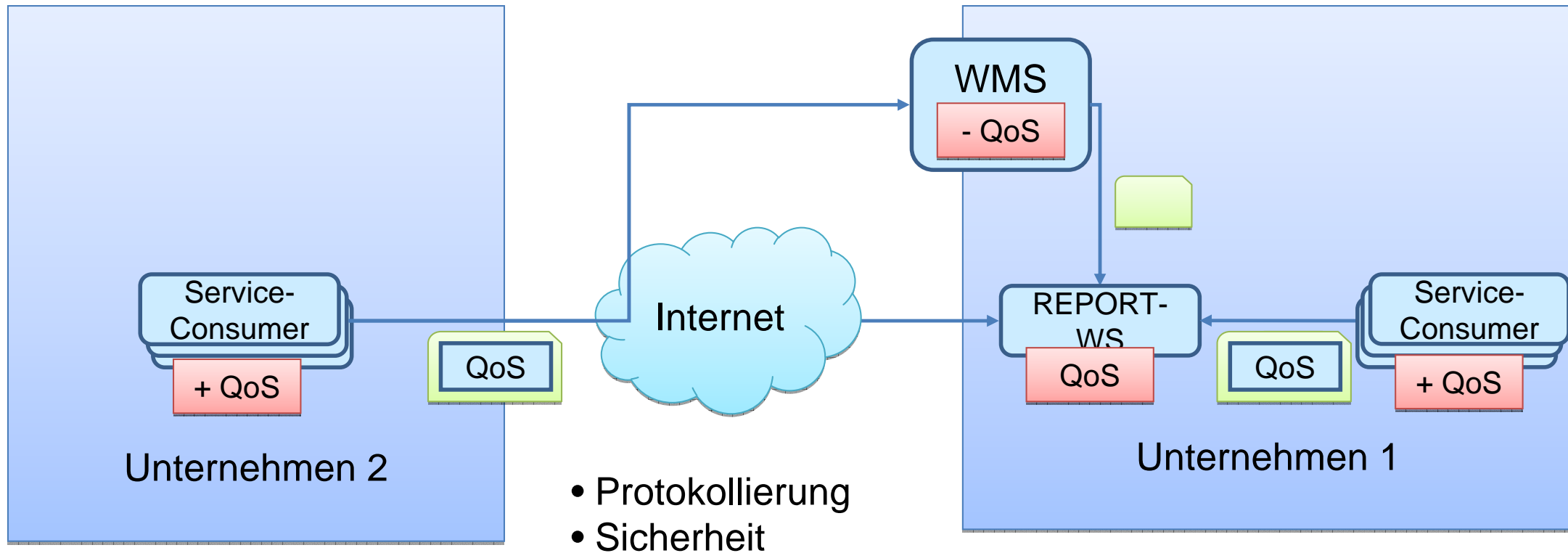
### Kommunikation mit QoS

WMS sorgt für Einhaltung der QoS



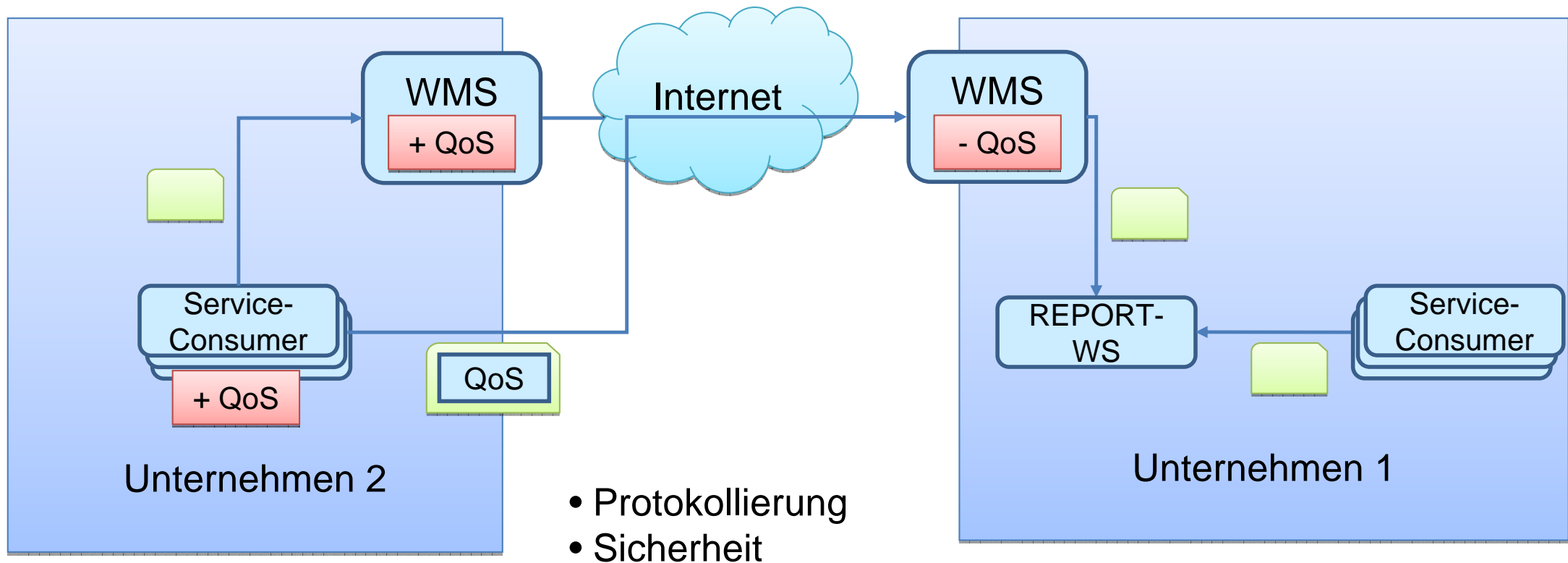
## 6.3 Überblick WMS

### Szenario: Unternehmensexterne Service-Aufrufe – 1(2)



## 6.3 Überblick WMS

### Szenario: Unternehmensexterne Service-Aufrufe – 2(2)



## 6.3 Überblick WMS

### Integration von WMS in vorhandene IT-Infrastrukturen

#### Integrationsbetrachtungen WMS bzgl. IT-Infrastruktur und Langfristiges Ziel (SOA-Initiative)

##### **SOA-Levels:**

- 1: Erste SOA-Initiative
- 2: Integrierte SOA-Anwendungen
- 3: SOA wird zum „strategischen Business Service“

#### **Auf welchen Leveln kann WMS unterstützen?**

## 6.3 Überblick WMS

### Integration von WMS in vorhanden IT-Infrastrukturen

#### **Level1:**

Prototypische Implementierungen erlauben kurze Entwicklungszyklen  
Erlangen von fachlichem Wissen über WS-\* Standards und Mediation

#### **Level2:**

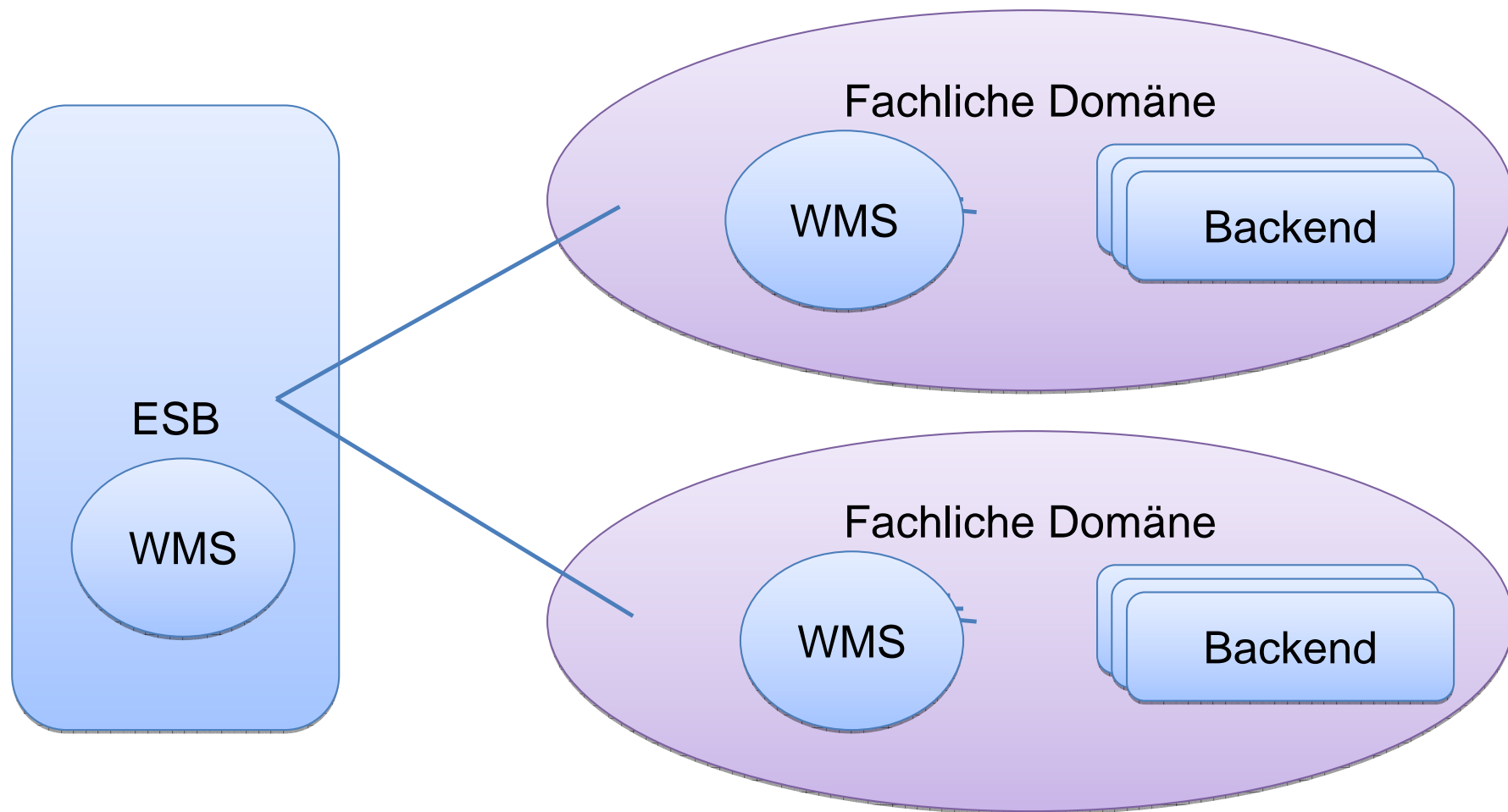
Ausbauen der prototypischen Projekte  
Jede fachliche Domäne ein WMS  
Gesamte Mediationsregeln der Domänen in WMS

#### **Level 3:**

Unternehmensweiter ESB  
Zentralisierung der WMS Regelwerke in den ESB oder belassen der WMS  
Zusammenführung der fachlichen Domänen

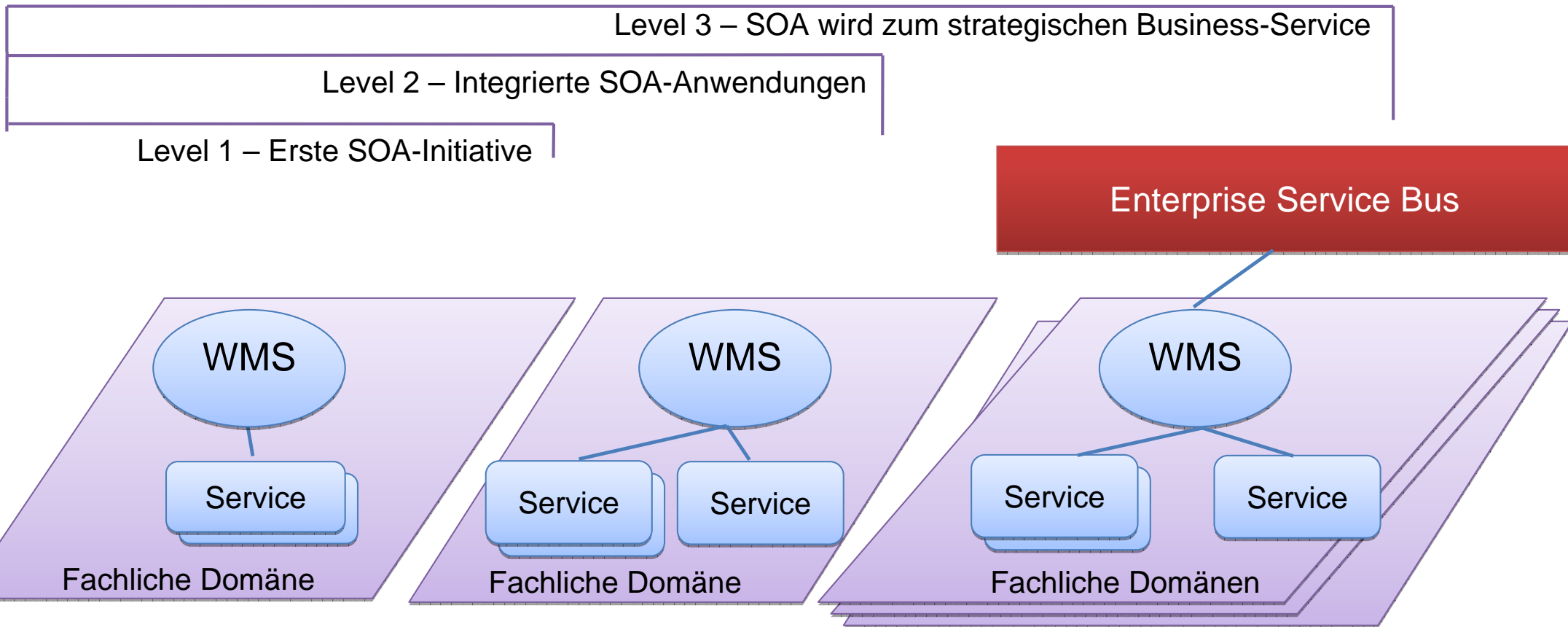
## 6.3 Überblick WMS

### Einsatz von WMS und ESB in Unternehmen



# 6.3 Überblick WMS

## Einsatz von WMS und ESB in Unternehmen



# Vorlesung „ Daten- und Systemintegration“

## **Kapitel 4      SOA und Web Service Mediation Systeme**

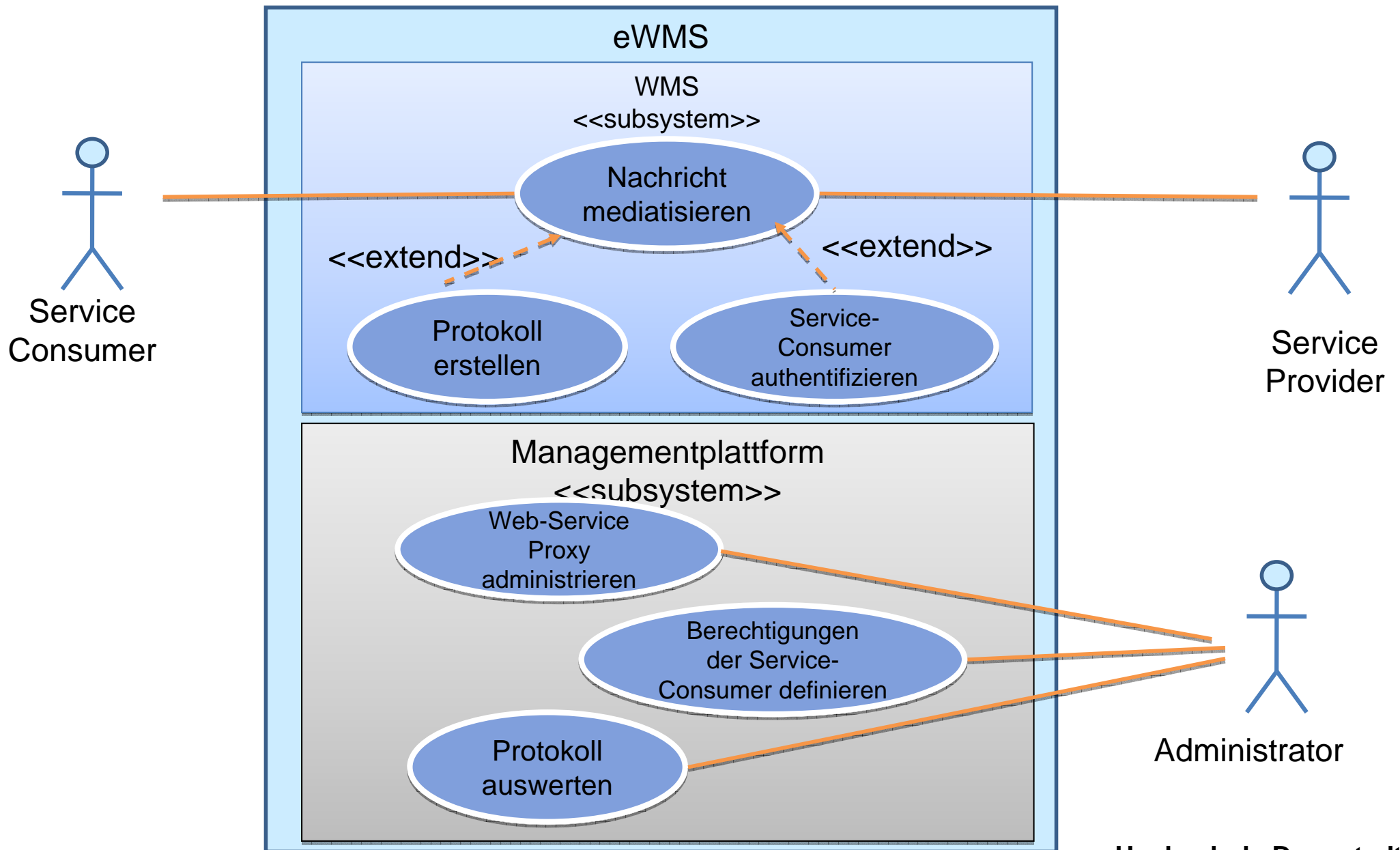
- 6.1      Entwicklung eines Webservice mit Oracle 10g (JDeveloper)
- 6.2      Begriffe und Überblick
- 6.3      Überblick WMS (Definition, Funktionalität, Anwendungsgebiete)
- 6.4      **Erweitertes WMS**
- 6.5      Synapse, prototypische Implementierung
- 6.6      JBI und WMS

## 6.4 Erweitertes WMS

### **Gestellte Anforderungen an das eWMS**

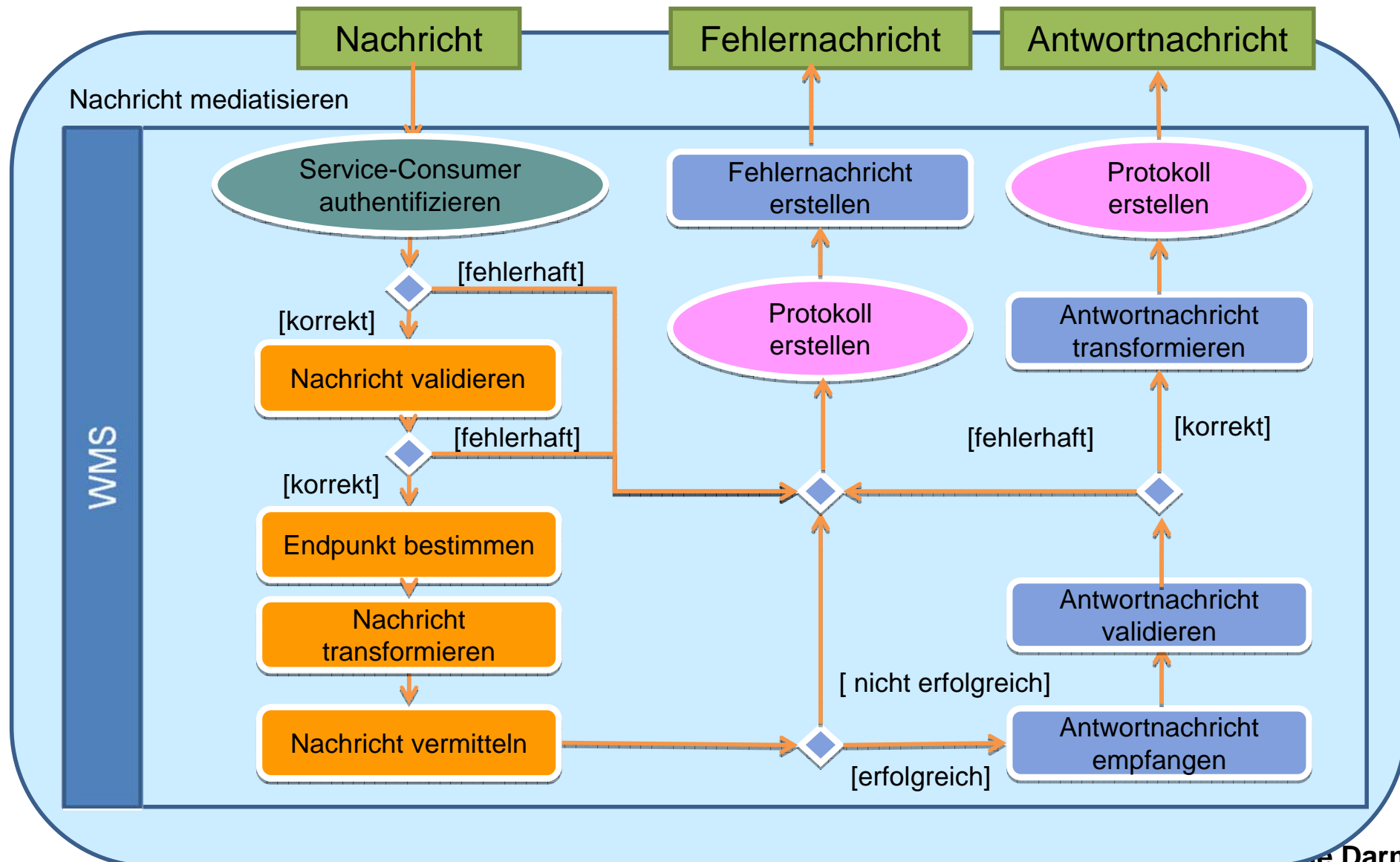
- Sichere Nachrichtenvermittlung
- Authentifizierung der Service-Consumer
- Validierung von Nachrichteninhalten
- Protokollierung von Nachrichtenmediationen
- Auswertung der Protokolle
- Publizierung von Services in ein UDDI
- Dynamische Verwaltung von Services zur Laufzeit

# 6.4 Erweitertes WMS



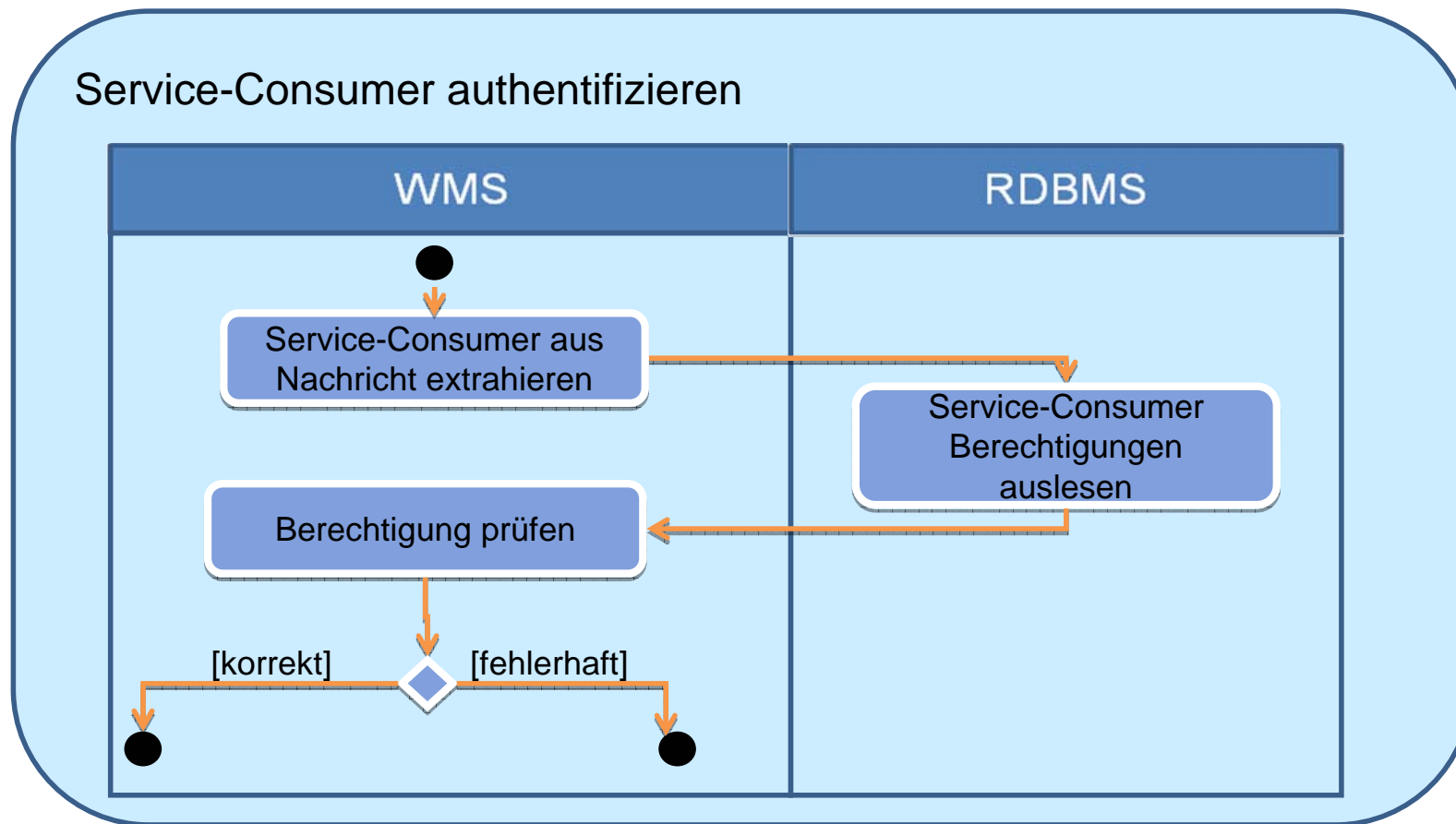
# 6.4 Erweitertes WMS

## eWMS: „Nachricht mediatisieren“ (VERTO-Pattern)



## 6.4 Erweitertes WMS

### eWMS: „Service Consumer authentifizieren“



# Vorlesung „ Daten- und Systemintegration“

## **Kapitel 4      SOA und Web Service Mediation Systeme**

- 6.1      Entwicklung eines Webservice mit Oracle 10g (JDeveloper)
- 6.2      Begriffe und Überblick
- 6.3      Überblick WMS (Definition, Funktionalität, Anwendungsgebiete)
- 6.4      Erweitertes WMS
- 6.5      Synapse, prototypische Implementierung**
- 6.6      JBI und WMS

## 6.5 Synapse

### **Eigenschaften**

- Web-Service Mediation Framework  
Nachrichten und Kommunikationsinfrastruktur (aufbauend auf den SOA Prinzipien)
- Management der Nachrichten und Verbindungen zwischen den angeschlossenen Systemen
- Vermitteln und Umwandlung von Nachrichten und Serviceinteraktionen unabhängig des Endpunktes
- Primäre Unterstützung der Sprachen C/C++/COBOL/Java/.NET

### **Konnektivität**

- Dynamisches Finden /Binden, Load-Balancing und Verfügbarkeit
- Zentrales Gateway und Routing von Web services
- Herstellen von statischen und dynamischen Verbindungen zwischen Requester und Provider
- Schnittstellen für Adapter zu schon bestehenden Systemen(Adapter), dazu gehören schon JMS und JCA
- Möglichkeit neue Protokolle einfach hinzuzufügen

## 6.5 Synapse

### **Transformation and Mediation**

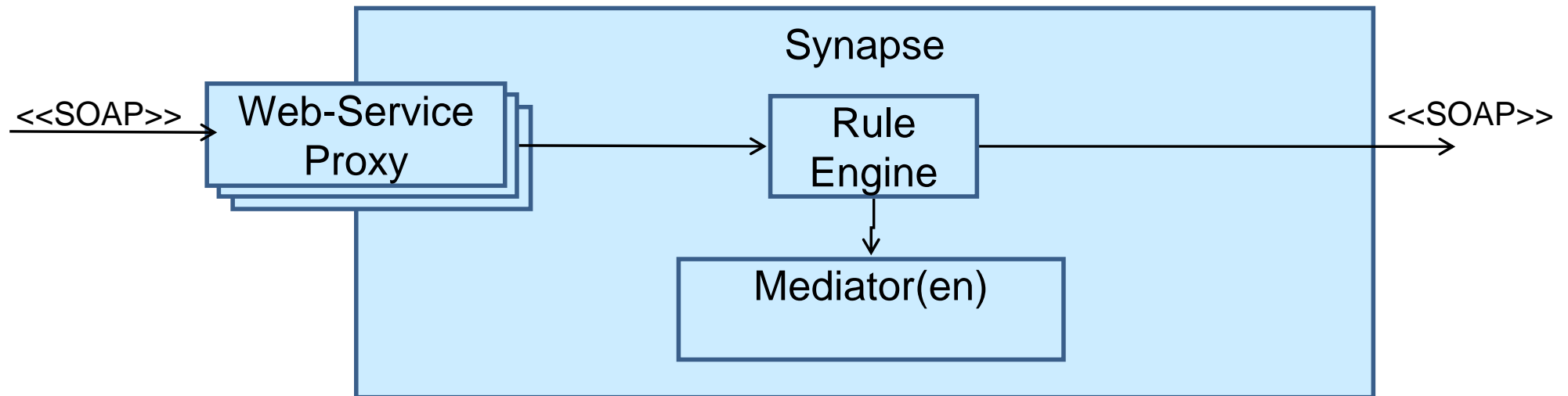
- Unterstützung div. “mediation models”, wie z. B. XSLT, XPATH, Java-Transformation

### **Management**

- Logging und Tracing
- Performancemessungen
- Ausfallerkennung, ...

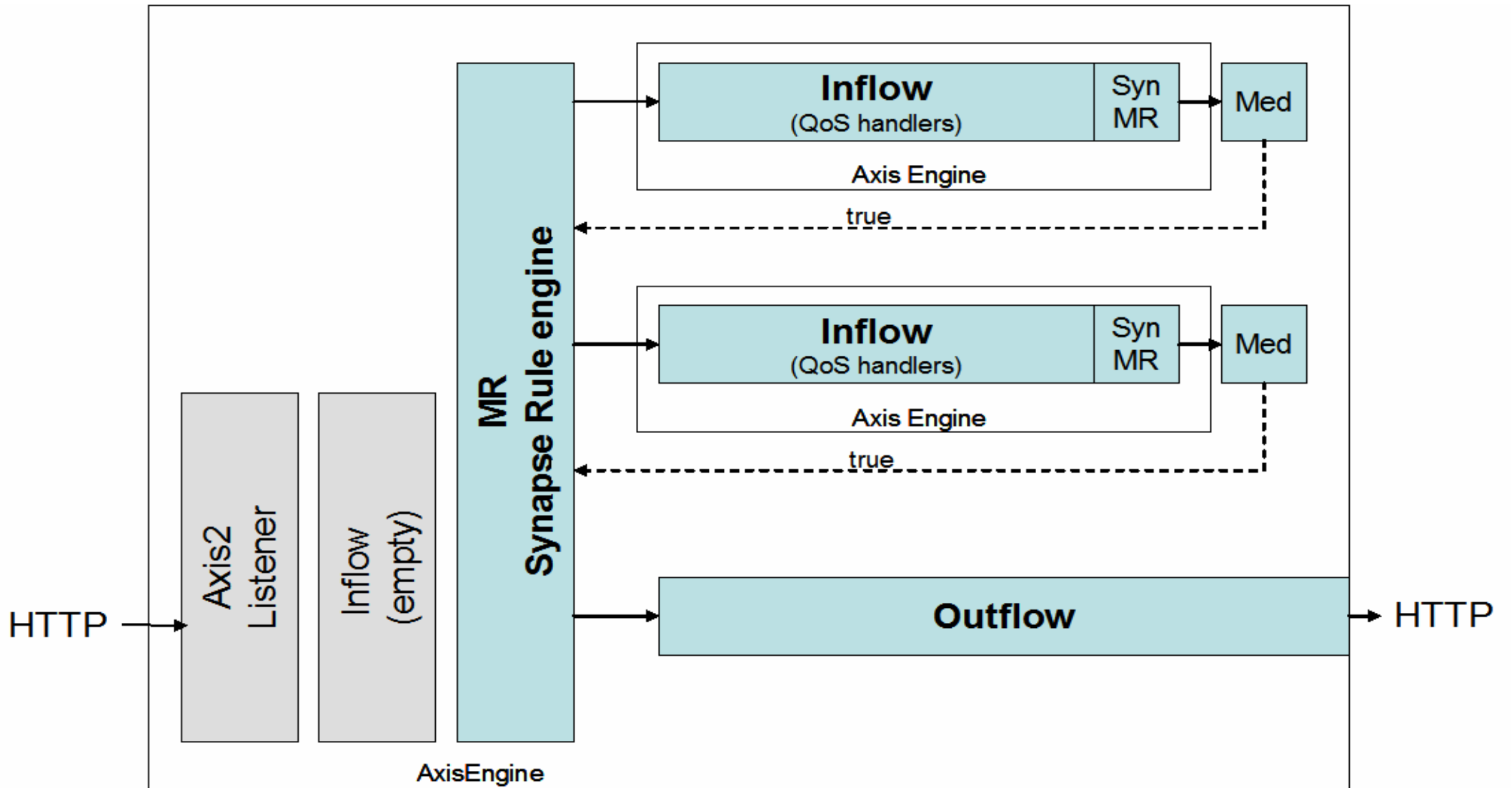
# 6.5 Synapse

## Systemkomponenten von Synapse

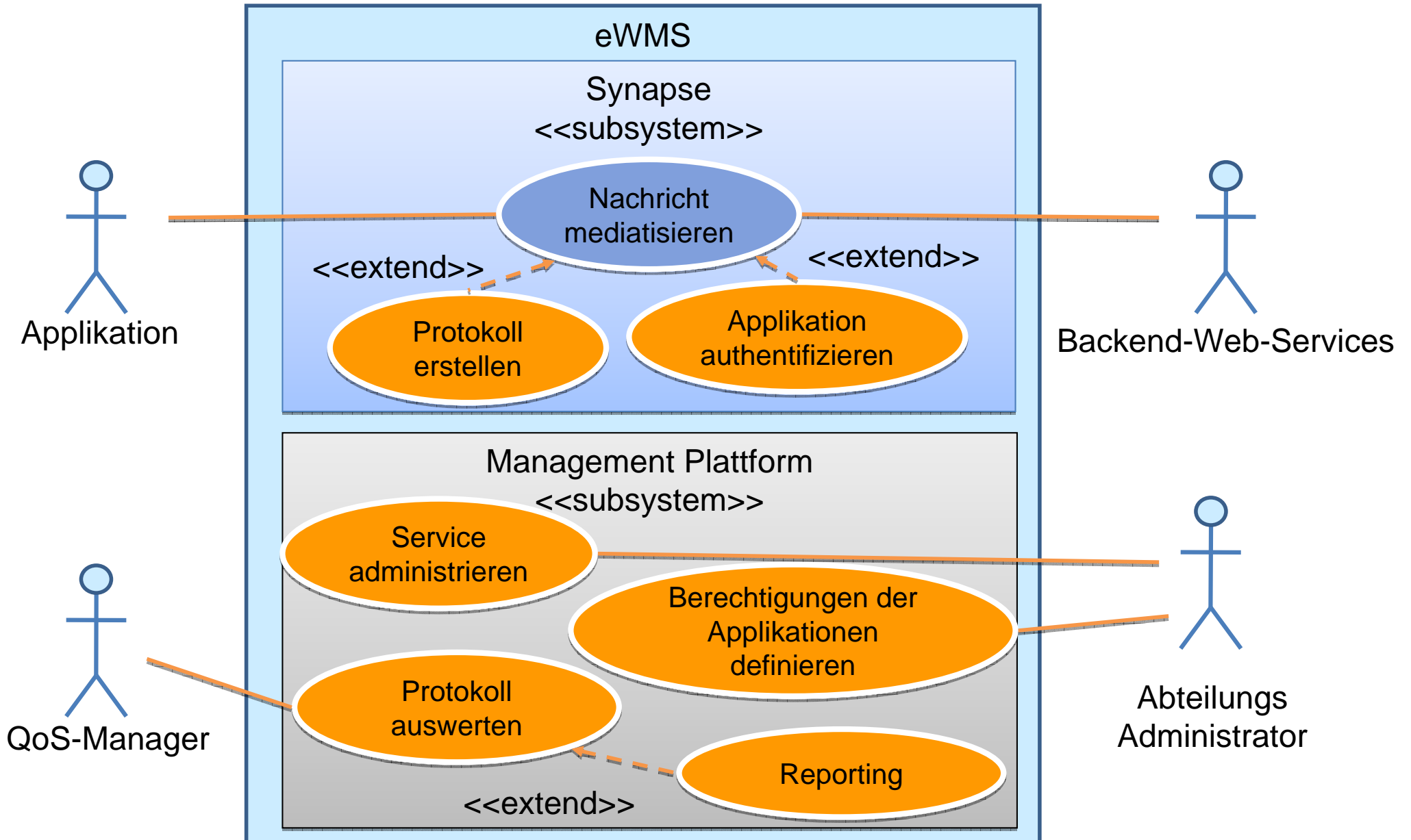


# 6.5 Synapse

## Interner Flow einer Nachricht

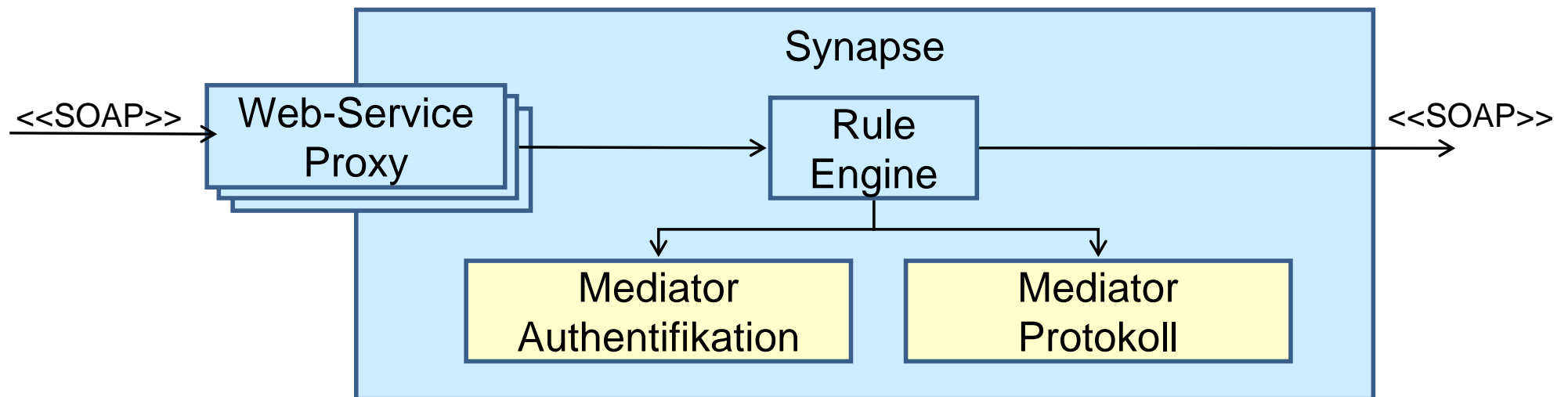


# 6.5 Synapse / Prototyp



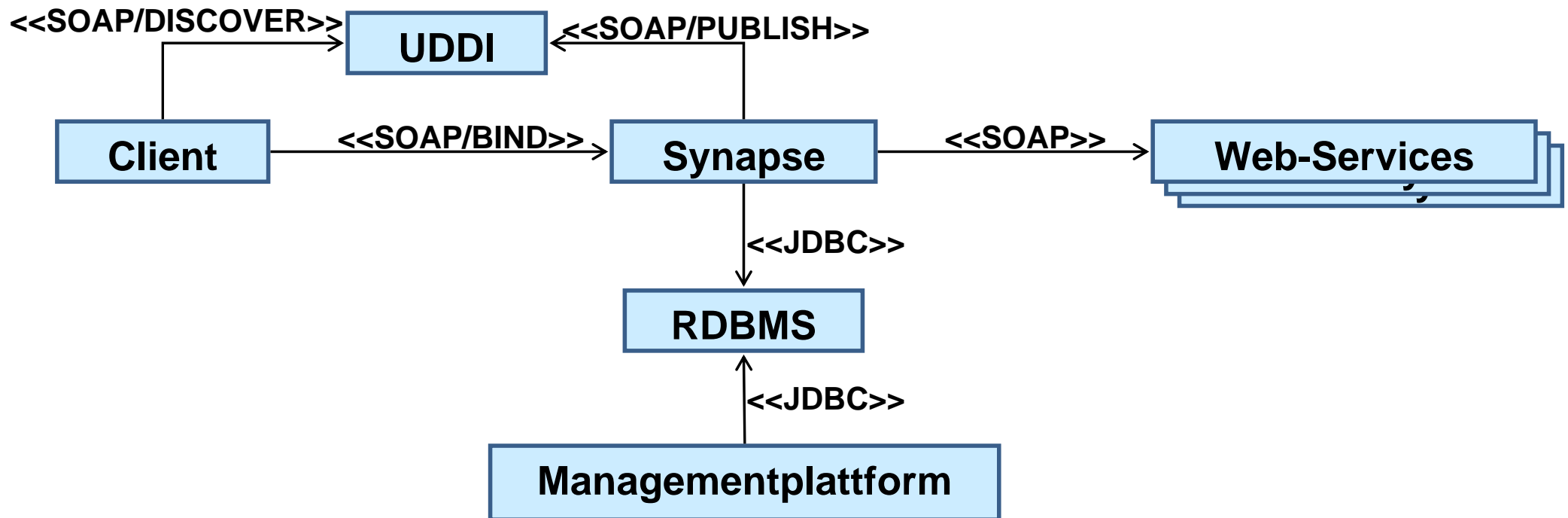
## 6.5 Synapse / Prototyp

### Erweiterung von Synapse durch Mediatoren

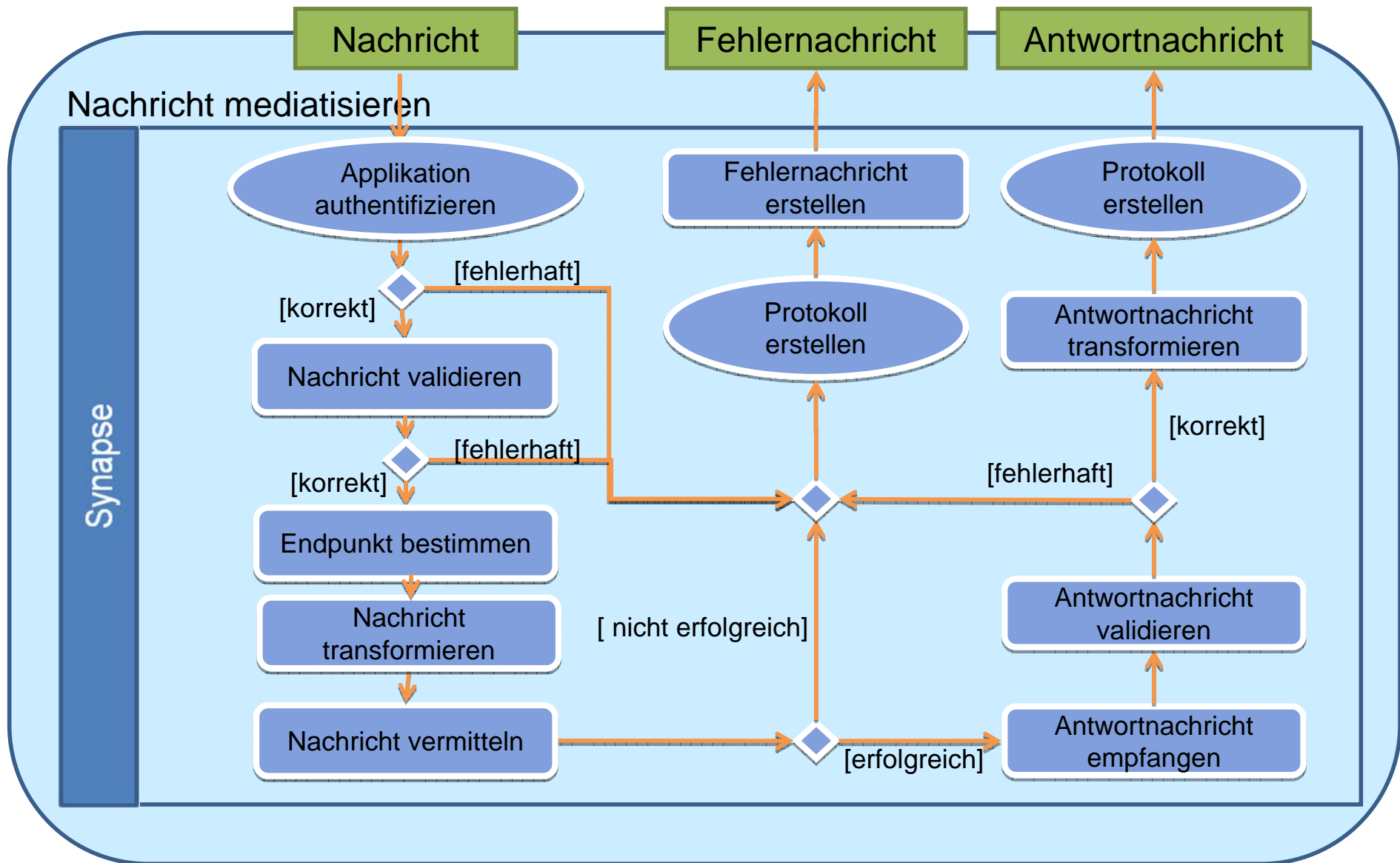


# 6.5 Synapse / Prototyp

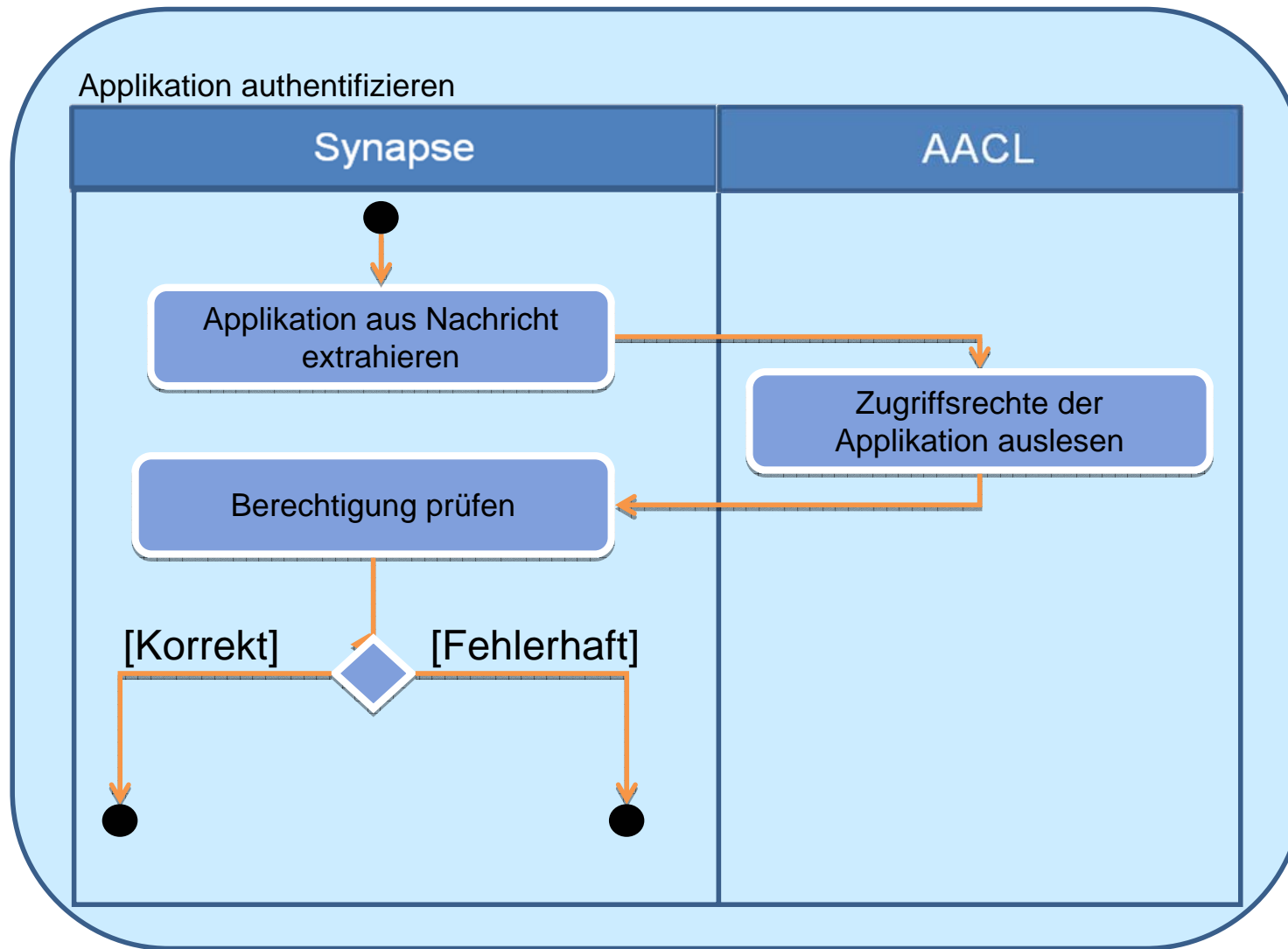
## Systemarchitektur eWMS



# 6.5 Synapse / Prototyp

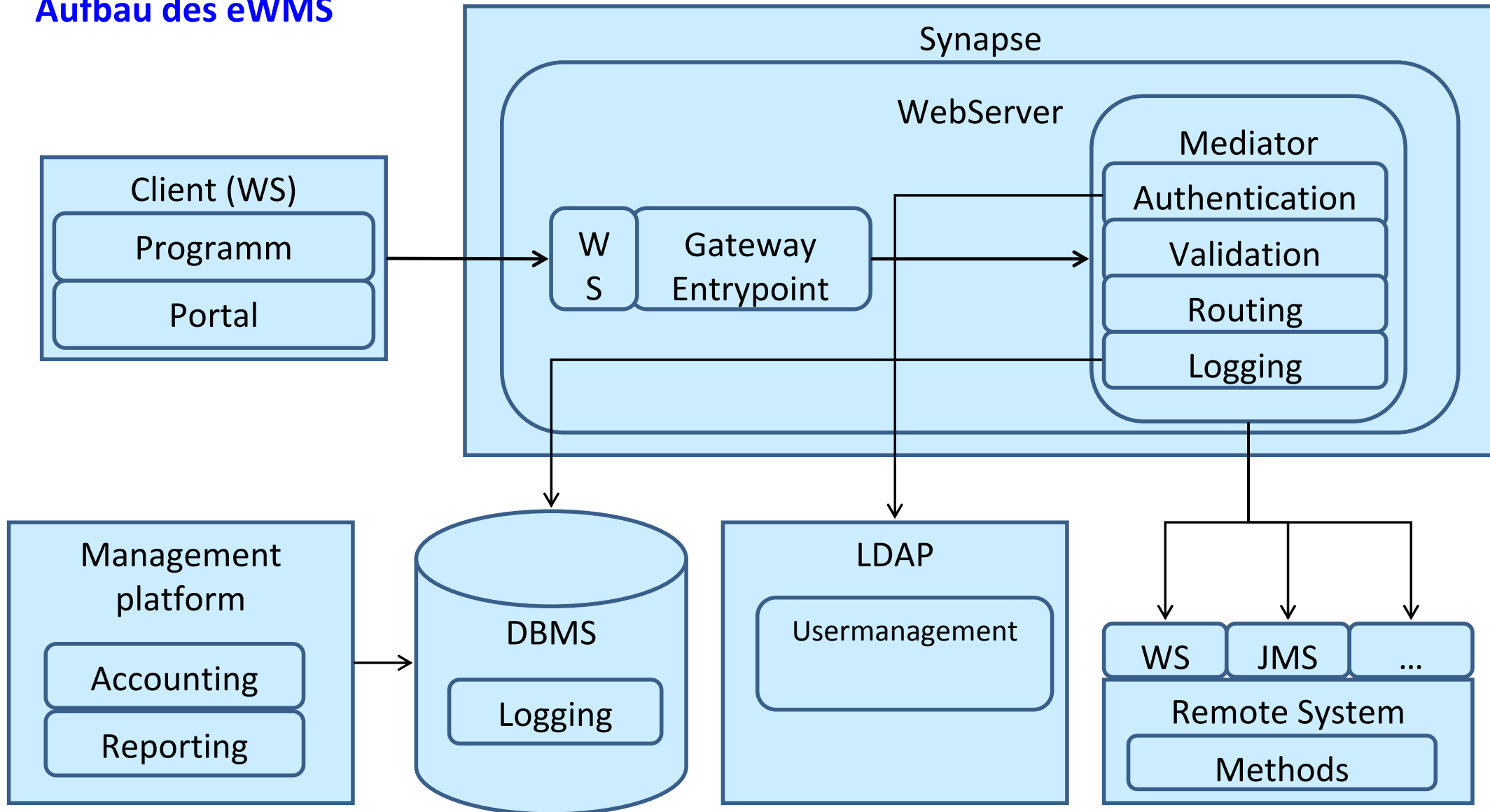


## 6.5 Synapse / Prototyp

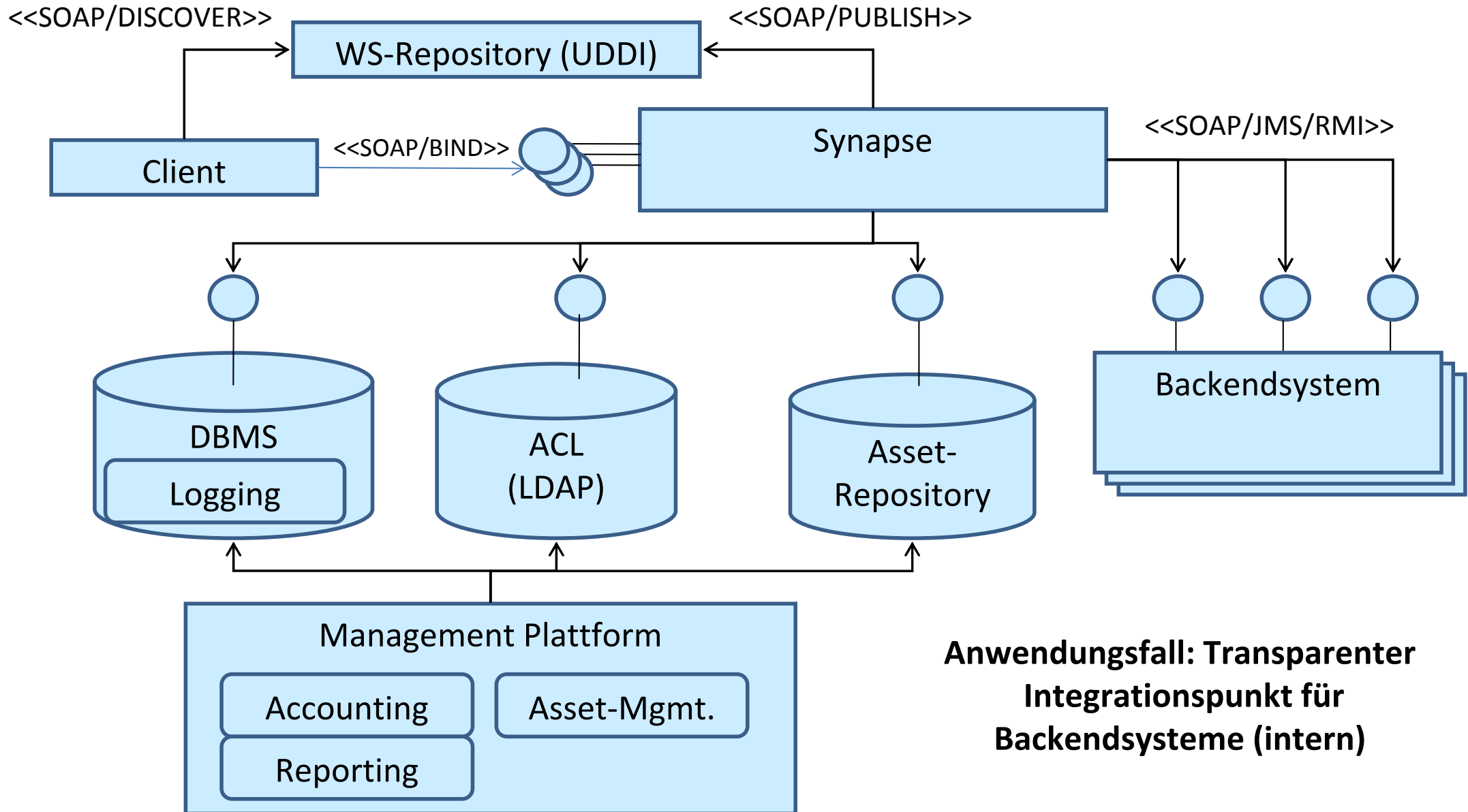


# 6.5 Synapse / Prototyp

## Aufbau des eWMS



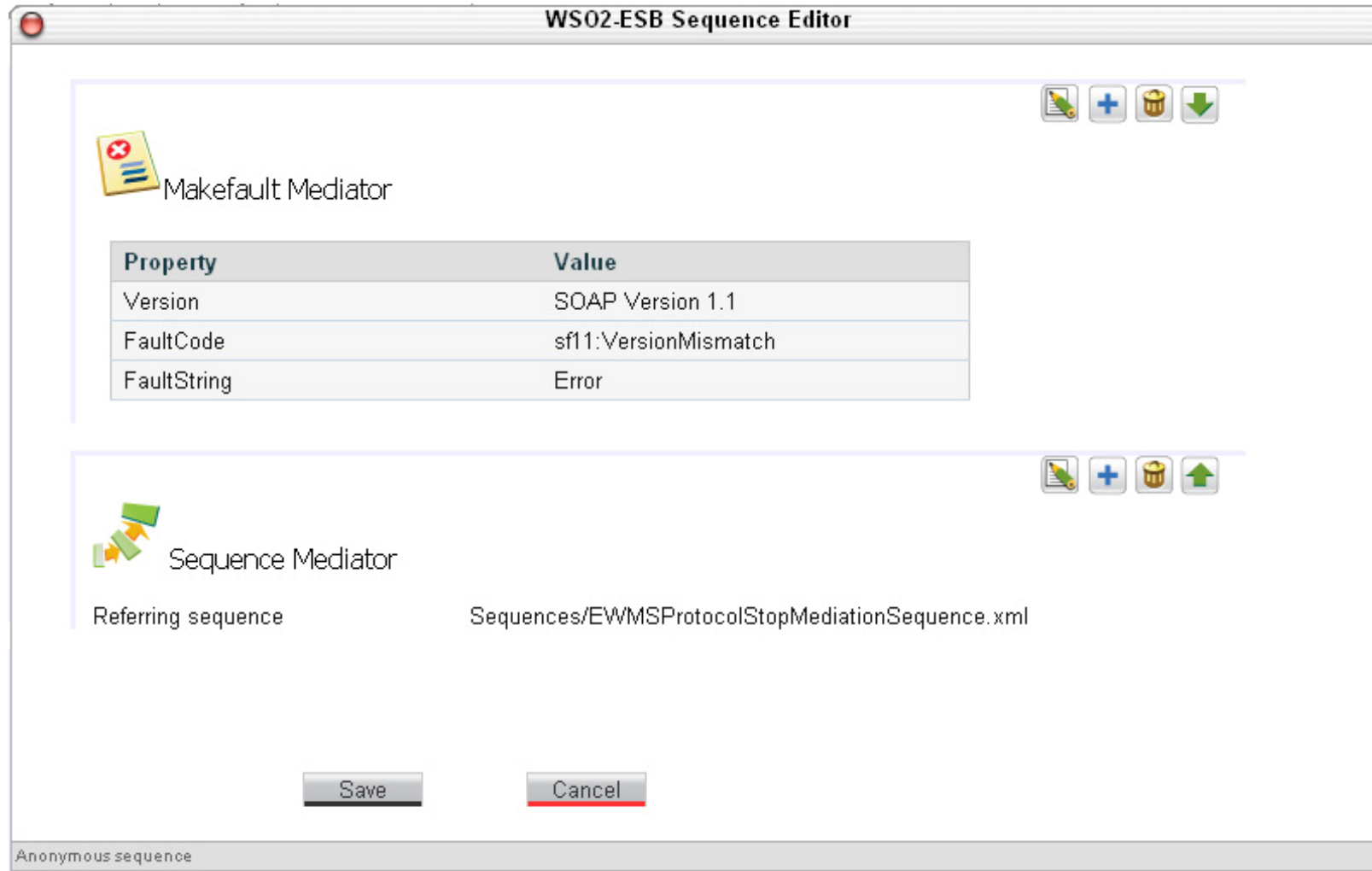
# 6.5 Synapse / Prototyp



**Anwendungsfall: Transparenter Integrationspunkt für Backendsysteme (intern)**

# 6.5 Synapse / Prototyp

## WS02-ESB



# 6.5 Synapse / Prototyp

## WS02-ESB

The screenshot displays the SoapUI 1.7.5 interface. On the left, a project tree shows a 'StockQuoteProxy' with several endpoints, including 'getQuote' which is currently selected. Below the tree, the 'Request Properties' panel shows details for the selected request: Name: T2: VALIDATIONERROR, Description, Message Size: 308, Encoding: UTF-8, Endpoint: http://zaknafein:8..., Bind Address, and Username: Applikation1.

The main workspace is split into two panes. The left pane shows the SOAP request XML:

```
<?xml version='1.0' encoding='UTF-8'>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header/>
  <soap:Body>
    <xsd:getQuote>
      <xsd:request>
        <xsd:symbol>ORACLE</xsd:symbol>
      </xsd:request>
    </xsd:getQuote>
  </soap:Body>
</soap:Envelope>
```

The right pane shows the SOAP response XML, which is a fault:

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode xmlns:sf="http://schemas.xmlsoap.org/soap/envelope/#faultcode">sf:ValidationException</faultcode>
      <faultstring>EWMS: SOAP-Message VALIDATIONERROR</faultstring>
      <detail/>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Below the XML panes, the 'SOAP Request' and 'SOAP Response' tabs are visible. The response tab shows a 'response time: 124ms (353 bytes)'. At the bottom, a log window displays the following messages:

```
Sat Sep 01 13:22:51 CEST 2007:INFO:initialized soapui-settings from [F:\JCoding\SOATools\soapui-1.7.5\bin\soapui-settings.xml]
Sat Sep 01 13:22:52 CEST 2007:INFO:Loading workspace from [C:\Dokumente und Einstellungen\Michael\default-soapui-workspace.xml]
Sat Sep 01 13:22:52 CEST 2007:INFO:Loaded project from [F:\JCoding\WS-Gateway\EclipseWS\ewmsMediators\soapUI\StockQuoteeWMS-soapui-project.xml]
Sat Sep 01 13:23:00 CEST 2007:INFO:Initializing SSL
Sat Sep 01 13:23:02 CEST 2007:INFO:Got response for [StockQuoteProxySOAP12Binding.getQuote:T1: SUCCESS] in 1210ms (855 bytes)
Sat Sep 01 13:23:13 CEST 2007:INFO:Got response for [StockQuoteProxySOAP12Binding.getQuote:T2: VALIDATIONERROR] in 124ms (353 bytes)
```

# 6.5 Synapse / Prototyp

## WS02-ESB

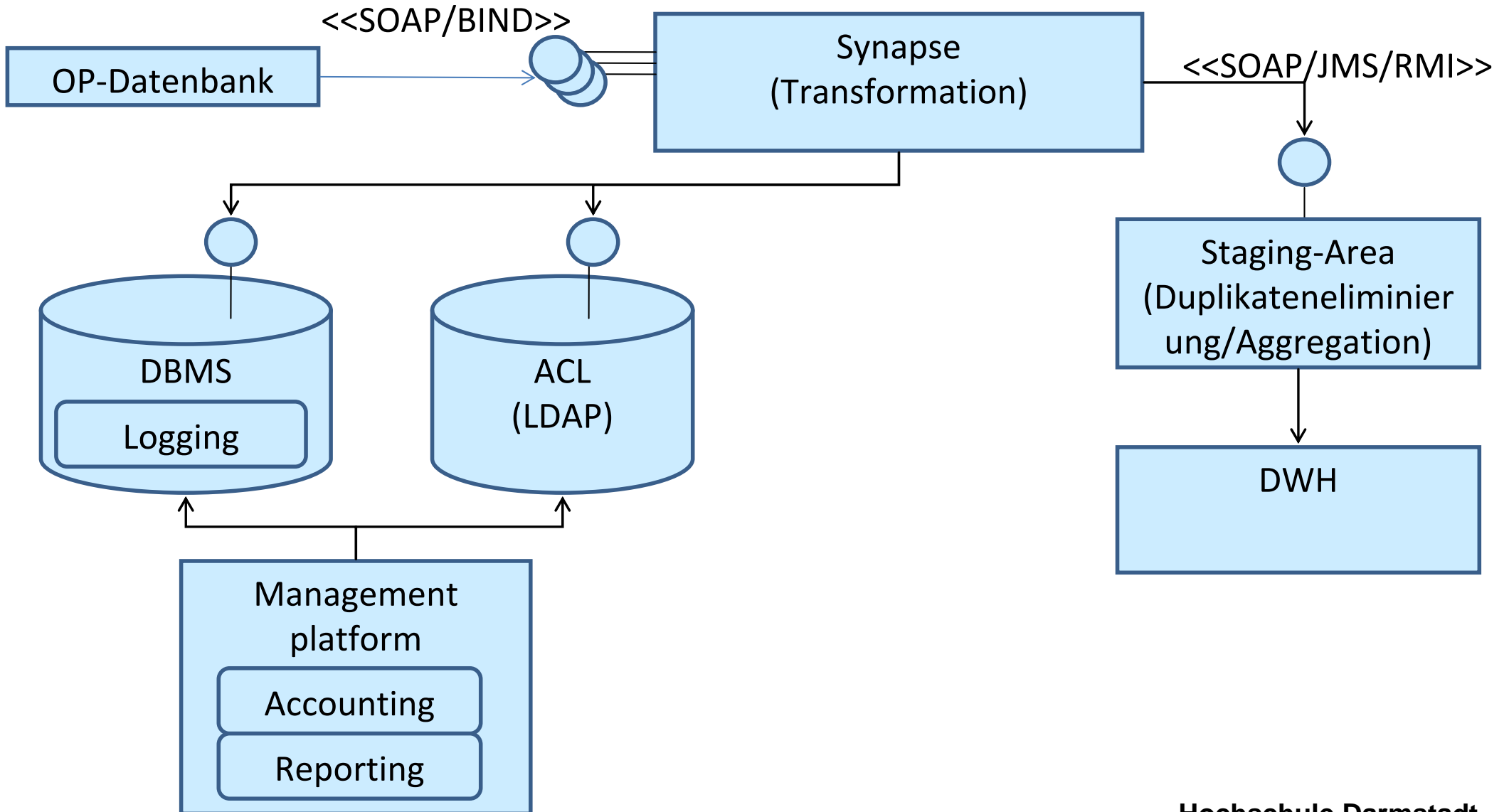
### SimpleStockQuoteService (Lasttest)

Test Step	min	max	avg	last	cnt	tps	bytes	bps	err
getQuote - Service	4	66	5.36	5	796	1862.85	679423	1589...	0
TestCase:	4	66	5.36	5	796	1865.67	679423	1591...	0

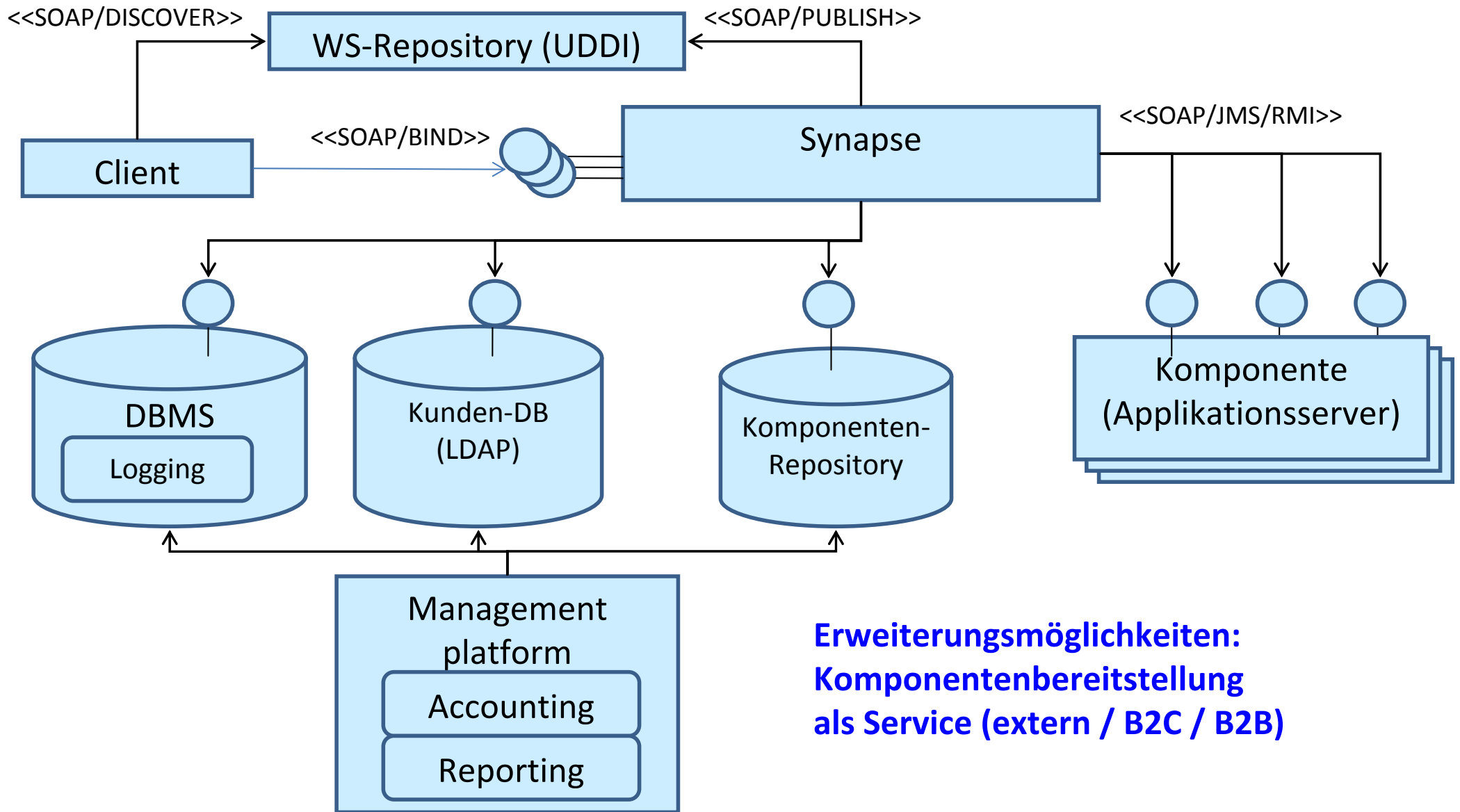
Die durchschnittliche Antwortzeit des SimpleStockQuoteServices liegt bei 5.07 ms (Spalte „avg“)

# 6.5 Synapse / Prototyp

## Erweiterungsmöglichkeiten: Transformationspunkt in ETL-Prozessen



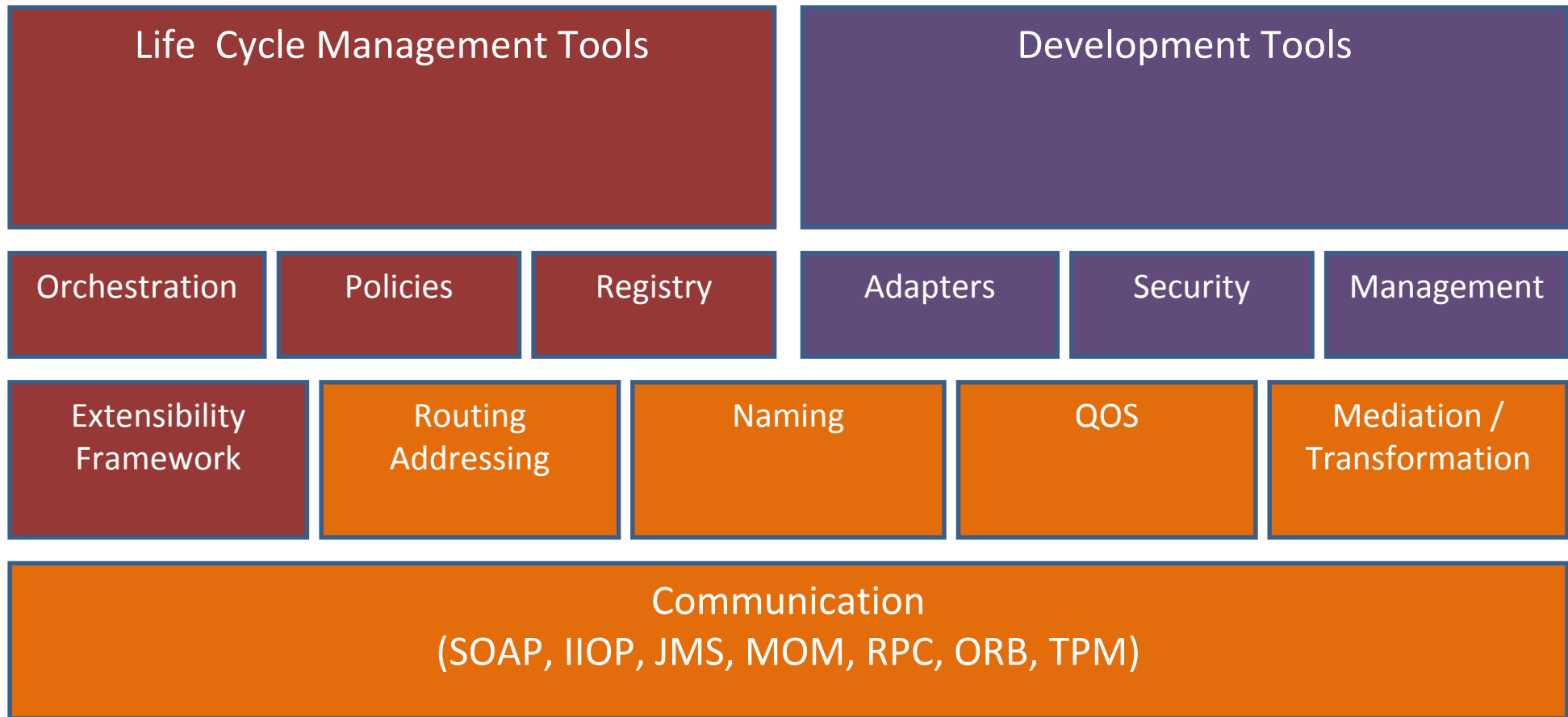
# 6.5 Synapse / Prototyp



**Erweiterungsmöglichkeiten:  
Komponentenbereitstellung  
als Service (extern / B2C / B2B)**

# 6.5 Synapse / Prototyp

## Gesamtarchitektur für eWMS



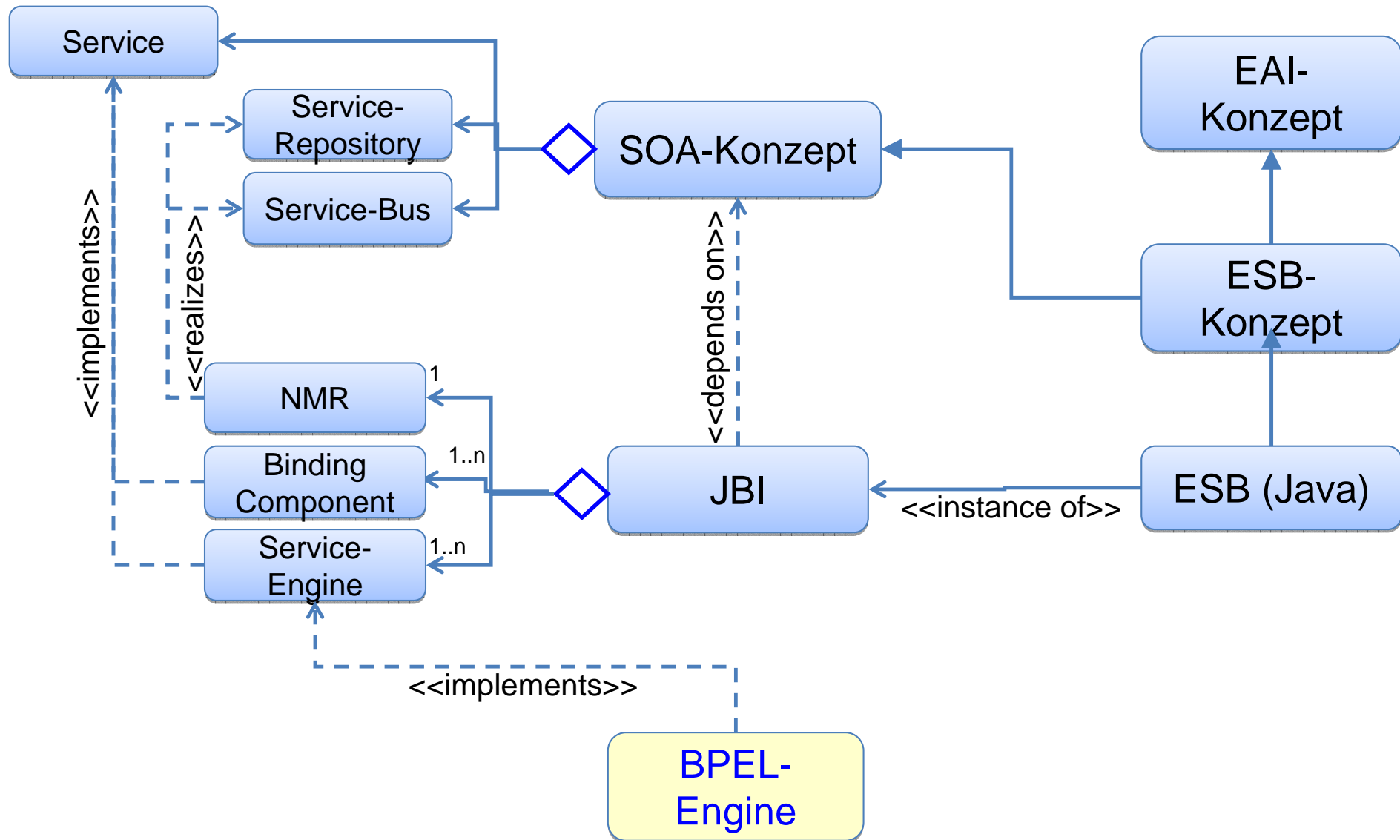
# Vorlesung „ Daten- und Systemintegration“

## **Kapitel 4      SOA und Web Service Mediation Systeme**

- 6.1      Entwicklung eines Webservice mit Oracle 10g (JDeveloper)
- 6.2      Begriffe und Überblick
- 6.3      Überblick WMS (Definition, Funktionalität, Anwendungsgebiete)
- 6.4      Erweitertes WMS
- 6.5      Synapse, prototypische Implementierung
- 6.6      JBI und WMS**

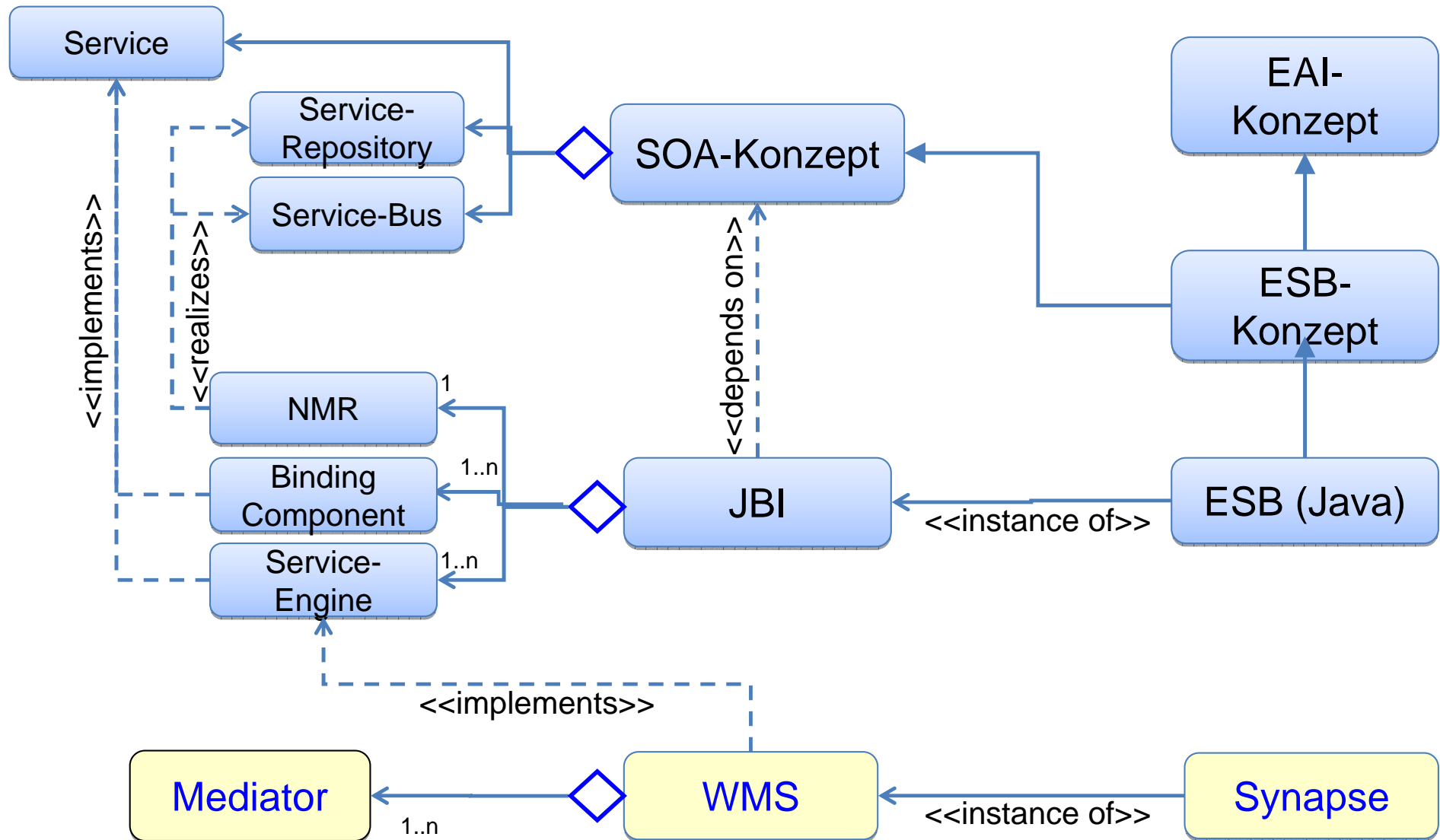
# 6.6 JBI und BPEL

## Abgrenzung: SOA, ESB, JBI, EAI, BPEL-Engine



# 6.6 JBI und WMS

## Abgrenzung: SOA, ESB, JBI, EAI, WMS



## 6.6 JBI

### Überblick JBI

Der [Java Community Process](#) einen Industriestandard unter dem Namen „Java Business Integration“ (JBI) als Java Specification Request (JSR) 208 definiert (Aug/2005, Version 1.0).

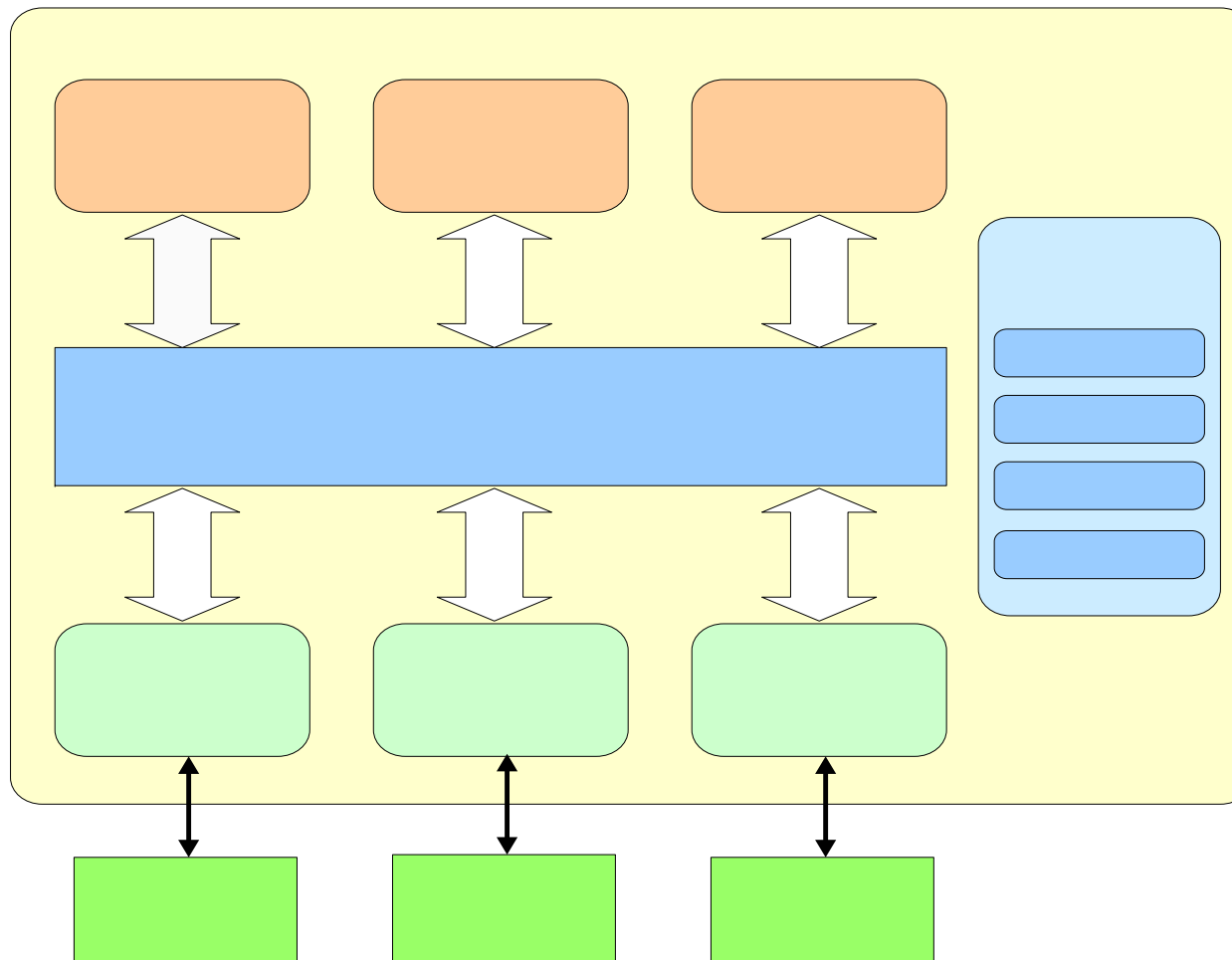
Der Standard wird von vielen Herstellern und Open Source Projekten unterstützt und basiert auf der [Java-Plattform](#).

Java Business Integration definiert ein [Plug-in Framework](#), mit dem verschiedene Komponenten einfach in das JBI-System eingebunden werden können. Dabei wird nicht vorgeschrieben, wie die einzelnen Komponenten intern aufgebaut sein müssen, sondern nur wie die [Schnittstellen zu den Komponenten](#) aussehen sollen.

Quelle: Masterarbeit Hr. Michael Steinbrecht

## 6.6 JBI

### Überblick JBI-Komponenten

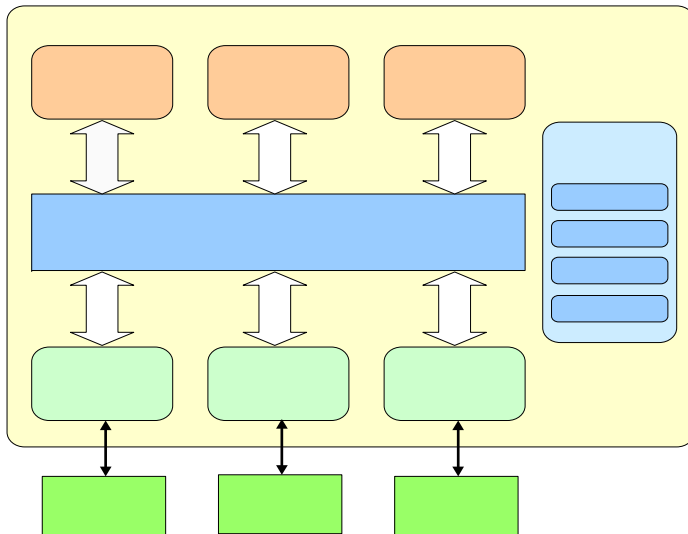


Die eigentlichen **Komponenten** von JBI lassen sich in Normalized Message Router (NMR), Service Engines (SE) und Binding Components (BC) unterscheiden.

Die einzelnen Komponenten sind lose gekoppelt und kommunizieren nicht direkt miteinander, sondern über das JBI-System. Nur das JBI-System kennt alle Komponenten und tritt als **Service-Mediator** auf.

## 6.6 JBI

### Überblick JBI-Komponenten – Service Engines

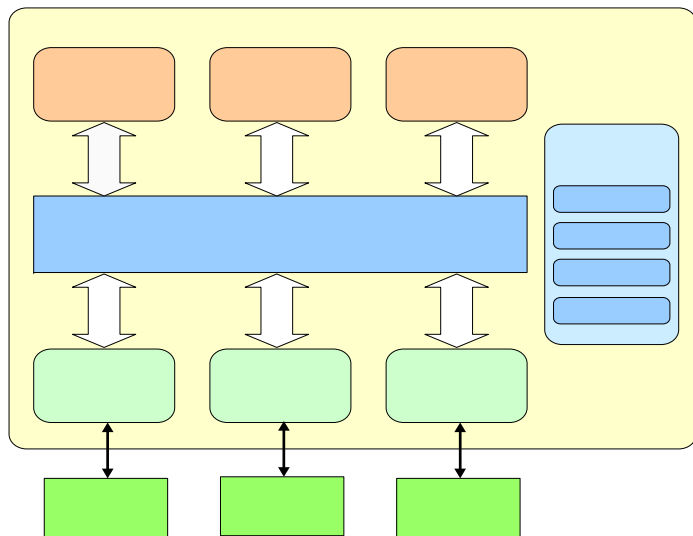


Service Engines stellen in einem JBI-System die **Business-Logik** zur Verfügung. Sie bieten Funktionalitäten über Services an und können ihrerseits Services von anderen Engines konsumieren.

Beispielsweise können SEs als **BPEL-Engines** zur Ausführung von Prozessen, als **XSLT-Engines** zur Transformation von Daten oder als **JavaEE-Engines** zur Ausführung von Enterprise Java Beans (EJBs) eingesetzt werden.

## 6.6 JBI

### Überblick JBI-Komponenten – Binding Components (vgl. SOA-Adaper)



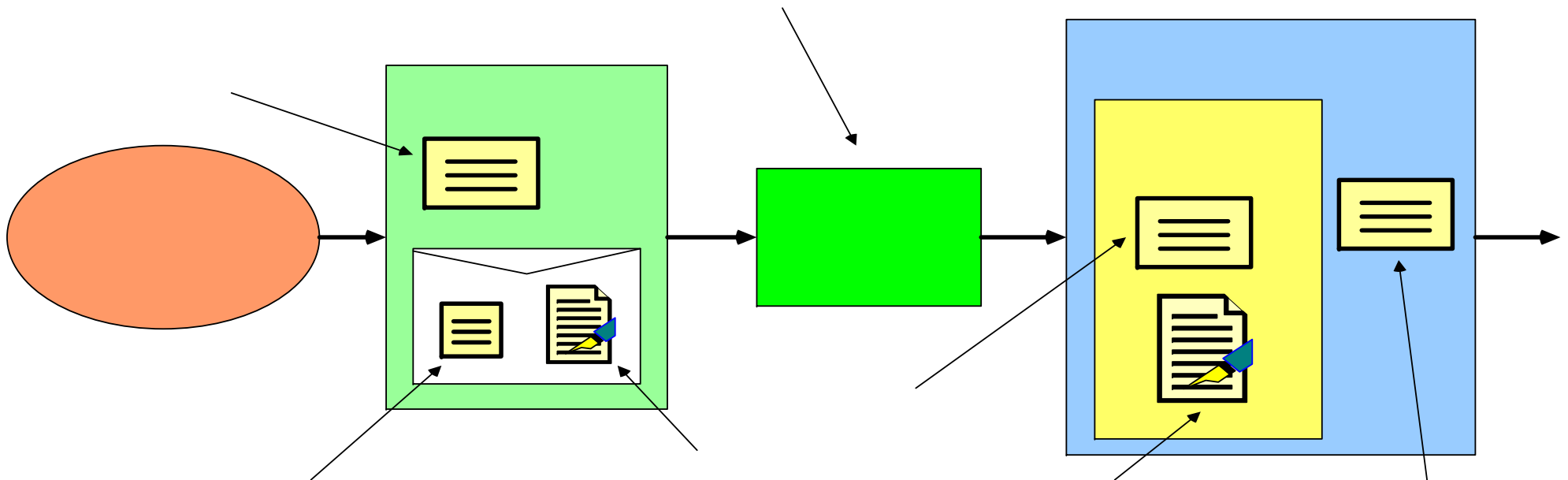
Binding Components dienen der Anbindung von externen **Service Consumern** und **Service Providern** an das JBI-System und stellen die Kommunikation zu Systemen außerhalb der JBI-Umgebung dar. Sie konvertieren spezielle Datenformate in das **MessageExchange Format**, als Standard-Format in der JBI-Welt und übernehmen damit die Protokollübersetzung zwischen JBI-Komponenten und externen Service Providern bzw. Service Consumern.

Beispiele für BCs können **HTTP/SOAP-Bindings** für die Anbindung von SOAP-Clients, **Java Message Service-Bindings** für die Anbindung von JMS-Clients oder **JDBC-Bindings** zur Integration von Datenbanken sein.

## 6.6 JBI

### Message Exchange Patterns

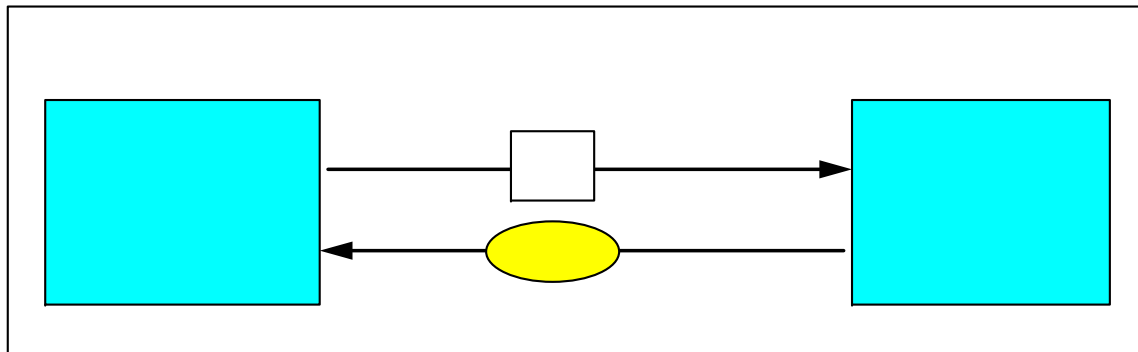
Nachrichtenkonvertierung über die SOAP/HTTP BC



## 6.6 JBI

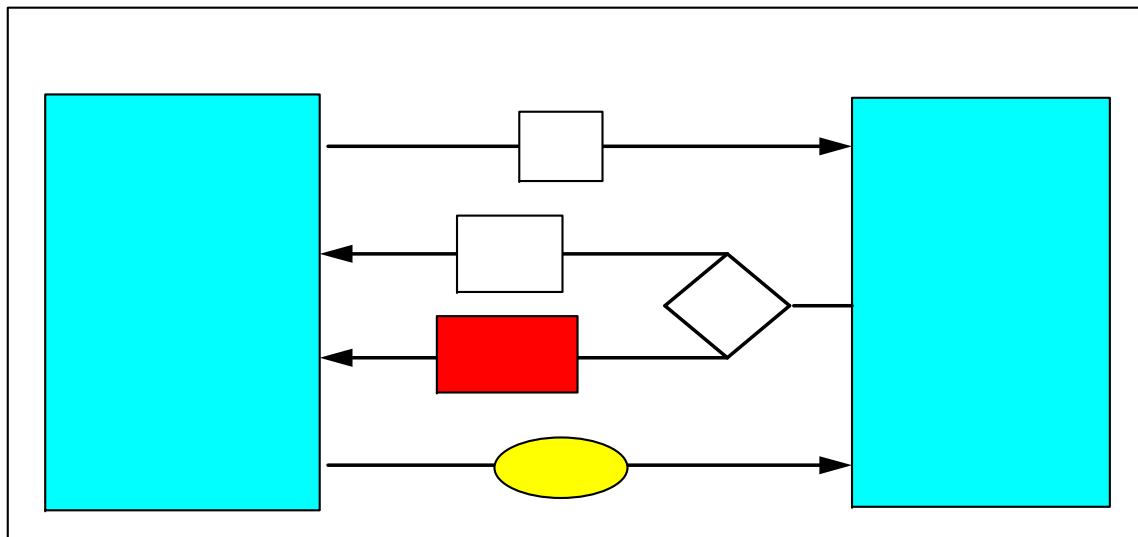
### JBI: Message Exchange Patterns

Das Messaging-Konzept von JBI basiert auf den WSDL 2.0 [Message Exchange Patterns](#) (MEPs). Nachfolgend sind zwei Patterns dargestellt:



#### Asynchroner Nachrichtenaustausch

Consumer als Service-Nutzer sendet eine IN-Nachricht als Request und erwartet danach keine Antwortnachricht vom Provider. Der Provider beendet nur mit dem „*DONE*“-Status die Kommunikation zwischen den beiden Parteien



#### Synchroner Nachrichtenaustausch

Der Consumer sendet eine Anfrage als IN-Nachricht und bekommt vom Provider entweder eine OUT-Nachricht als Antwort im Erfolgsfall oder eine Fehlermeldung als FAULT-Nachricht, wenn bei der Bearbeitung der Anfrage ein Fehler aufgetreten ist. Der Service-Nutzer beendet die Kommunikation in jedem Fall wiederum mit einem „*DONE*“-Status.

## 6.6 JBI und ESB

### JBI: SUN's OpenESB – Komponenten für Service Engines

<b>BPEL Service Engine</b>	<b>Komponente zur Ausführung von BPEL-Prozessen</b>
<b>Java EE Service Engine</b>	<b>Stellt die gesamte Java Enterprise Edition als JBI-Komponente zur Verfügung, so dass EJBs mit anderen JBI-Komponenten kommunizieren können</b>
<b>XSLT Service Engine</b>	<b>Ermöglicht XSLT-Transformationen</b>
<b>POJO Service Engine</b>	<b>Business Logik, die in Java geschrieben ist, kann als JBI-Komponente zur Verfügung gestellt werden</b>
<b>Scripting Service Engine</b>	<b>Ermöglicht die Benutzung von Scriptsprachen innerhalb von JBI-Komponenten</b>
<b>Encoding Service Engine</b>	<b>Komponente zur Transformation von Nachrichten</b>
<b>IEP Service Engine</b>	<b>Stellt Funktionalitäten für intelligente Ereignisverarbeitung zur Verfügung</b>

## 6.6 JBI und ESB

### **JBI: SUN's OpenESB – Komponenten für Binding Components**

<b>HTTP Binding Component</b>	<b>Komponente für den Versand und Empfang von Nachrichten über das HTTP-Protokoll</b>
<b>JMS Binding Component</b>	<b>Ermöglicht die Kommunikation über Java Messaging Service</b>
<b>eMail Binding Component</b>	<b>Binding Komponente zum Senden und Empfangen von Emails</b>
<b>File Binding Component</b>	<b>Stellt Funktionalitäten für den Zugriff auf Dateien im Dateisystem zur Verfügung</b>
<b>Database Binding Component</b>	<b>Bietet Zugriff auf Datenbanken über das JDBC-Protokoll</b>
<b>FTP Binding Component</b>	<b>Stellt Funktionalitäten für den Zugriff auf Dateien über FTP zur Verfügung</b>
<b>LDAP Binding Component</b>	<b>Komponente für den Zugriff auf Informationen über das LDAP-Protokoll</b>
<b>CORBA Binding Component</b>	<b>Bildet einen CORBA Service als JBI-Endpunkt ab</b>
<b>SAP Binding Component</b>	<b>Komponente für den Zugriff auf SAP-Systeme</b>

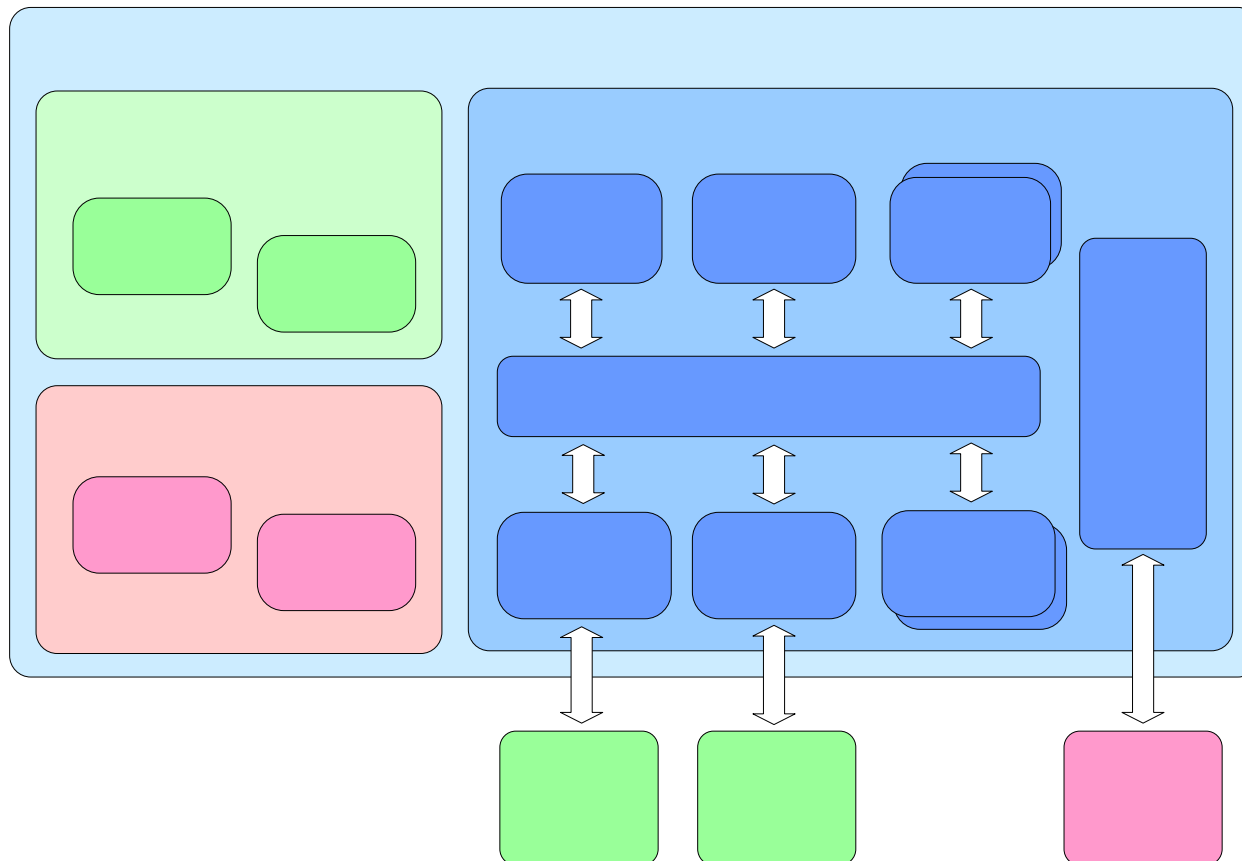
## 6.6 JBI und ESB

### JBI: Ein Standard für einen ESB?

Basisfunktionalität	JBI
<b>Service-Bus auf Basis eines Standards (z.B. JMS oder andere MOM)</b>	<b>Normalized Message Router</b>
<b>Mediation (Transformation, implizites- &amp; explizites Routing)</b>	<b>Transformation durch BCs. Explizites Routing durch WSDL-Service Names. Implizites Routing durch das Anschließen einer Mediationskomponente an den NMR.</b>
<b>Einhaltung von Industriestandards für EAI</b>	<b>XML, WSDL 2.0. Andere Standards über Plug-In Konzept integrierbar.</b>
<b>Unterstützung von Web-Services</b>	<b>Über BCs</b>
<b>Repository für interne Services</b>	<b>Service-Provider werden intern über WSDL-Service-Names adressiert.</b>
<b>Administration und Monitoring von internen Services</b>	<b>Management-Komponenten</b>

## 6.6 JBI und ESB

### JBI: SUN's OpenESB - Architektur

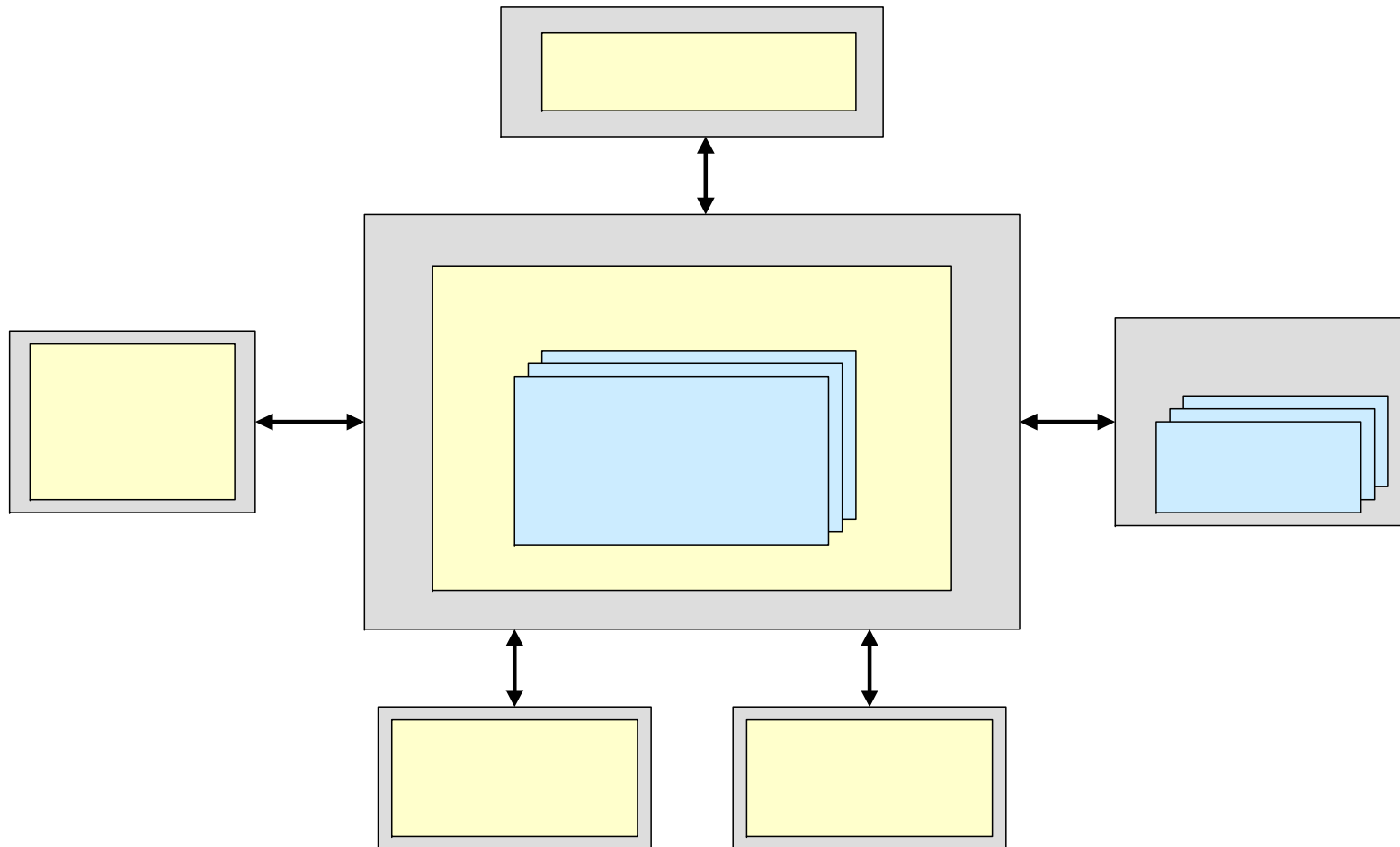


Der blaue Bereich stellt die Runtime- Komponenten von **OpenESB** dar und entspricht der Architektur eines JBI-Systems. Der JBI-Container beinhaltet dabei den **Runtime Kern** in Form des **Normalized Message Router**. Dieser stellt Basisdienste zur Verfügung und ermöglicht eine Kommunikation in dem JBI-System. Weiterhin enthält der JBI-Container eine **System Management Komponente**, alle installierten **Service Engines** und **Bindung Components**.

Neben dem **JBI-Container** enthält der Application Server noch einen **EJB-** und einen **Web-Container**. In den EJB-Container können Enterprise Java Beans (EJB) Applications deployed werden und der Web-Container kann zur Verwaltung von Web-Anwendungen genutzt werden.

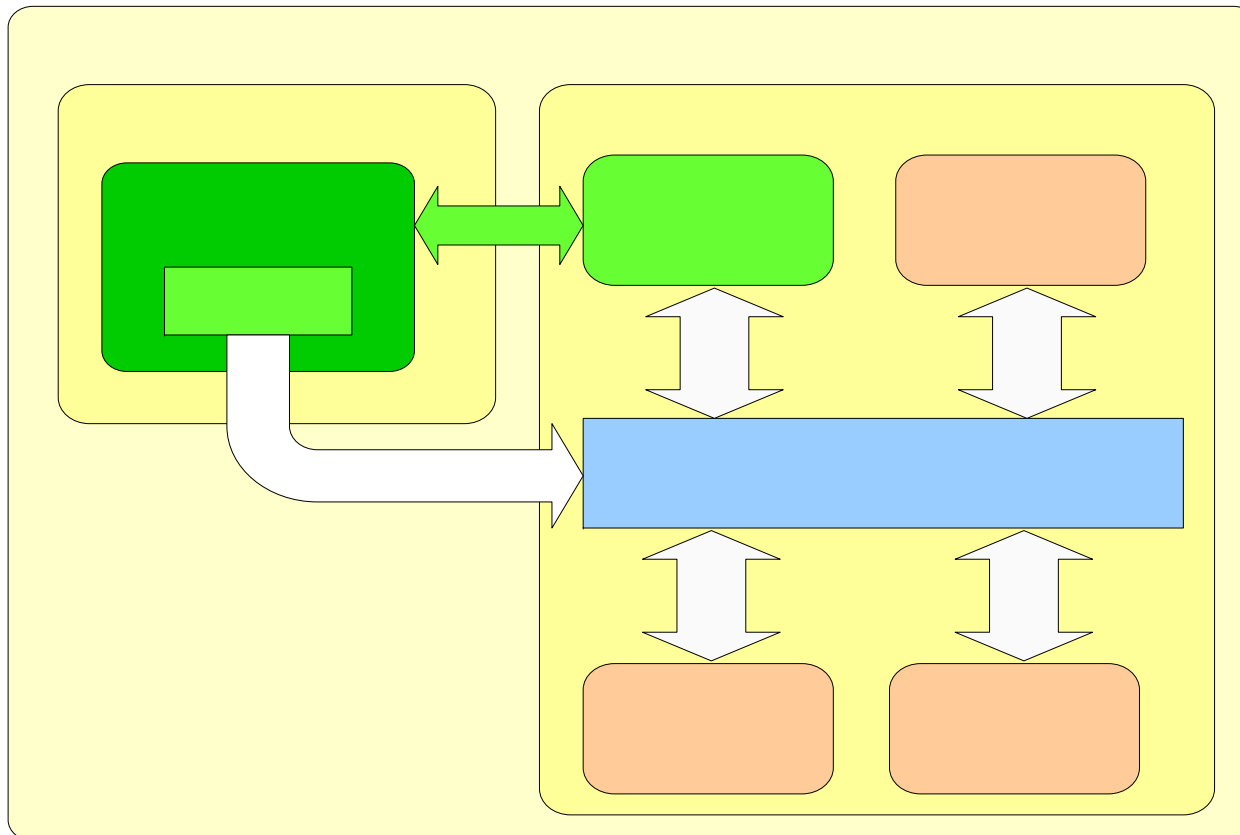
## 6.6 JBI und WfMOpen (XPDL-Engine)

### Workflow Referenzmodell der WfMC



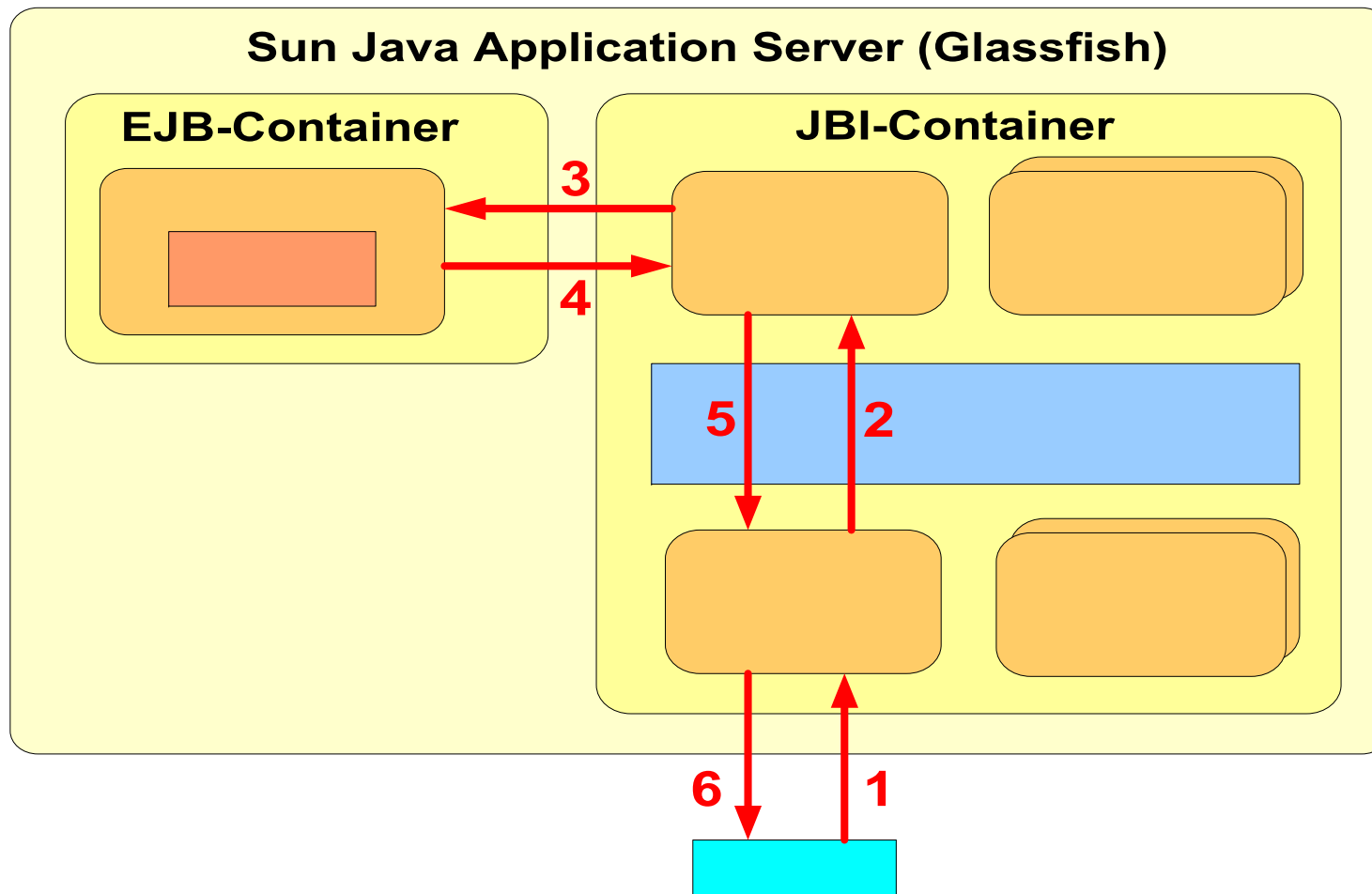
## 6.6 JBI und WfMOpen (XPDL-Engine)

### Integration der XPDL-Engine WfMOpen



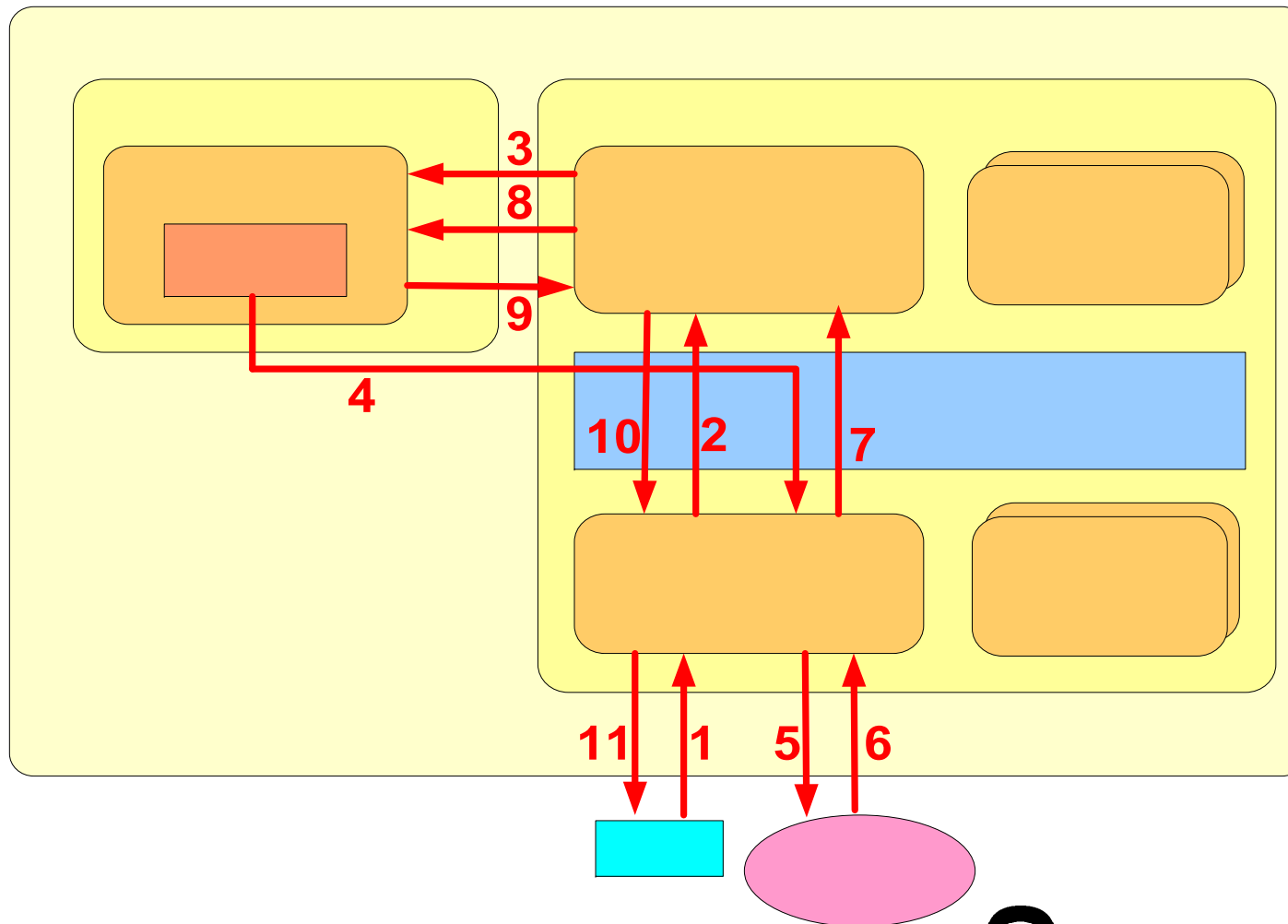
## 6.6 JBI und WfMOpen (XPDL-Engine)

### Integration der XPDL-Engine WfMOpen



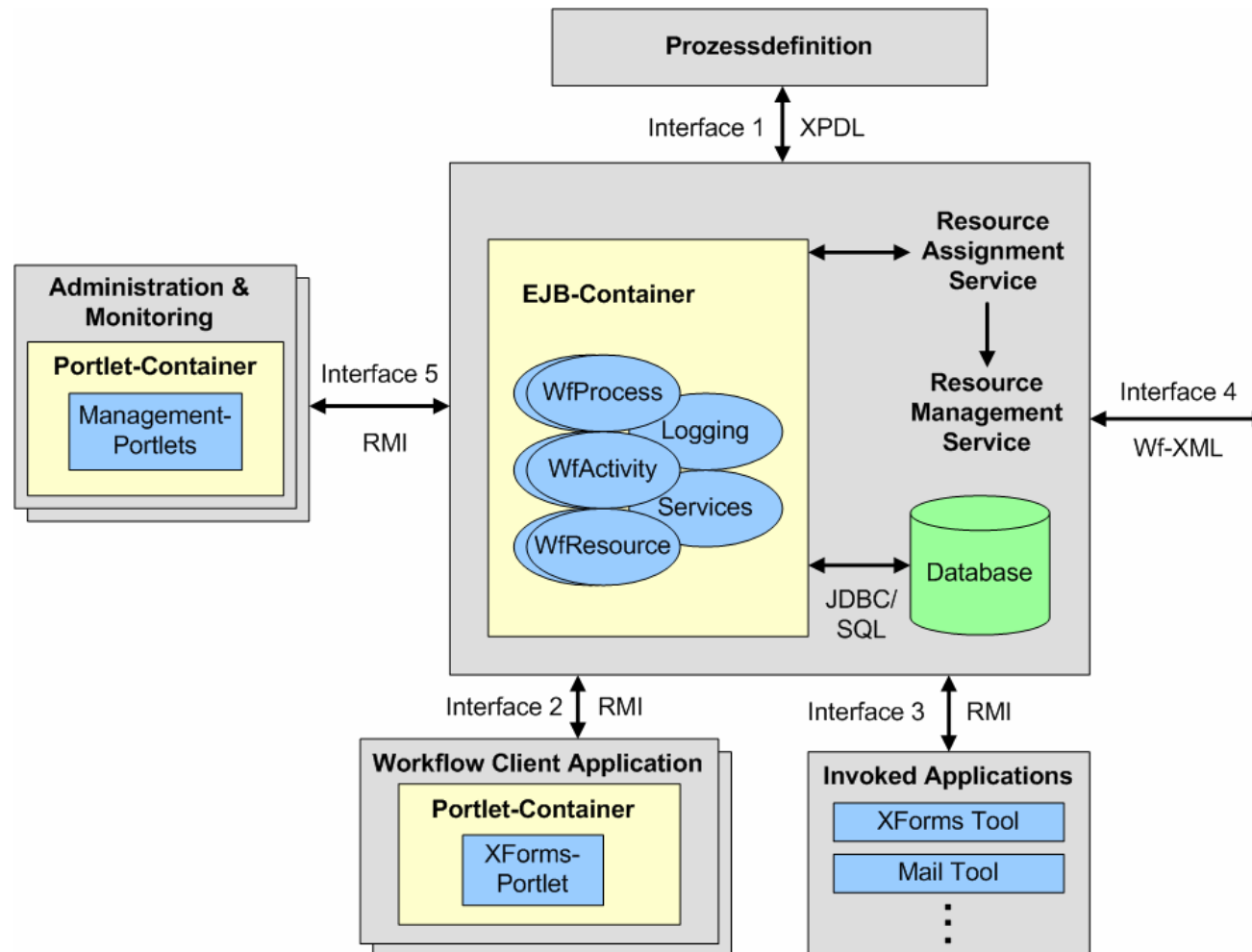
## 6.6 JBI und WfMOpen (XPDL-Engine)

### Integration der XPDL-Engine WfMOpen



# 6.6 JBI und WfMOpen (XPDL-Engine)

## Struktur von WfMOpen



# 6.6 JBI und ESB

## Integration von WMS in JBI

