Angewandte Softwareentwicklung
Serviceorientierte Architekturen

WS 2014/2015

Markus Berg
Hochschule Wismar
Fakultät für Ingenieurwissenschaften
Bereich Elektrotechnik und Informatik

markus.berg@hs-wismar.de
http://mmberg.net

Teil I: Grundlagen

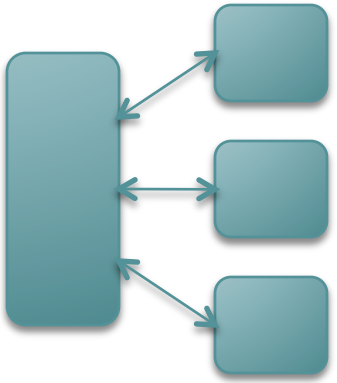
Dienste, Orchestrierung, Geschäftsprozesse

Motivation

- Warum Dienste?
  - Dienste im Sinne der Wiederverwendung / Modularisierung
- Status Quo
  - Selbstständige Dienste, die keine Beziehung zueinander haben
- Ziel
  - Dienst, der andere Dienste benutzt

Dienstekomposition

- Web Service Komposition
  - Orchestrierung
    - Mehrere Dienste kombinieren (zu einer Komposition) innerhalb eines steuernden Dienstes
      - Dienste wissen nicht wie sie verwendet / kombiniert werden (jeder Dienst arbeitet autark)
  - Im Vergleich: Choreographie
    - Mehrere Dienste arbeitet zusammen ohne zentralen steuernden Dienst
    - Die Dienste müssen voneinander wissen



„service orchestration is the coordination and arrangement of multiple services exposed as a single aggregate service“ *[Mulesoft]*

Service-orientierte Architekturen

- Gartner (1996)
- Architekturmuster für verteilte Systeme, keine Technologie
- Nicht zwangsläufig Web Services bzw. SOAP
- Ziele:
  - Wiederverwendbarkeit
  - Modularisierung
  - Kapselung
    - Zugriff auf Datenquellen
    - Schnittstellen zu Drittsystemen
  - Systemunabhängigkeit (Programmiersprache, Plattform, Betriebssystem)
- Kombination mehrere „kleiner“ Dienste zu neuen „Mehrwertdiensten“

> *„The policies, practices, frameworks that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer. Services can be invoked, published and discovered, and are abstracted away from the implementation using a single, standards-based form of interface. (CBDI)“* - *http://msdn.microsoft.com/en-us/library/aa480021.aspx*

Service Orientation

- Vgl. Object Orientation
- Statt Objekten sind Dienste die zentralen Bausteine
  - Verdecken Inneres/Implementierung (Kapselung)
  - Haben definierte Schnittstellen
  - Verbinden Informationen und Verhalten

Dienst (Service)

- Autark
  - Alleine lauffähig
  - Geringe Abhängigkeiten
- Repräsentiert einen fachlichen Anwendungsfall
  - oft ein kleines Stück Software, das genau ein Problem löst
- Kann aus weiteren Diensten bestehen
  - Durch Komposition können komplexere Anwendungsfälle gelöst werden (Services höherer Ebene)
- Lose Kopplung
  - Definierte Schnittstelle
  - Interaktion über Nachrichten
- Blackbox

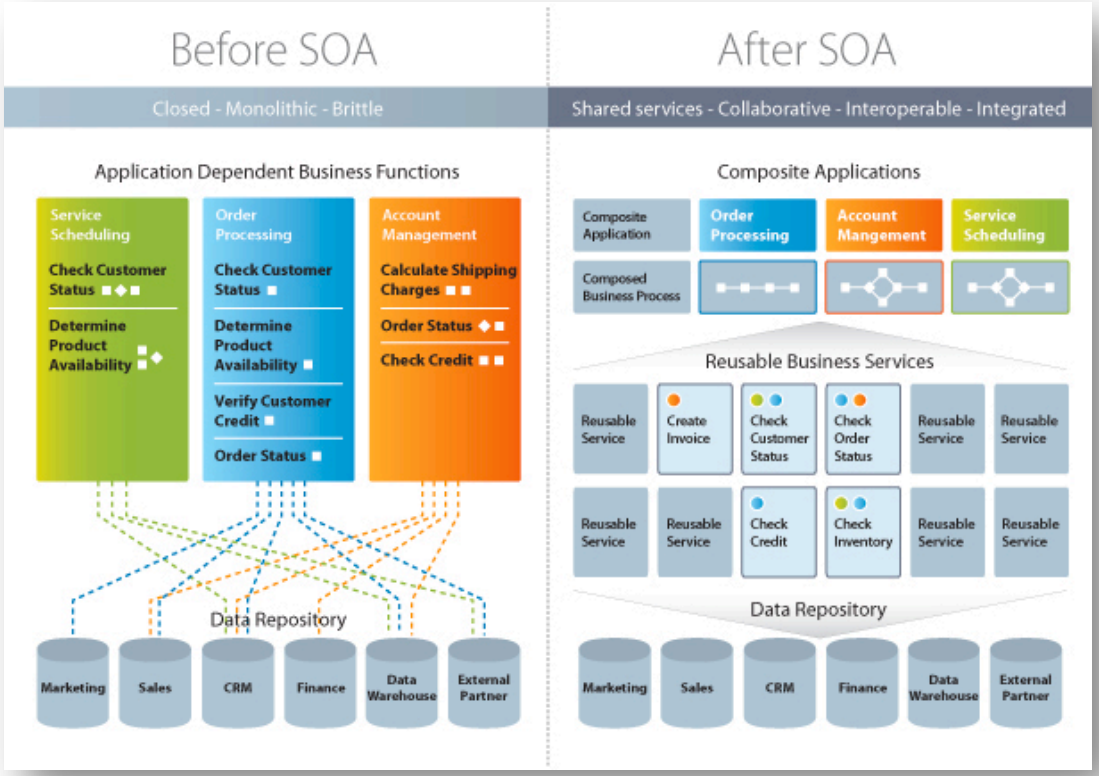
Granularität von Diensten

- bzw. „Schneiden“ von Diensten
- Zu groß: geringe Wiederverwendbarkeit, da der Dienst nur das eine Problem lösen kann (zu viele nicht allgemeingültige Bestandteile), d.h. zu geringe Allgemeingültigkeit
  - z.B. Dienst der die Bezahldaten entgegennimmt und anschließend eine Reise bei einem bestimmten Anbieter bucht
- Zu klein: hohe Wiederverwendbarkeit, viele Schnittstellen und Aufrufe, Overhead, zu wenig Mehrwert, „lohnt sich nicht dafür einen Service aufzurufen“
  - z.B. Dienst der prüft, ob ein Flag gesetzt ist

Orchestrierung von Diensten

- Da die verwendeten Dienste nicht miteinander kommunizieren, muss es einen führenden Steuerprozess geben, der die Business Logik abbildet und z.B. die Ausführungsreihenfolge bestimmt
- WSDL bestimmt nur Schnittstelle, jedoch nicht Komposition von Diensten

Architekturvergleich



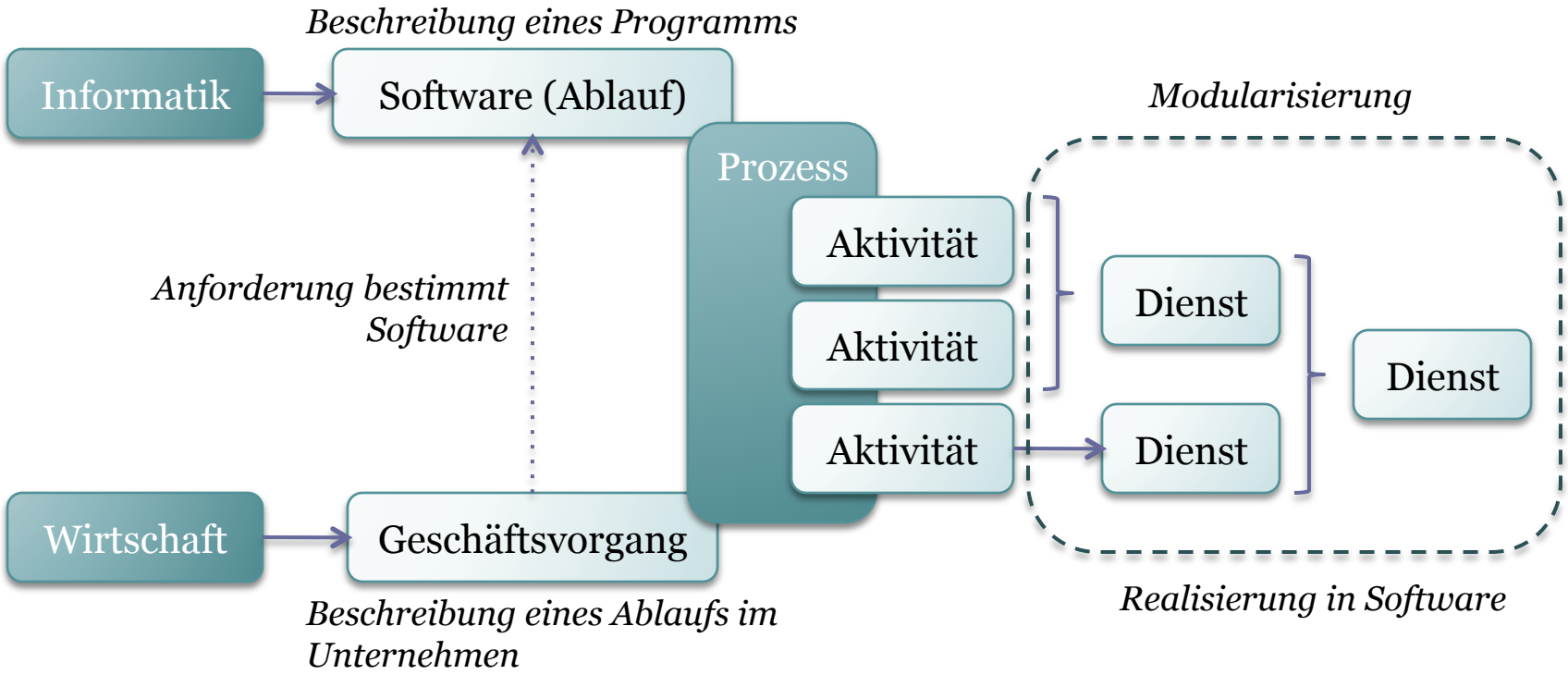
http://www.tridens.si/expertise/soa/

Architekturvergleich (II)

- Architektur beschreibt Komponenten und deren Interaktion aus verschiedenen Blickwinkeln
- z.B.
  - Model View Controller
    - Daten, Ausgabe/Eingabe, Verarbeitung
  - Schichten
    - Z.B. OSI, jede Schicht kann nur auf Funktionalitäten der niederen Schichten zugreifen
  - Blackboard
    - Nachrichten werden in einen zentralen Speicher geschrieben
    - Interessenten werden darüber informiert und können Nachricht abholen
  - SOA
  - Client/Server
  - Peer-to-Peer
  - ...

Technologie vs. Business

- Näherungsmöglichkeiten aus zwei Richtungen



Geschäftsprozesse

- Miteinander verknüpfte Aktivitäten zur Erreichung eines fachlichen/betrieblichen Ziels
  - D.h. unternehmensinterne und –externe Abläufe
  - Kontext: BWL/Wirtschaftsinformatik
  - Englisch: Business Process
- Erzeugt über mehrere Schritte aus einem Eingangswert einen Ausgangswert, der einen (i.d.R. messbaren) Mehrwert darstellt (d.h. er ist wertschöpfend) und ein Geschäftsziel realisiert
  - Unterteilbar (Teilprozesse)
- Prozesse werden optimiert (Ressourcenbedarf, Zeit, ….)

> *„Ein Geschäftsprozess besteht aus einer Folge von Einzelaktivitäten, mit denen festgelegte Geschäftsziele erreicht werden sollen.“* [BSI]

Geschäftsprozesse

- z.B.
  - Toner bestellen
    - Druckermodell wählen
    - Anzahl der benötigten Toner angeben
    - Preise vergleichen
    - Bestellung auslösen
    - Bezahlen
  - Student immatrikulieren
    - Matrikelnummer generieren
    - Stammdaten eingeben
    - Studiengang auswählen
    - Studentenausweis drucken
    - Accounts anlegen (z.B. E-Mail)
  - …

Kreditantrag bei der Bank

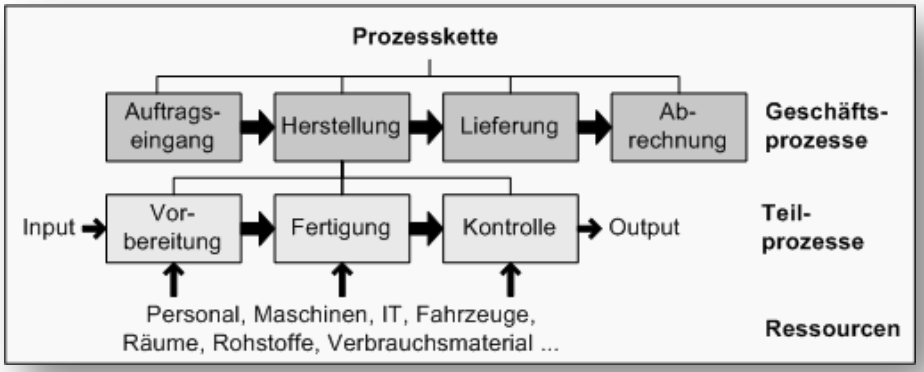
Buchung einer Dienstreise

Schadensfallabwicklung bei der Versicherung

Mitarbeiterrechner einrichten

Geschäftsprozesse

- Jeder Prozess besteht aus einer Folge von Aktivitäten
- Vereinfacht als Prozesskette dargestellt:



https://www.bsi.bund.de

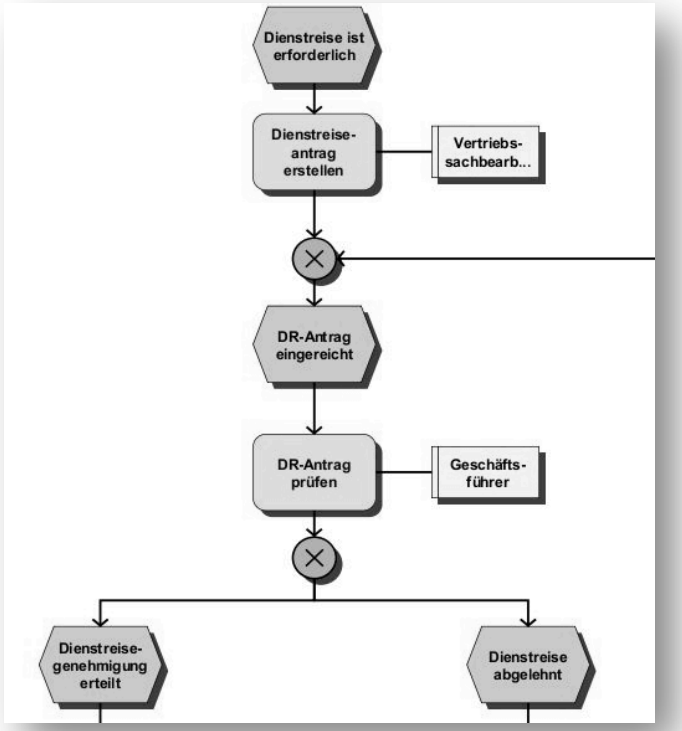
Geschäftsprozessmanagement

> *„Im Rahmen des Geschäftsprozessmanagements werden Abläufe in Unternehmen beschrieben, dokumentiert, optimiert und überwacht.“*
> [Kocian]

- Analogie Boxenstop [Kocian]:
  - Fahrer ist Kunde
  - Leistung des Boxenteams ist ein Dienst
  - Prozess startet bei Ankunft an Box
    - Reifen abmontieren
    - Neue Reifen montieren
    - Tanken
    - …
  - Prozess ist messbar (Sekunden)
  - Prozess kann optimiert werden (z.B. durch anderen Ablauf, Reihenfolge, Parallelisierung)
    - Iterativer Prozess der Verbesserung

Geschäftsprozessmodellierung

- GPM bzw. BPM (Business Process Modeling)
- z.B. mit:
  - **EPK**: ereignisgesteuerte Prozesskette
    - Bestandteil von SAP
  - **BPMN**: Business Process Model and Notation
- Fokus auf BWL (nicht auf Software)
  - Als Informatiker bzw. zur Beschreibung der technischen Sicht verwendet man eher UML, Programmablaufpläne, Petrinetze,...

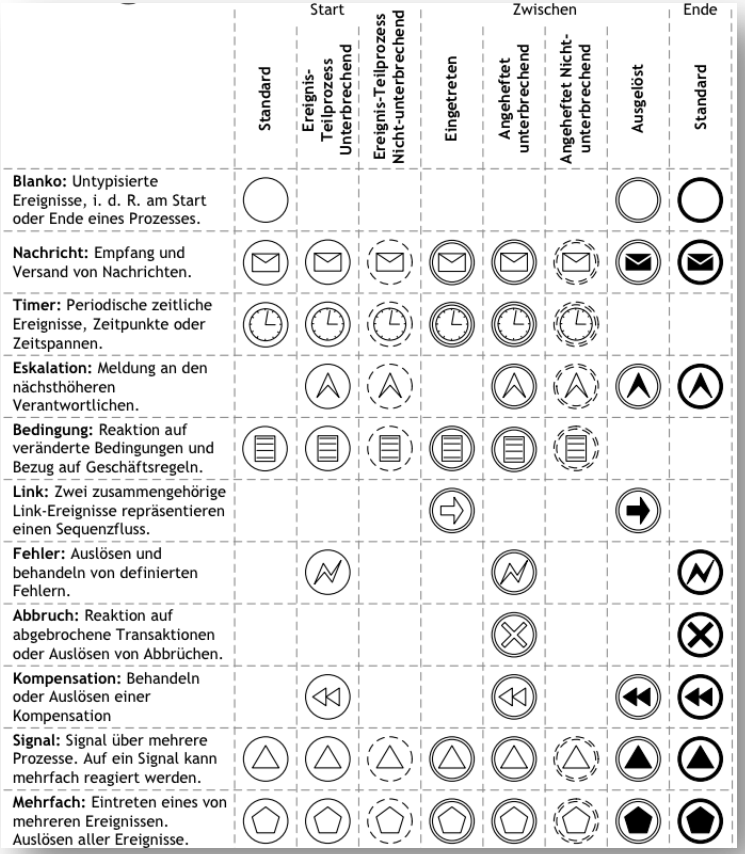


padtank.de

EPK-Beispiel

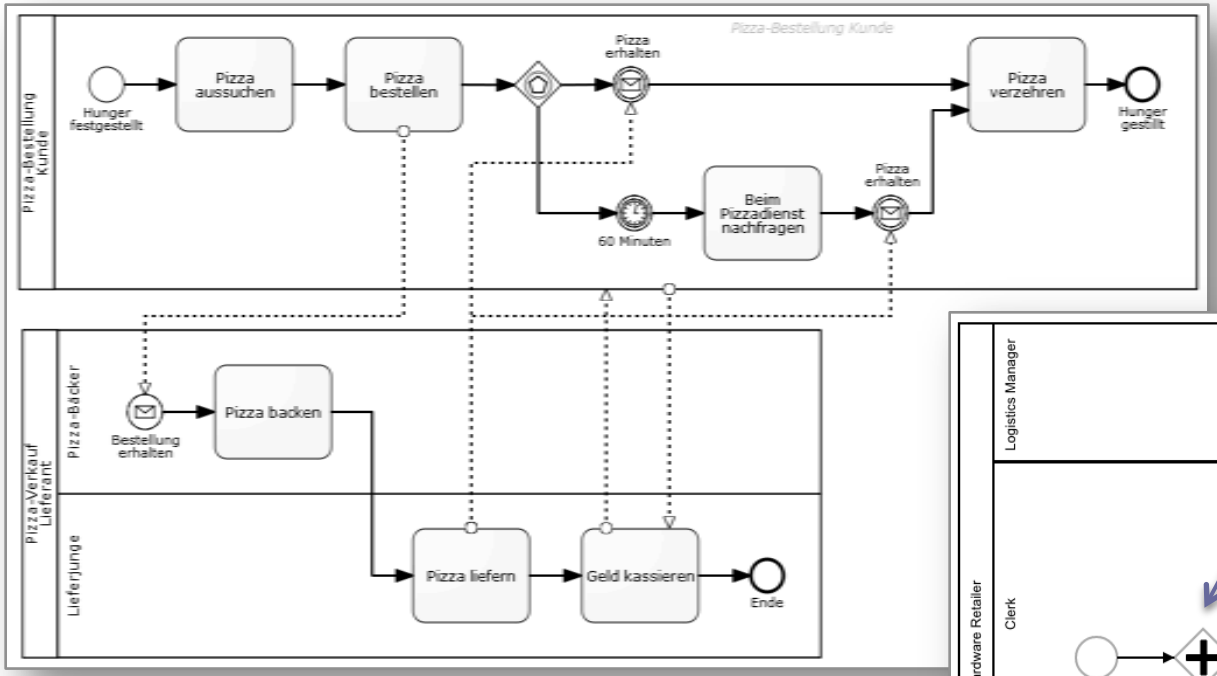
BPMN

- Grafische Sprache zur Definition von Arbeitsabläufen (Geschäftsprozessen)
- 2001, IBM
- Seit 2011: BPMN 2.0
  - Speicherung als XML (dadurch austauschbar)
  - Ziel: BPMN soll ausführbar sein
    - Angeblich soll BPEL (Business Process Execution Language, siehe Teil II) dadurch an Bedeutung verlieren
    - Allerdings gibt es eine große Lücke zwischen einer abstrakten Definition eines Geschäftsprozesses und ausführbarem Code!

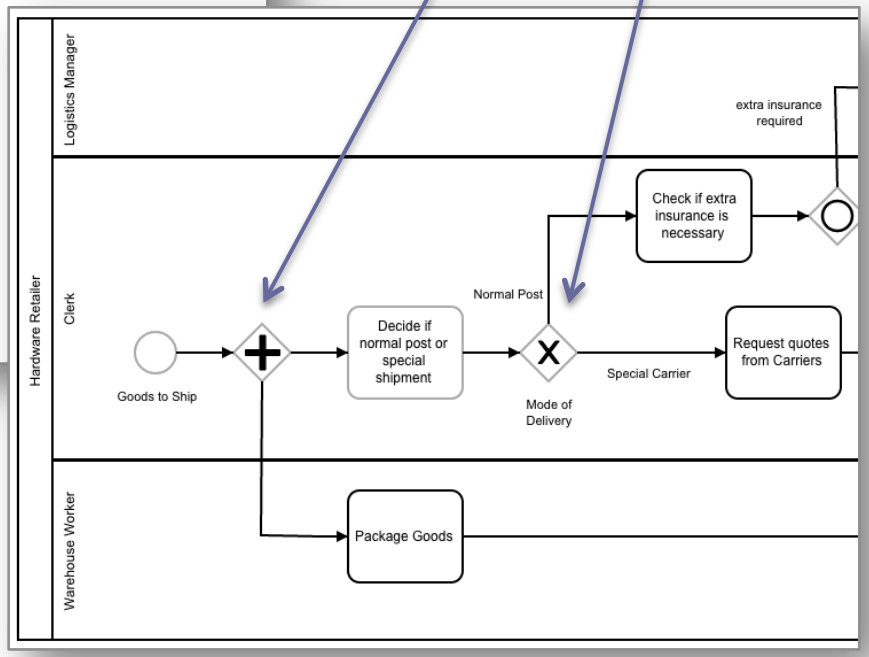


http://www.bpmb.de/images/BPMN2_0_Poster_DE.pdf

BPMN



http://www.heise.de/developer/meldung/BPMN-2-0-fuer-eine-bessere-Zusammenarbeit-zwischen-Fachabteilung-und-IT-1175099.html



http://camunda.org/bpmn/tutorial/

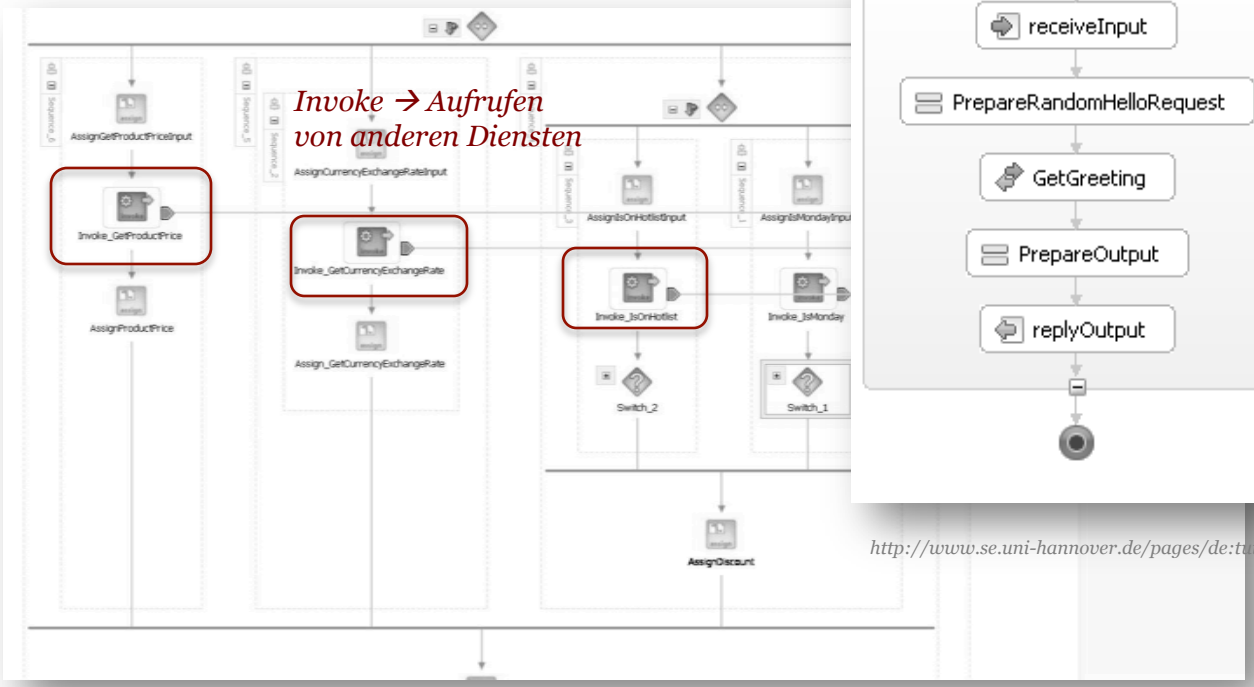
BPMN → Software

- Ziel ist es, diese abstrakten aus der Wirtschaft stammenden Prozessbeschreibungen in Software zu realisieren
- Hierbei finden wir ein „Gap“ vor: nicht jede Beschreibung lässt sich 1:1 in Software übertragen
  - Bsp.: Entscheidung welche Spedition genutzt wird kann einen komplexen Algorithmus mit mehreren WebServcie-Aufrufen und Subprozessen beinhalten (wiegen, Speditionspreise vergleichen, Auslastung und Lieferdauer berücksichtigen, Entscheidung treffen)
  - Bsp.: Nicht alle Bestandteile sind komplett durch Software automatisierbar (z.B. Verpacken von Produkten erfordert evtl. menschliches Eingreifen)
- Beachtung des Dienstegedankens
  - Wiederverwendbarkeit
    - Modularisierung
    - Kapselung

BPEL

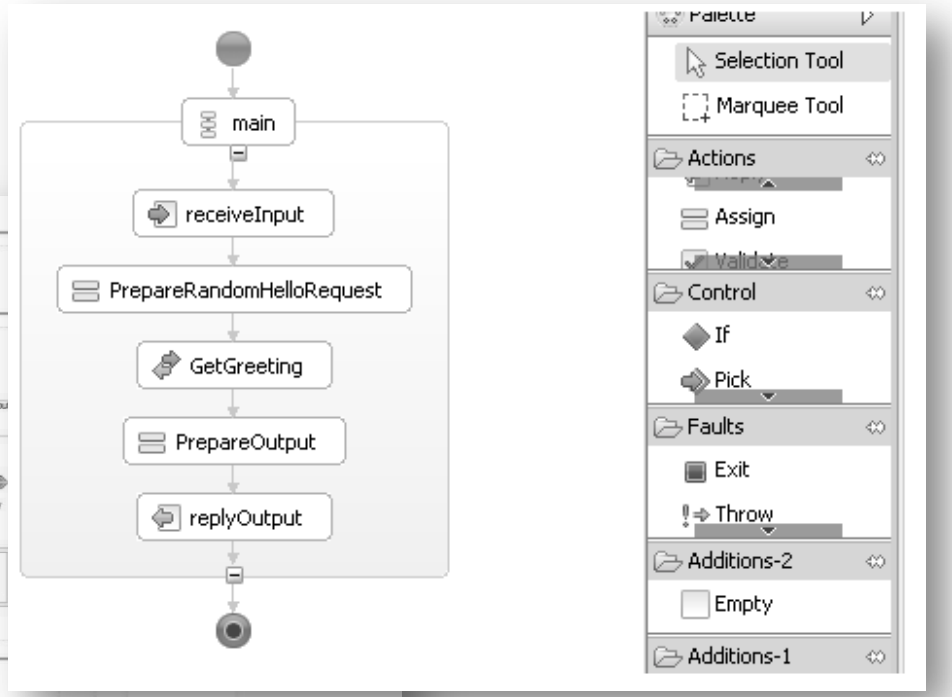
- Geschäftsprozesse als Webservices beschreiben
- BPEL Prozess selbst ist ein Webservice (SOAP)
- BPEL Prozess interagiert mit anderen Webservices
  - BPEL orchestriert Webservices (hat also die Kontrolle)
- XML-basierter Standard
  - Wird in der Regel visualisiert (jedoch keine primär grafische Sprache, d.h. grafische Elemente nicht standardisiert)
- Unterstützt XPath, XSLT und beinhaltet Kontrollstrukturen (Bedingungen, Schleifen), Variablen, Exception-Handling
- Ist ausführbar (BPEL Engine)

BPEL (Beispiel)



Oracle SOA Suite

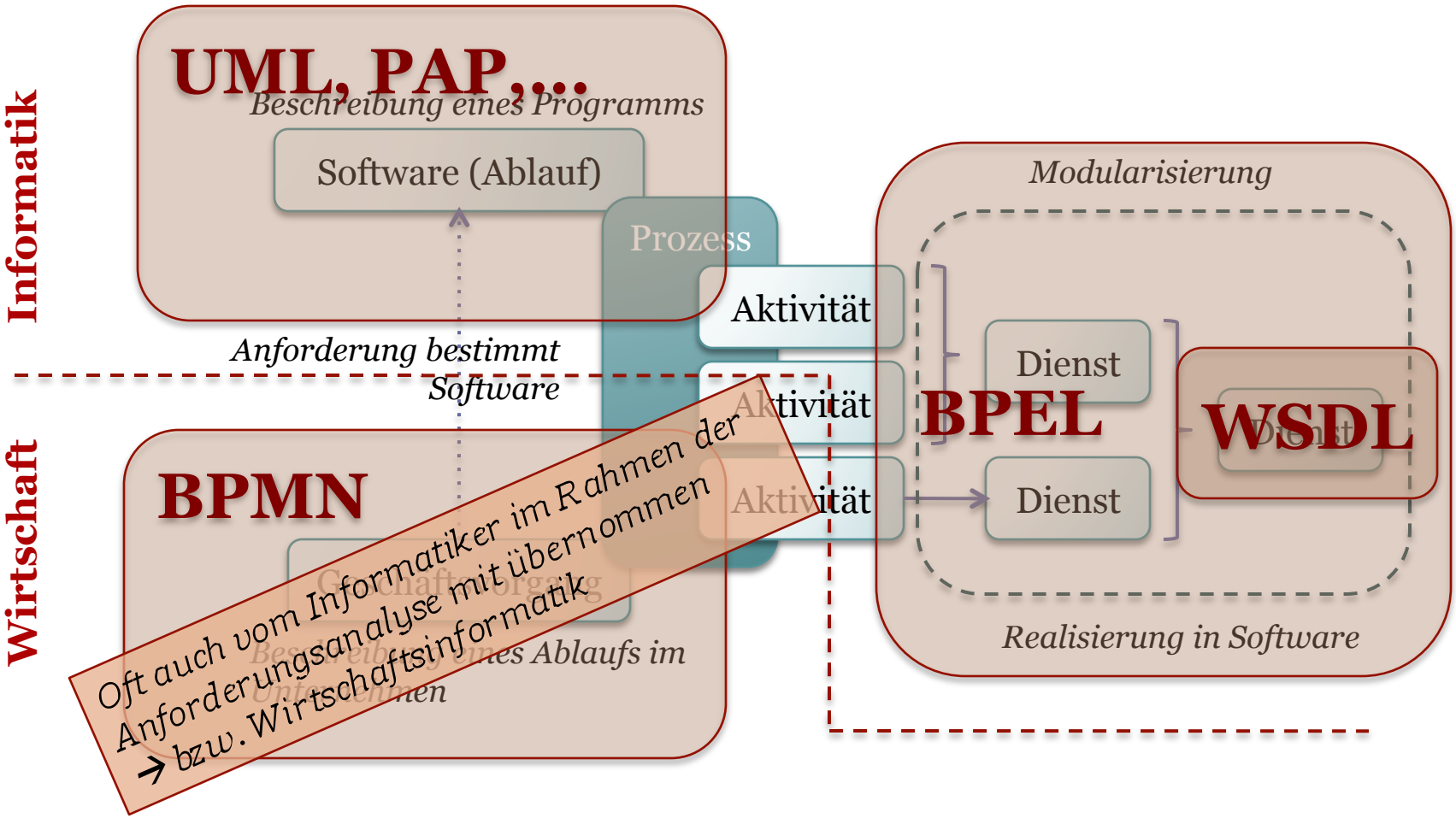
http://technology.amis.nl/2005/04/20/implementing-a-business-process-in-bpel-using-oracle-bpel-10g-beta-3-getsalesquote-leverages-plsql-and-webservices/



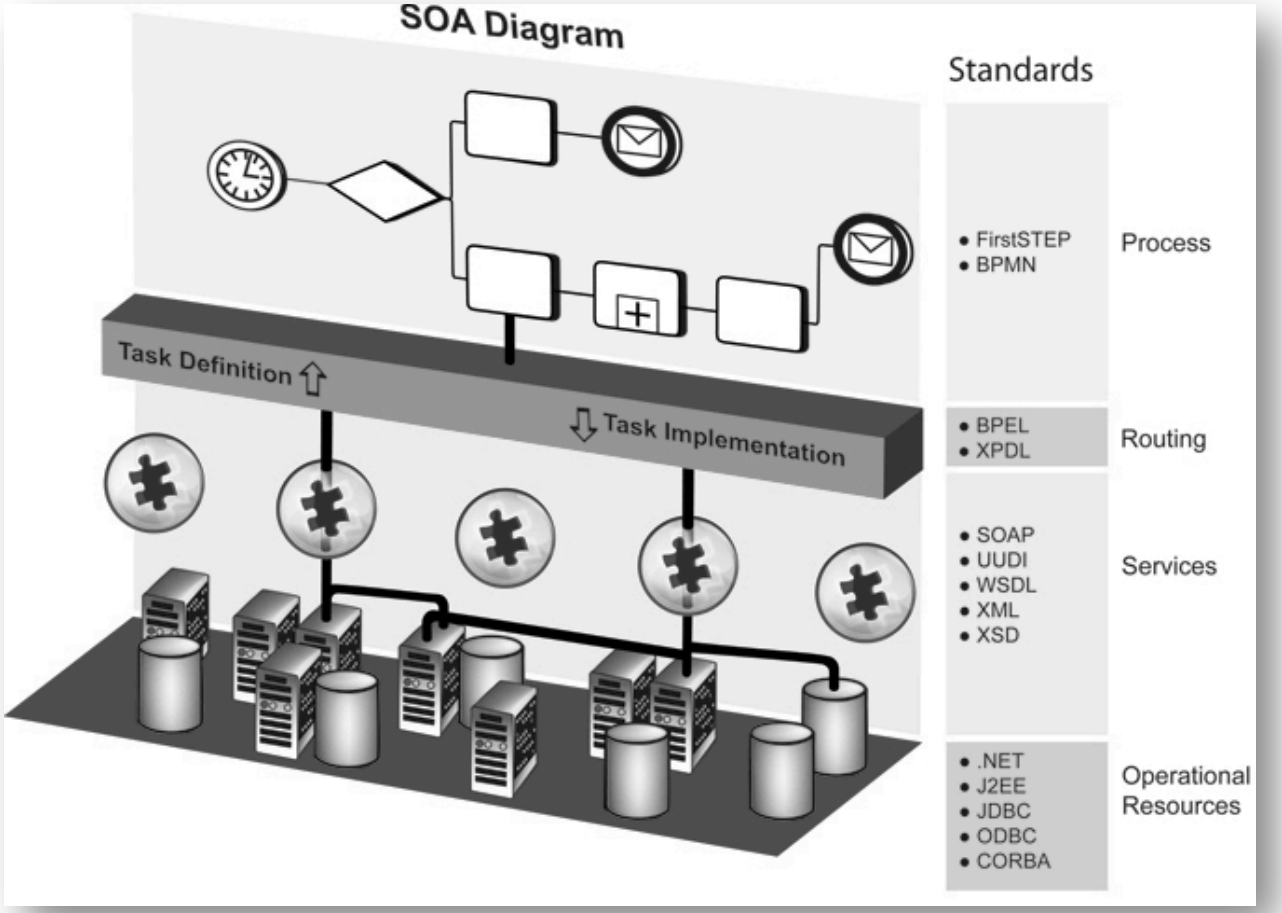
http://www.se.uni-hannover.de/pages/de:tutorials_bpel_ode_simpleinvoke

Apache ODE

Überblick


Informatik
Wirtschaft
UML, PAP,...
Beschreibung eines Programms
Software (Ablauf)
Anforderung bestimmt Software
BPMN
Beschreibung eines Ablaufs im Unternehmen
Prozess
Aktivität
Aktivität
Aktivität
Modularisierung
BPEL
Dienst
Dienst
WSDL
Realisierung in Software
Oft auch vom Informatiker im Rahmen der Anforderungsanalyse mit übernommen
→ bzw. Wirtschaftsinformatik

Überblick (II)



http://www.interfacing.com/uploads/images/soa3.jpg

Beschreibung (BWL)

↓

Beschreibung (Software)

↓

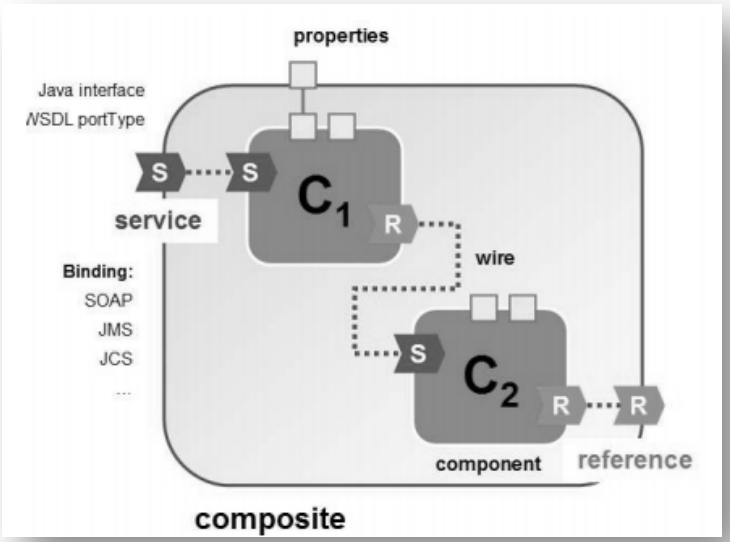
Implementierung (Software)

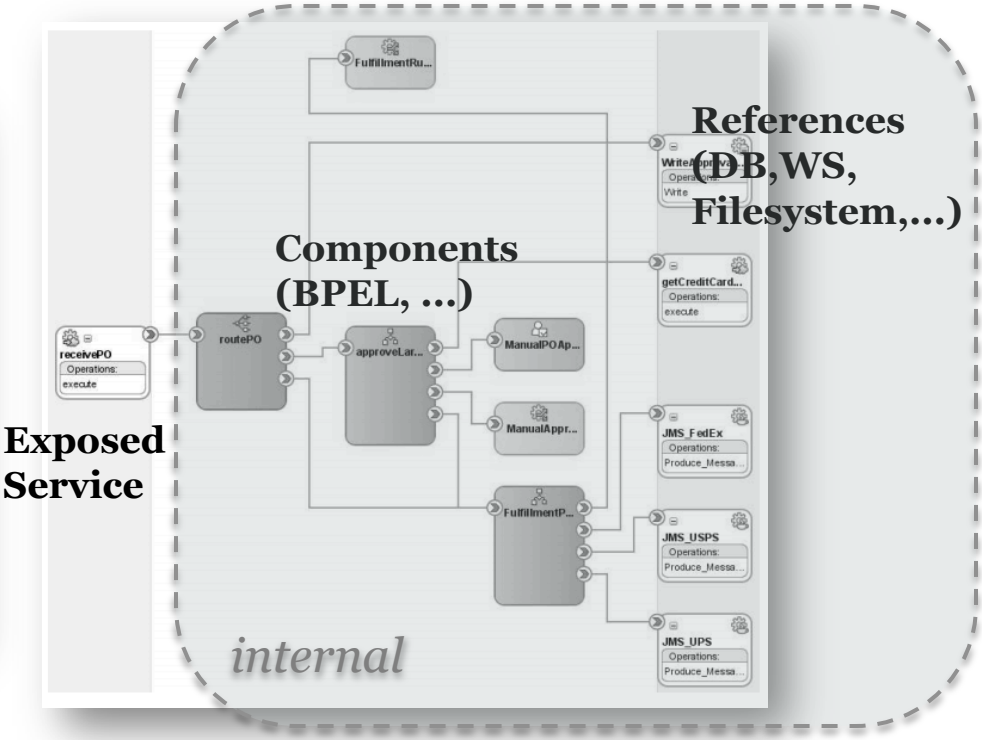
SCA (Service Component Architecture)

- Beschreibung von serviceorientierten Architekturen
- BPEL als Orchestrierung von Webservices
  - Aufruf externer Dienste (definiert außerhalb des BPEL-Prozesses)
- SCA befindet sich auf einer höheren Ebene
  - Entspricht einem Container („Composite“) der Komponenten enthält
    - Enthält mehrere Dienste (z.B. BPEL-Prozesse)
    - D.h. kapselt BPEL nochmals
      - BPEL verbindet WS, SCA verbindet BPEL (und mehr)
  - Externe Dienste werden über Referenzen eingebunden
  - Beschränkt sich nicht auf Webservices, sondern über Adapter können auch Datenbanken etc. angebunden werden
  - Basiseinheit ist das „Composite“
    - Wird ebenfalls als Webservice (nach außen) zur Verfügung gestellt
    - Komponenten nach außen i.d.R. nicht sichtbar

SCA

„[…] a specification that describes how the various enterprise pieces are created and assembled together as modular components […]“





http://www.oracle.com/technetwork/topics/entarch/whatsnew/oracle-sca-the-power-of-the-composi-134500.pdf

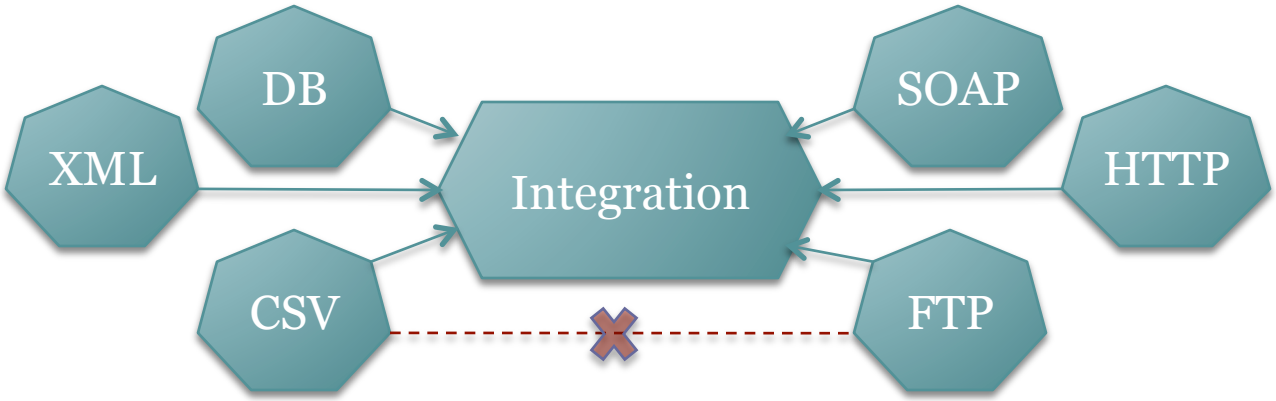
- Jede Komponente wird als Service bereitgestellt und nutzt externe Services (Referenzen)

Integration

- Verknüpfung von (existierenden, heterogenen) Anwendungen, sodass sie als koordiniertes Ganzes arbeiten
  - „Kleber“ zwischen den Komponenten
    - Schaffen von Schnittstellen zu Software, die keine Schnittstellen hat
    - Transformation von Datenformaten und Protokollen
    - Zentraler Punkt zur Vermittlung zwischen Anwendungen
- Enterprise Application Integration (EAI)
  - Verknüpfung von Anwendungen innerhalb eines Unternehmens
    - Buchhaltung
    - CRM (Customer Relationship Management)
    - Lagerhaltung
    - ...
  - Wenn SOA streng umgesetzt wird, theoretisch nicht nötig, da es bereits definierte Schnittstellen gibt
    - In der Praxis müssen jedoch Anwendungen miteinander interagieren, die nicht SOA-konform sind
    - Durch EAI kann jedoch eine SOA entstehen
- B2B (Business to Business) Integration
  - Integration über Unternehmensgrenzen hinweg

Integrationsplattform

- Ziel: Anwendungen müssen Daten austauschen können
- Anwendungen sind heterogen
  - Verschiedene Programmiersprachen, Betriebssysteme, Datenformate, Datenquellen, Schnittstellen, Protokolle, …
- Die Integration von Anwendungen erfolgt über eine Integrationsplattform
  - Besitzt eine Vielzahl von Schnittstellen
    - Zu Datenbanken, SOAP, REST, FTP,…
  - Integration beinhaltet auch Datentransformation



Middleware

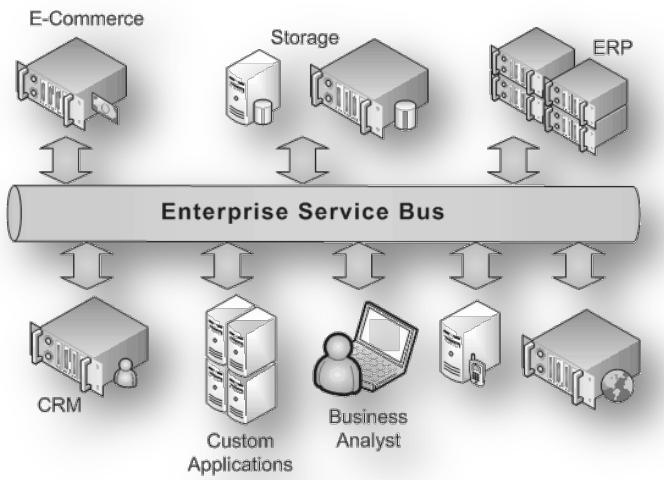
- Integrationsplattform ist eine Middleware
  - Bietet Funktionalitäten an (über die des Betriebssystems hinaus), sodass Anwendungen einfacher kommunizieren können
    - Über die Middleware; Anwendungen kommunizieren nicht direkt miteinander
  - Zwischenschicht, die Funktionalitäten durch Kombination von Funktionen niederer Ebene zur Verfügung stellt
    - Verbergen von Komplexität
  - Oft verteilte Anwendungen → SOA/SCA
  - z.B. *Oracle Fusion Middleware*

(Enterprise) Service Bus (ESB)

- Begriffe sehr nah beieinander und schwer unterscheidbar
  - Integrationsplattform ~ Middleware ~ Service Bus
  - Integrationsplattform ist eine Middleware, die als Service Bus realisiert sein kann

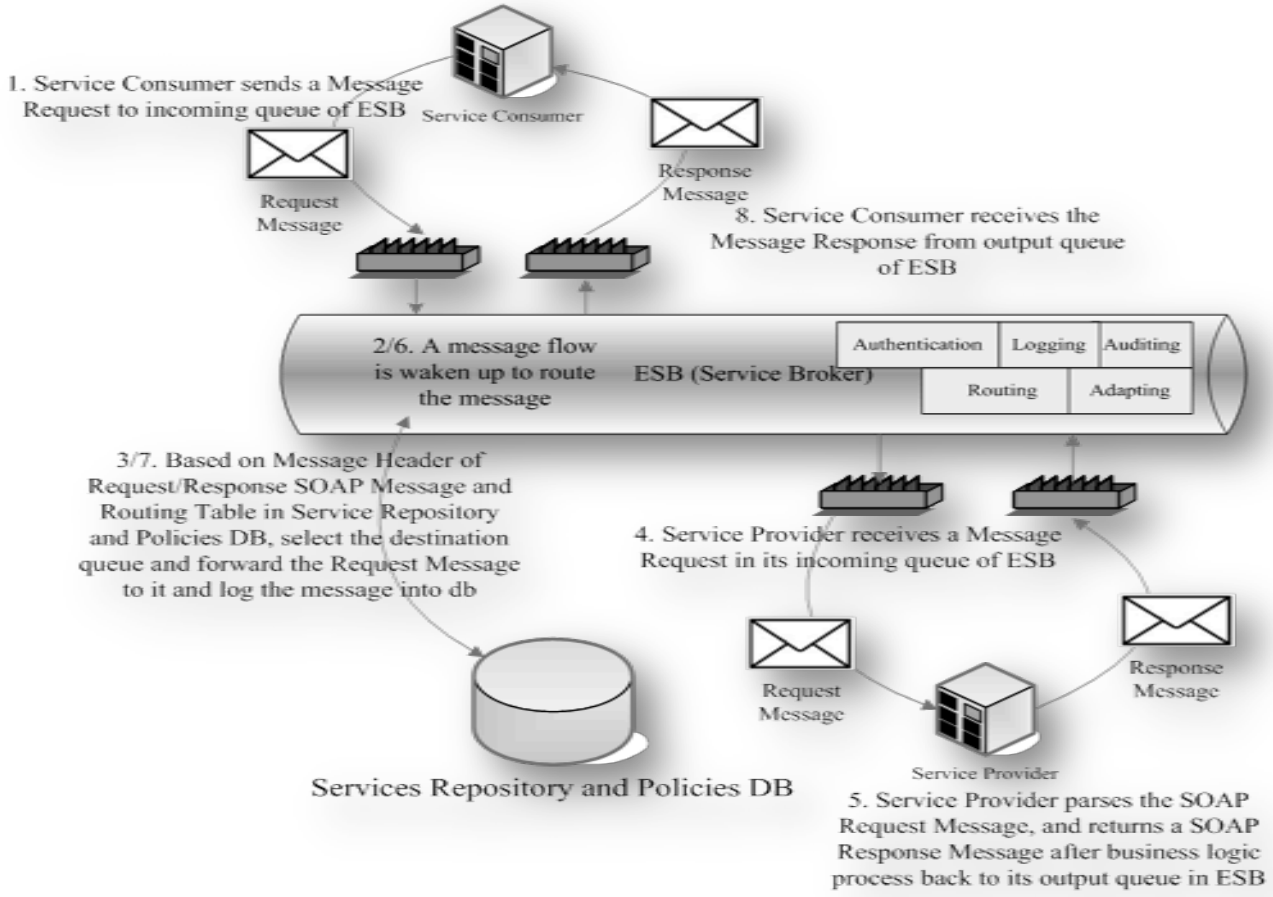
> *„Ein ESB stellt eine Middleware-Integrationsinfrastruktur dar und wird oft als Grundlage für SOA verwendet.“*
> *[Torsten Horn]*

- Weitere Abstraktionsschicht, die den Zugriff auf Anwendungen / Dienste koordiniert
- Vergleichbar mit Hardware-Bus
  - Teilnehmer kommunizieren über den Bus (senden Nachrichten)
  - Keine Point-to-Point-Kommunikation
- Fokussiert auf Routing und Protokoll-transformation der Nachrichten



http://www.centeractive.com/de/content/enterprise-service-bus

Service Bus: Routing



http://www.exadelfs.com/images/enterprise_service_bus.png

Verwirrung?

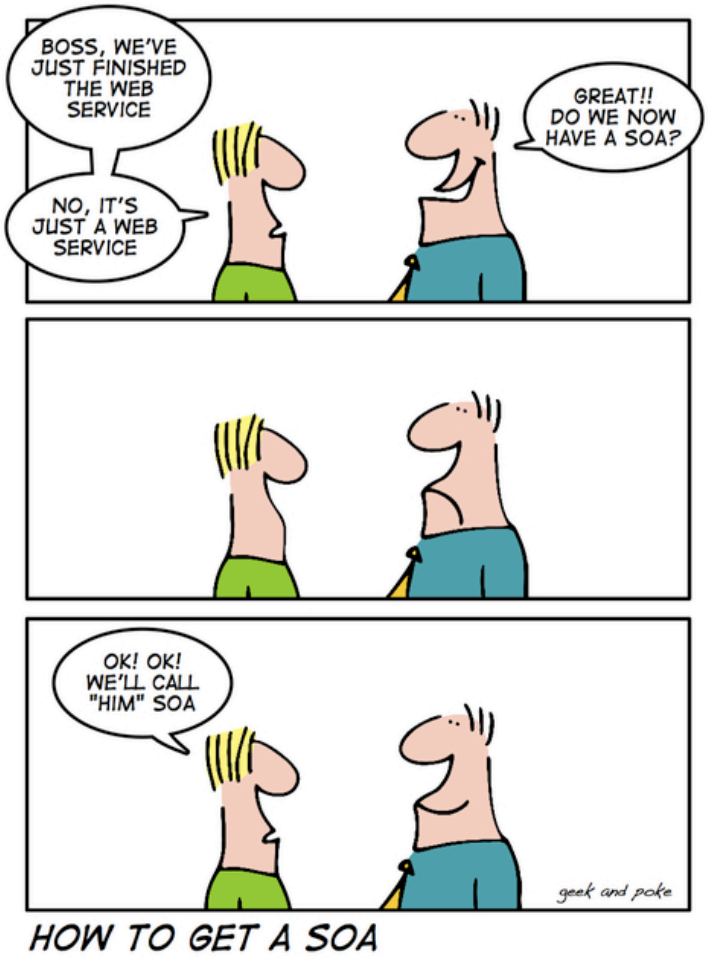
- Was sollten wir uns merken?
  - In einer SOA werden Anwendungen als Dienste (oft SOAP) angeboten
    - Dies führt zu einer starken Modularisierung und Wiederverwendbarkeit
  - Dienste bilden oft Geschäftsprozesse ab (BPMN)
  - Dienste rufen andere Dienste auf
    - Der Aufruf wird koordiniert → Orchestrierung (BPEL)
  - Dienste basieren (meist) auf gleichem Protokoll / Standard (SOAP)
    - Trotzdem muss zwischen ihnen vermittelt werden (unterschiedliche Schemata → Datentransformation)
      - BPEL / XSLT → auf einer Middleware (mit BPEL-Engine) ausgeführt
  - Es gibt eine Zeit vor SOA: Es existieren Anwendungen mit denen interagiert werden muss, obwohl sie nicht als Dienste zur Verfügung stehen
    - Integration notwendig
      - Eine Integrationsplattform ist eine Anwendung einer Middleware
      - Bietet über Adapter Zugriff auf verschiedene Systeme (nicht nur WS, auch FTP, Filesystem, DB,...)
  - Der Nachrichtenaustausch kann über einen Service Bus erfolgen

SOA Facts ;-)

„One person successfully described SOA completely, and immediately died“

„SOA is the only thing Chuck Norris can't kill.“

soafacts.com



http://geekandpoke.typepad.com/.shared/image.html?/photos/uncategorized/2008/07/12/itsnosoa.jpg

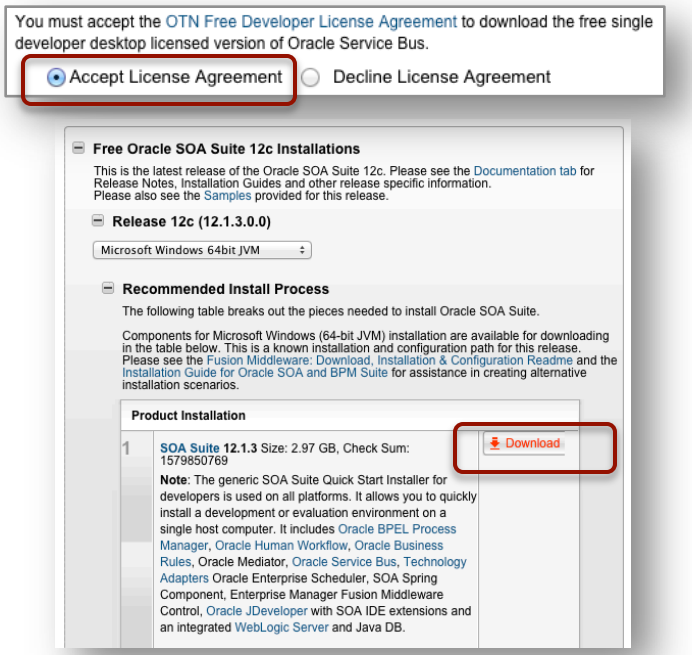
Quellen und weiterführende Literatur

- http://www.mulesoft.com/resources/esb/service-orchestration-and-soa
- http://msdn.microsoft.com/en-us/library/aa480021.aspx
- https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzSchulung/Webkurs1004/3_BusinessImpactAnalysieren/1_GeschaeftsprozesseBestimmen/GeschaeftsprozesseBestimmen_node.html
- http://iwi.wiwi.hu-berlin.de/~viehweger/wi-geschaeftsprozess.pdf
- https://www.hs-neu-ulm.de/fileadmin/user_upload/Forschung/HNU_Working_Paper/HNU_WP16_Kocian_Geschaeftsprozessmodellierung.pdf
- http://www.bpmn.org/
- http://www.bpmb.de/images/BPMN2_0_Poster_DE.pdf
- http://www.bpmn-tool.com/tutorial/
- http://www.se.uni-hannover.de/pages/de:tutorials_bpel_ode_simpleinvoke
- https://dpunkt.de/leseproben/3340/4_Business%20Process%20Execution%20Language%20(BPEL).pdf
- http://www.oracle.com/technetwork/topics/entarch/whatsnew/oracle-sca-the-power-of-the-composi-134500.pdf
- http://www.mulesoft.com/resources/esb/integration-middleware-technology
- http://www.torsten-horn.de/techdocs/soa.htm

Installation Oracle SOA Suite

Oracle Fusion Middleware

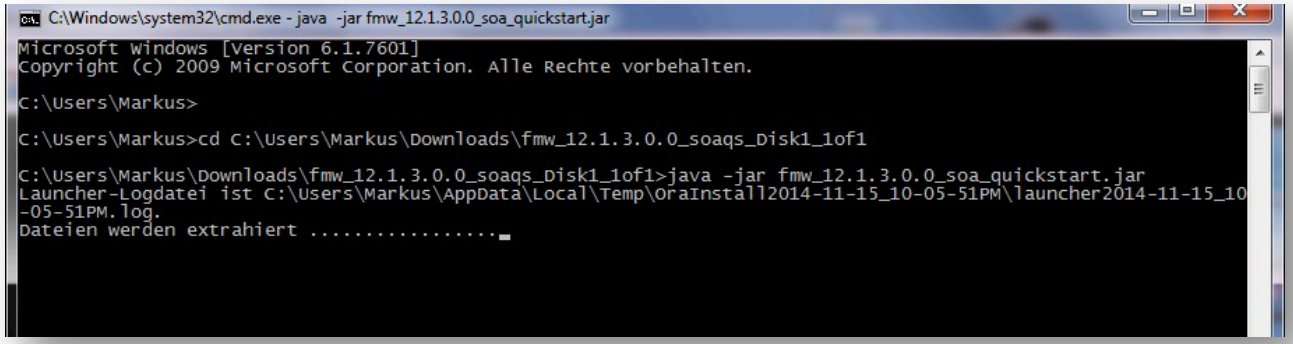
- Oracle SOA Suite 12c
  - http://www.oracle.com/technetwork/middleware/soasuite/downloads/index.html
  - Kostenlose Entwicklerversion (nicht kommerziell)
  - License Agreement akzeptieren
  - Oracle Account notwendig zum Download



„[…] use the Programs only for the purpose of developing, testing […], prototyping and demonstrating your application(s), and not for any other purpose. […] You may not:
- use the Programs for your own internal data processing or for any commercial or production purposes, or use the Programs for any purpose except the development, testing, prototyping, and demonstrating of your application(s) […]“

Installation

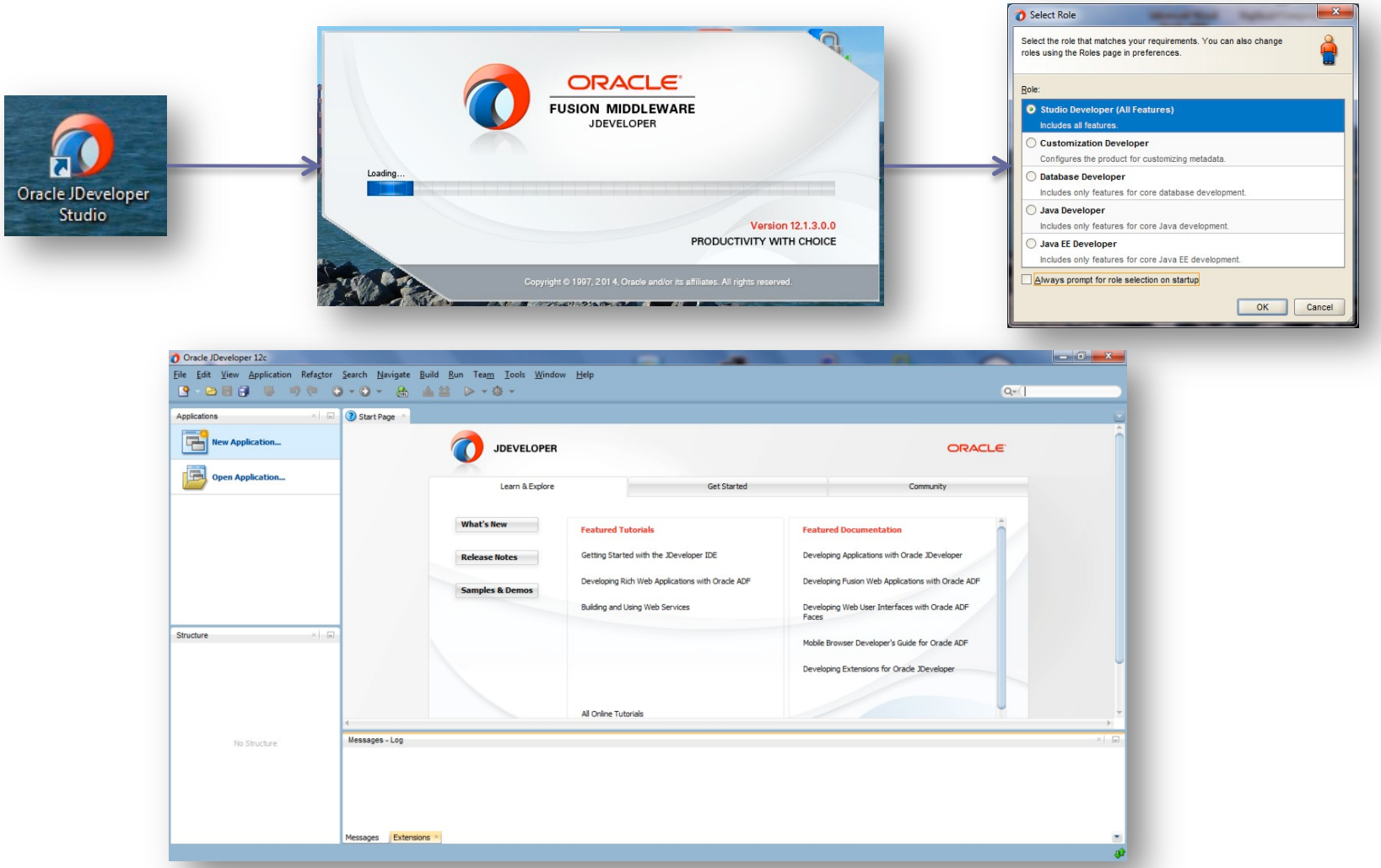
- ZIP entpacken
- JDK muss installiert sein (JRE reicht nicht aus)
- Kommandozeile mit Administratorrechten öffnen
- Wechseln in das entsprechende Verzeichnis (wo Download entpackt wurde)
- Starten der jar-Datei (mit JDK)



Installation

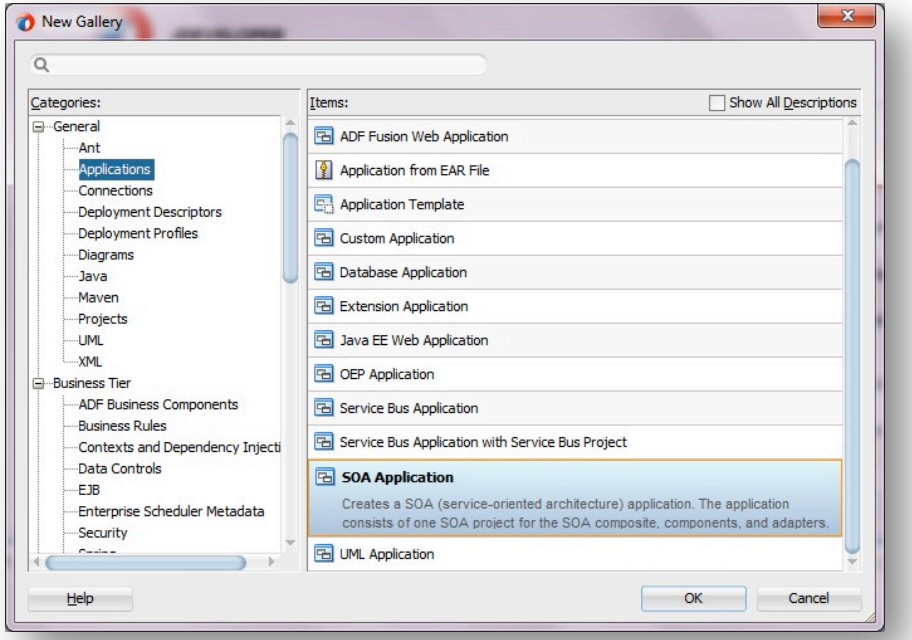
- Assistent öffnet sich…

Erster Start

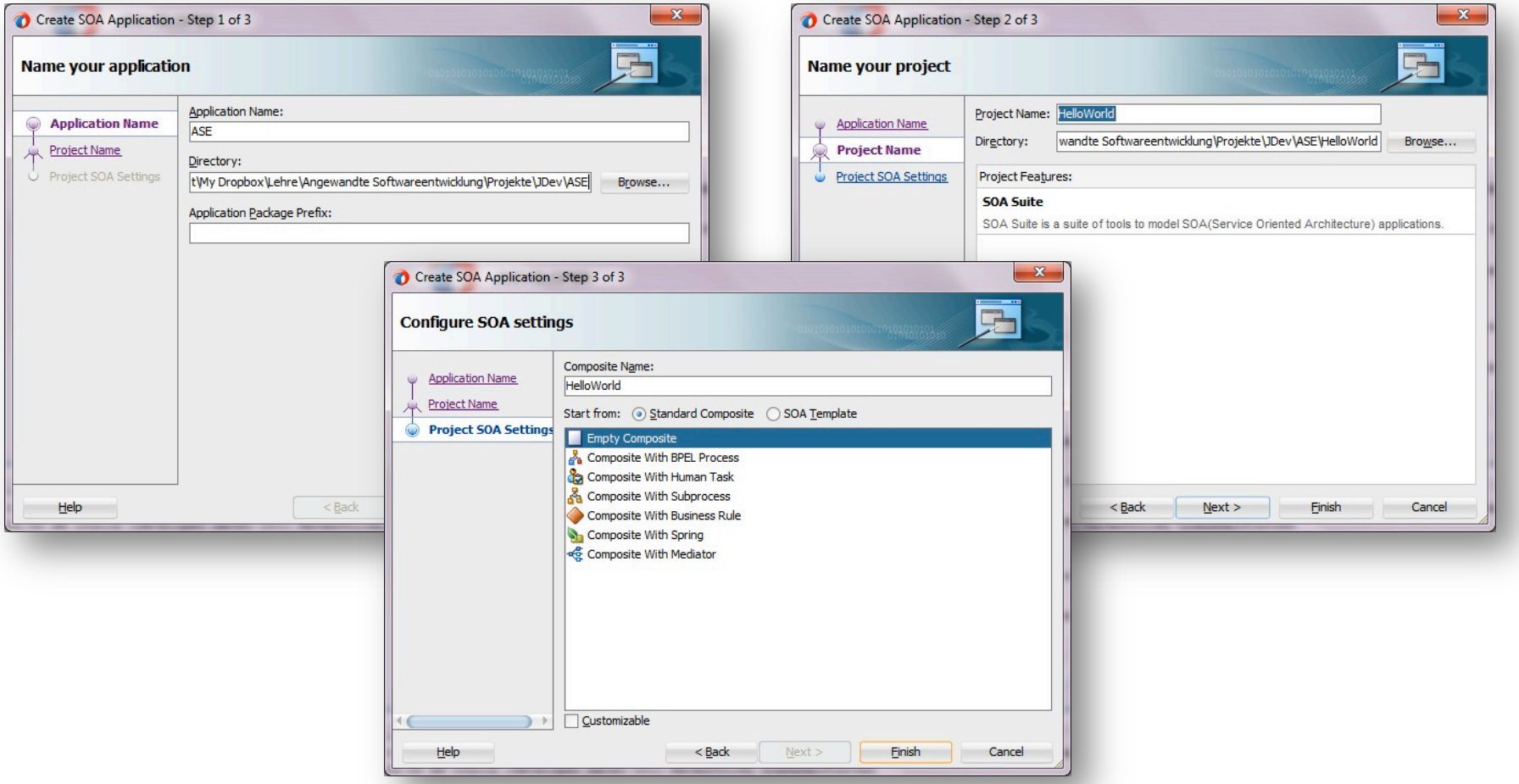


Erstes „Projekt“

- Projekte heißen hier „Applikationen“
  - Und Applikationen enthalten „Projekte“ ;-)
- New Application
  - SOA Application

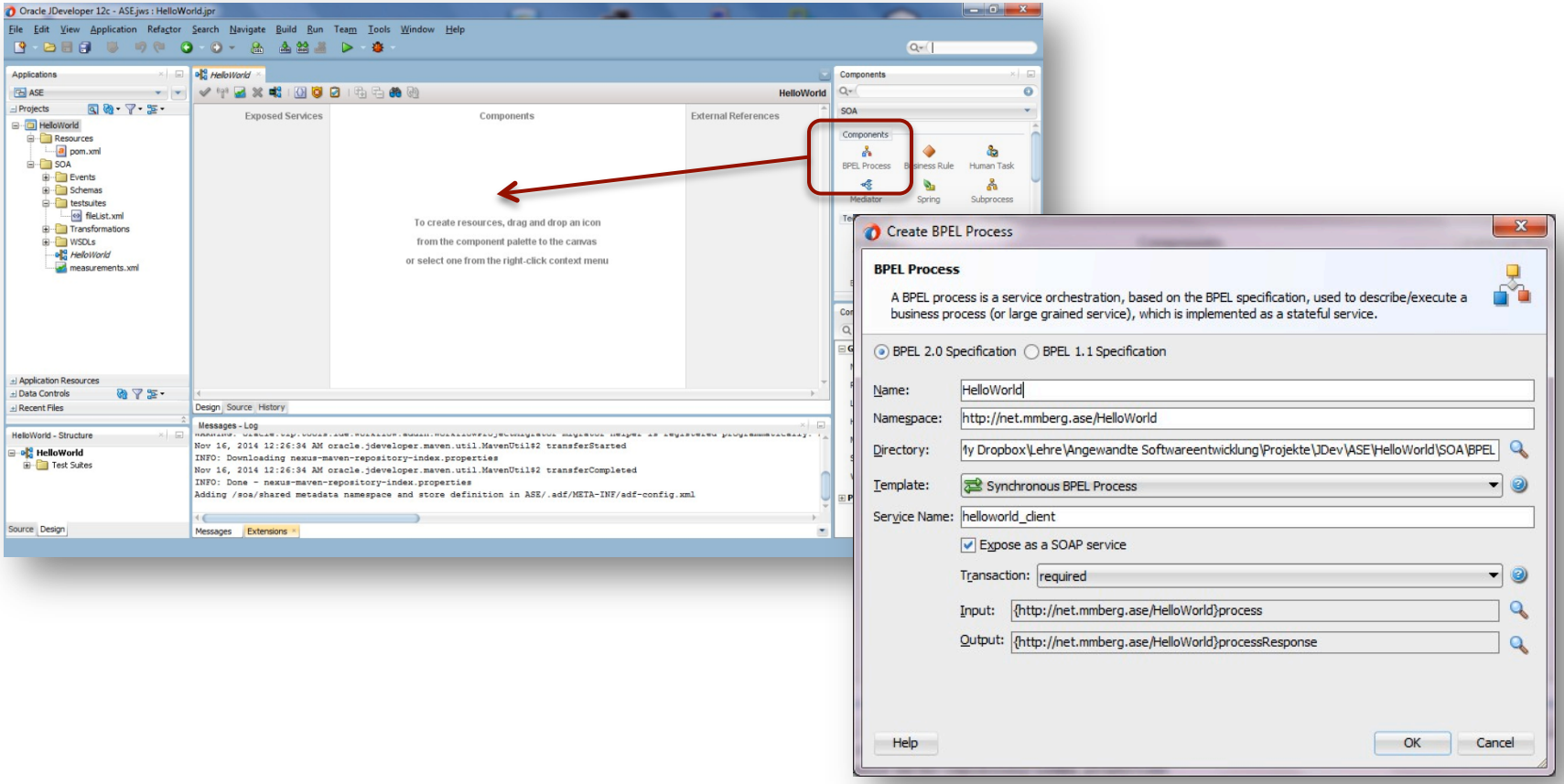


Applikation und Projekt erstellen

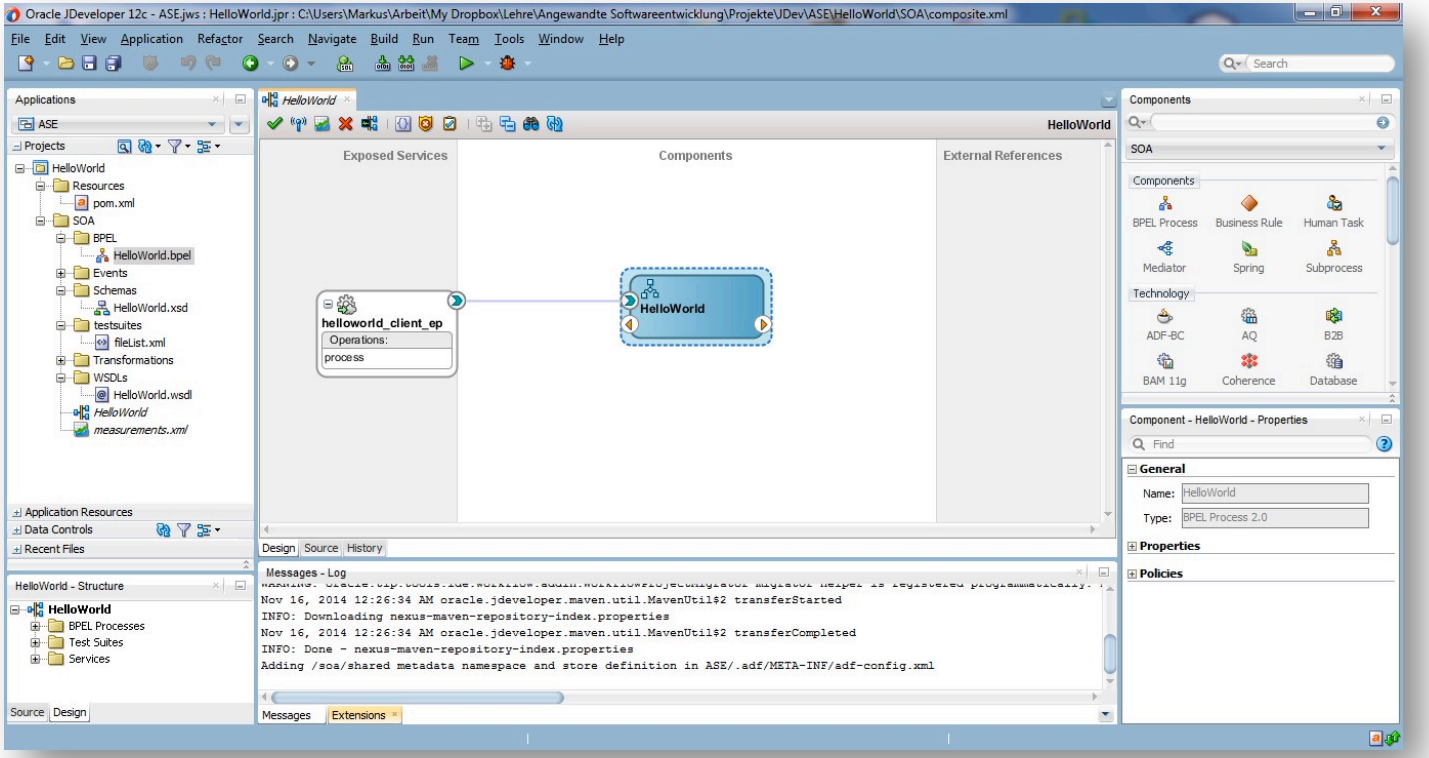


Leeres Composite

- BPEL Prozess erzeugen (in die Mitte ziehen)

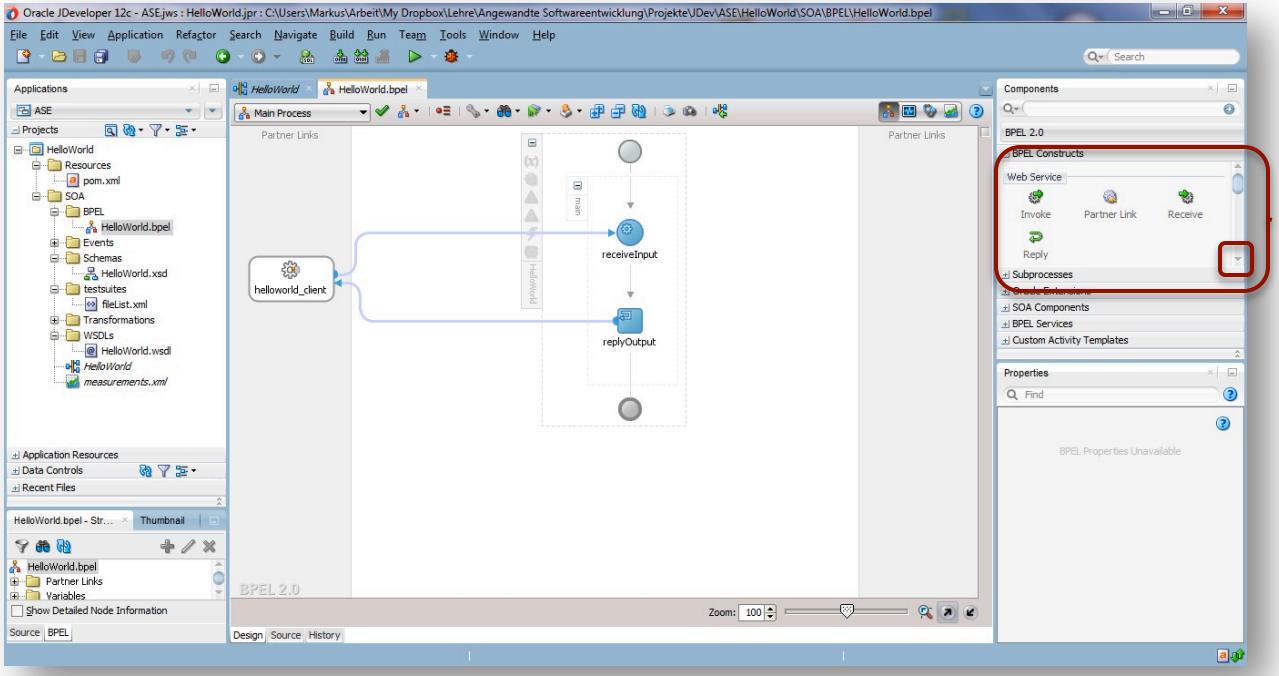


Erster BPEL-Prozess im Composite



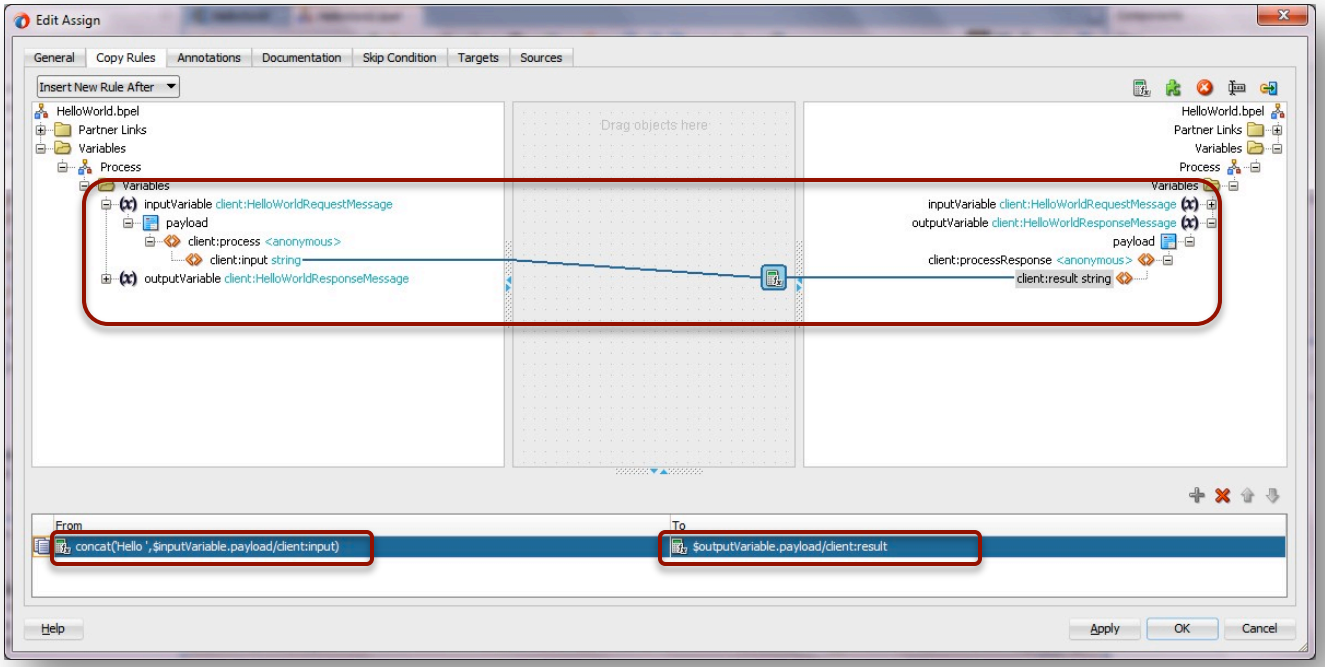
BPEL

- BPEL Prozess durch Doppelklick öffnen
- *Assign* hinzufügen

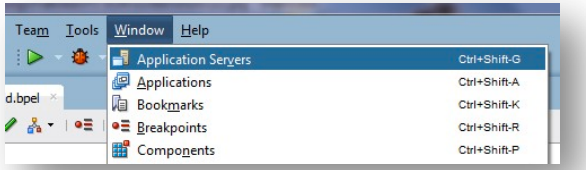


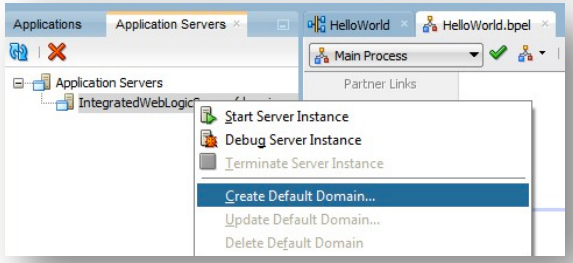
Assign konfigurieren

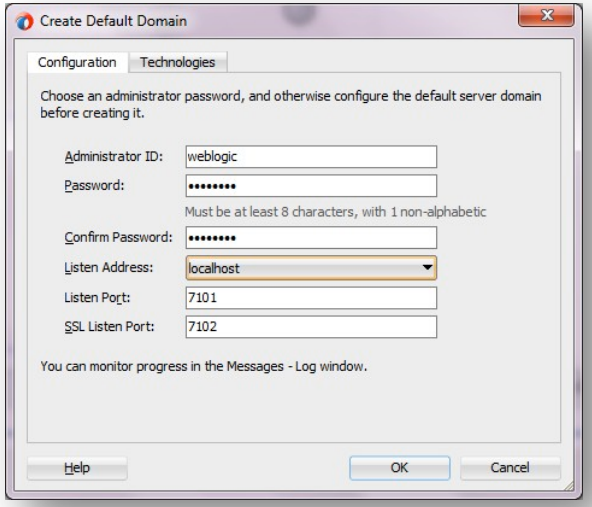
- Wir nutzen ein *concat* (XPath) und kombinieren „Hello“ mit dem Eingabewert und weisen das Resultat dem Ausgabewert zu

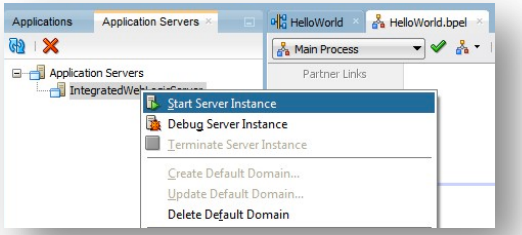


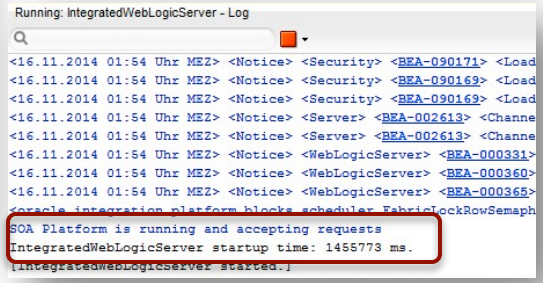
Application Server konfigurieren





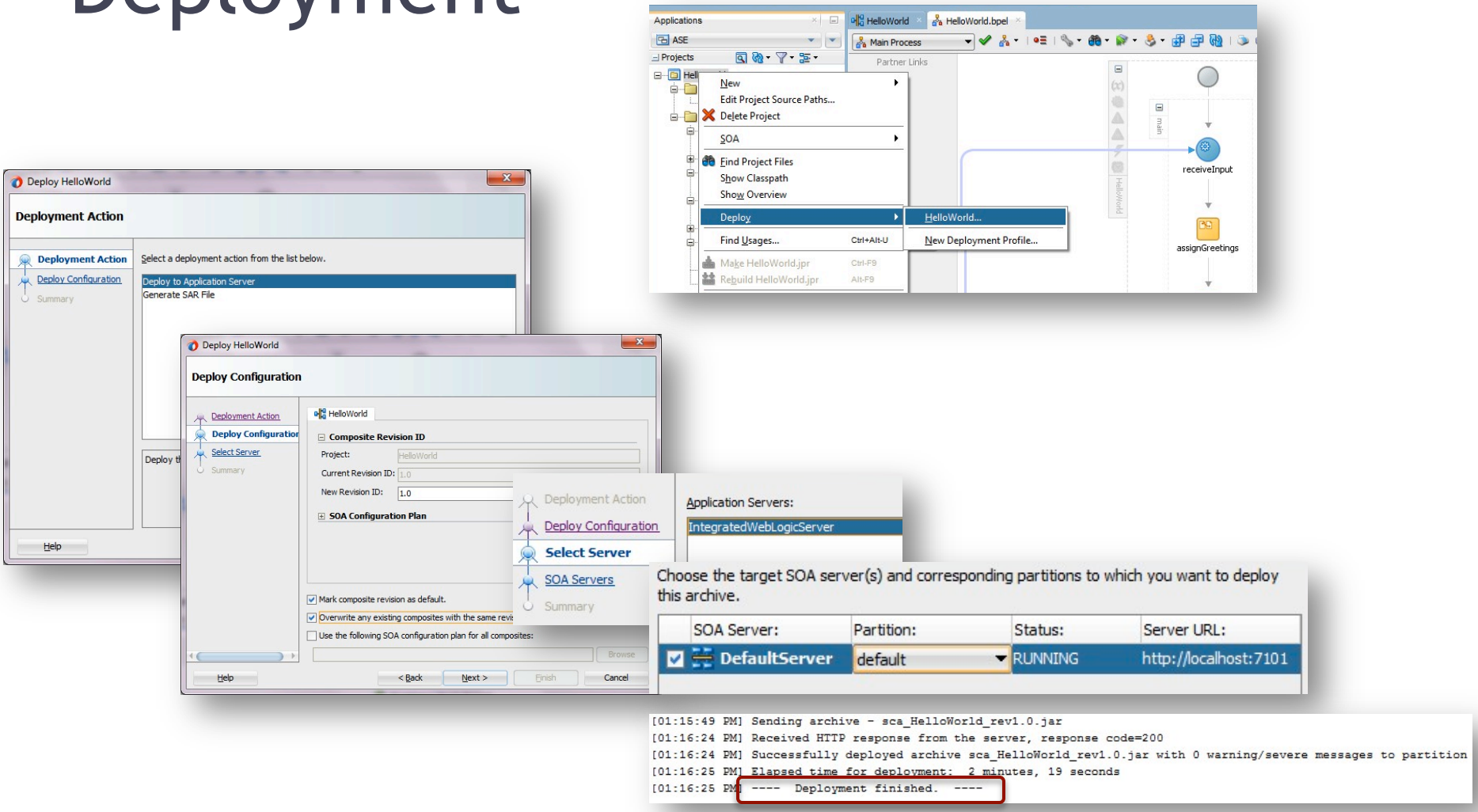






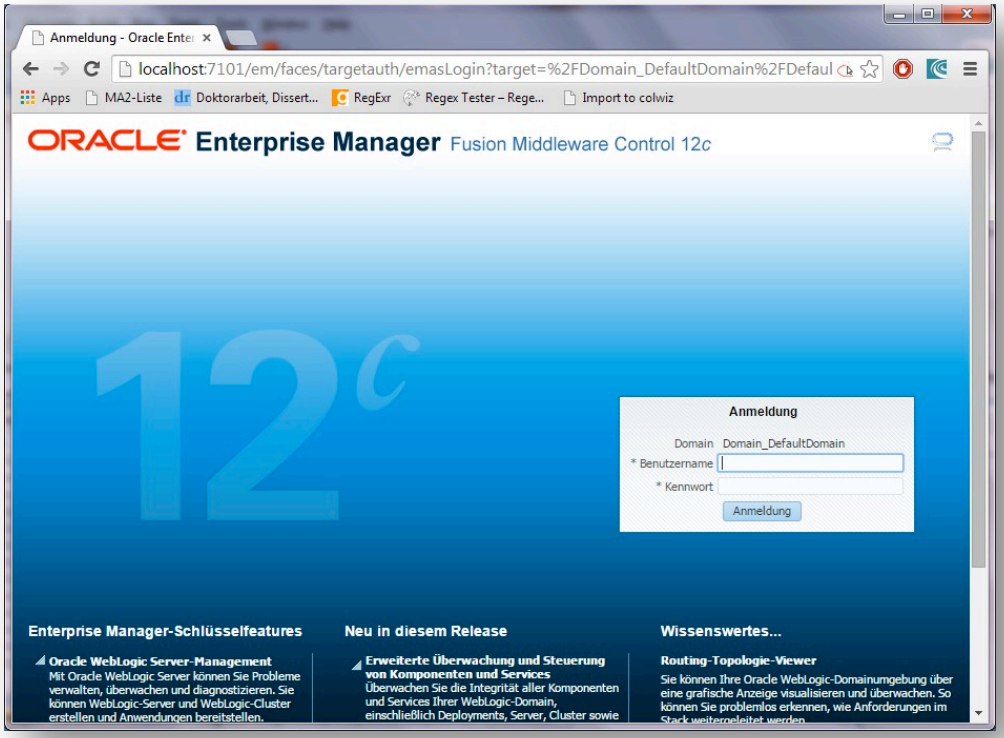
- Standard-Passwort: welcome1
- Listen-Address: localhost reicht aus
- Erstellen der Domäne dauert sehr lange (>15 Minuten keine Seltenheit je nach RAM)
- Starten des Servers ebenfalls sehr langwierig (hier: 24 Minuten)

Deployment

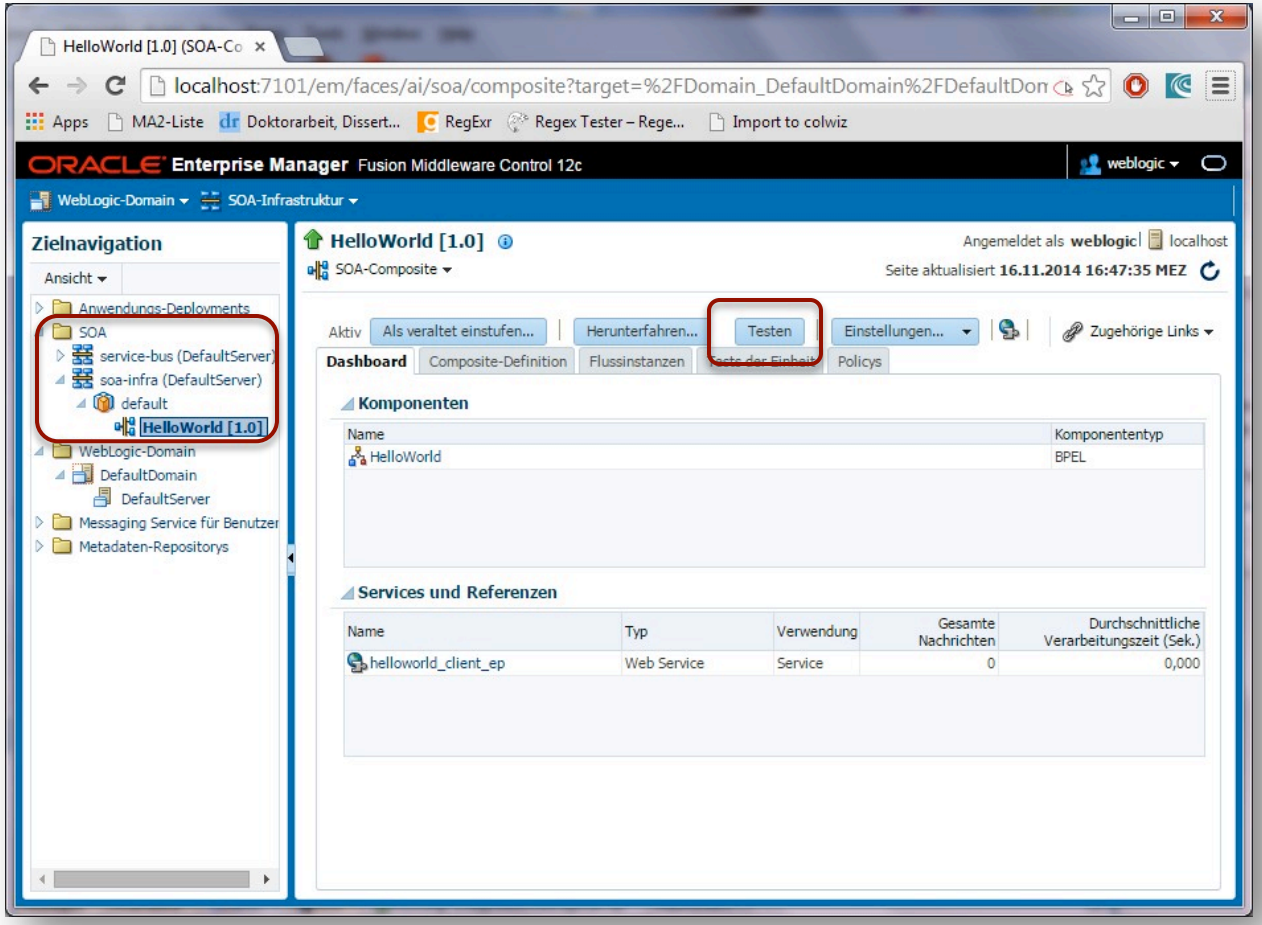

Applications
HelloWorld
HelloWorld.bpel
Main Process
Partner Links
New
Edit Project Source Paths...
Delete Project
SOA
Find Project Files
Show Classpath
Show Overview
Deploy
HelloWorld...
New Deployment Profile...
Find Usages... Ctrl+Alt-U
Make HelloWorld.jpr Ctrl-F9
Rebuild HelloWorld.jpr Alt-F9
receiveInput
assignGreetings
Deploy HelloWorld
Deployment Action
Select a deployment action from the list below.
Deploy to Application Server
Generate SAR File
Deployment Action
Deploy Configuration
Summary
Help
Deploy HelloWorld
Deploy Configuration
Deployment Action
Deploy Configuration
Select Server
Summary
HelloWorld
Composite Revision ID
Project: HelloWorld
Current Revision ID: 1.0
New Revision ID: 1.0
SOA Configuration Plan
Mark composite revision as default.
Overwrite any existing composites with the same revi
Use the following SOA configuration plan for all composites:
Browse
Help
< Back
Next >
Finish
Cancel
Deployment Action
Deploy Configuration
Select Server
SOA Servers
Summary
Application Servers:
IntegratedWebLogicServer
Choose the target SOA server(s) and corresponding partitions to which you want to deploy this archive.
SOA Server:
Partition:
Status:
Server URL:
DefaultServer
default
RUNNING
http://localhost:7101
[01:15:49 PM] Sending archive - sca_HelloWorld_rev1.0.jar
[01:16:24 PM] Received HTTP response from the server, response code=200
[01:16:24 PM] Successfully deployed archive sca_HelloWorld_rev1.0.jar with 0 warning/severe messages to partition
[01:16:25 PM] Elapsed time for deployment: 2 minutes, 19 seconds
[01:16:25 PM] ---- Deployment finished. ----

Enterprise Manager

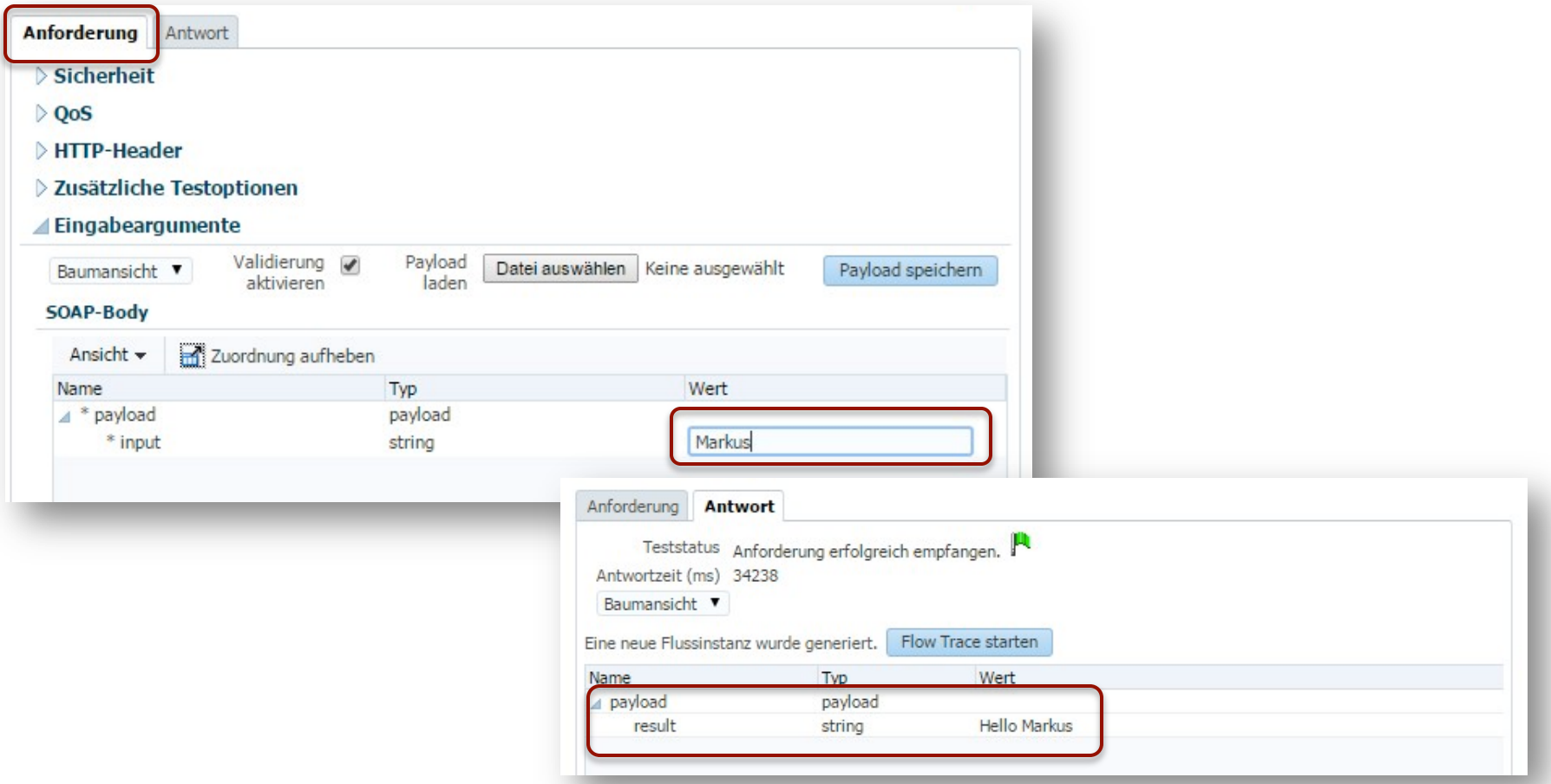
- http://localhost:7101/em
- weblogic / welcome1



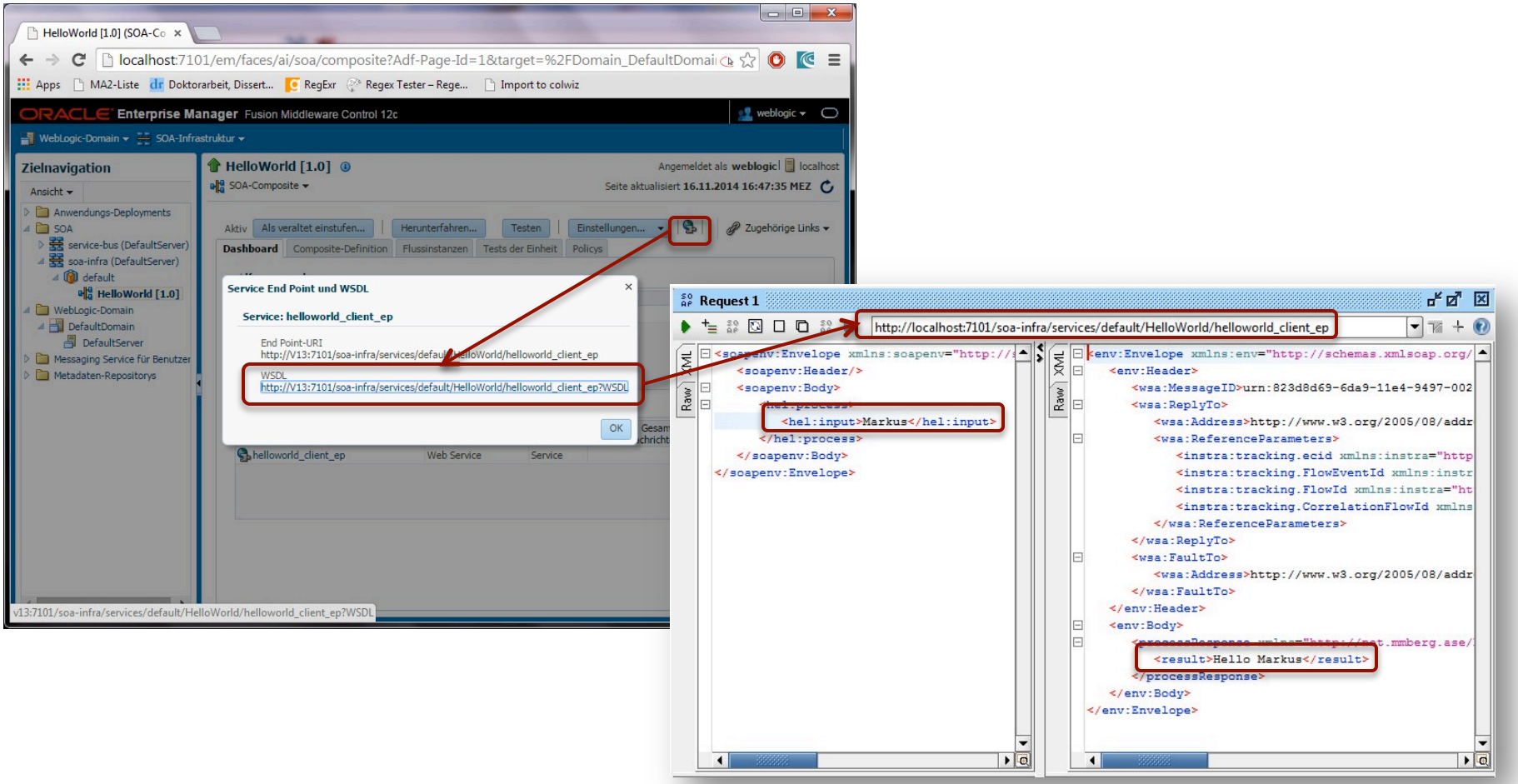
Enterprise Manager (EM)



EM: Testen


Anforderung
Antwort
Sicherheit
QoS
HTTP-Header
Zusätzliche Testoptionen
Eingabeargumente
Baumansicht
Validierung aktivieren
Payload laden
Datei auswählen
Keine ausgewählt
Payload speichern
SOAP-Body
Ansicht
Zuordnung aufheben
Name
Typ
Wert
* payload
payload
* input
string
Markus
Anforderung
Antwort
Teststatus
Anforderung erfolgreich empfangen.
Antwortzeit (ms)
34238
Baumansicht
Eine neue Flussinstanz wurde generiert.
Flow Trace starten
Name
Typ
Wert
payload
payload
result
string
Hello Markus

Testen (Soap UI)



Teil II: BPEL

Motivation

- Dienste liegen bis jetzt eigenständig vor
  - Keine Kommunikation zwischen Diensten
- Ziel der Modularisierung ist jedoch Wiederverwendbarkeit
  - D.h. Kommunikation zwischen Diensten
    - Komposition
- Zentral gesteuerte Komposition: Orchestrierung
  - BPEL dient zur Orchestrierung von Webservices
    - Grundlage einer SOA
      - „Programming in the large“ → Koordination
- BPEL-Prozess selbst ist auch ein Webservice
- Kontext von BPEL sind Geschäftsprozesse, die ausgeführt werden sollen (= Workflows)

BPEL

- „Two-Level-Programming“
  - Services in Programmiersprache entwickeln (z.B. Java oder C#) und bereitstellen (WSDL)
    - „Programming in the small“
  - Auf höherer Abstraktionsebene orchestrieren (BPEL), d.h. deren Interaktion koordinieren
    - „Programming in the large“
- Frage nach der Granularität
  - Was wird in herkömmlicher Programmiersprache geschrieben und was in BPEL?
    - BPEL bietet grundlegende Funktionen wie Schleifen und Fallunterscheidungen (→ Programming in the small möglich)
    - Für kleine Aufgaben lohnt es sich nicht einen Java-WS zu schreiben, aber was ist klein?

BPEL

- Ein Definitionsversuch:

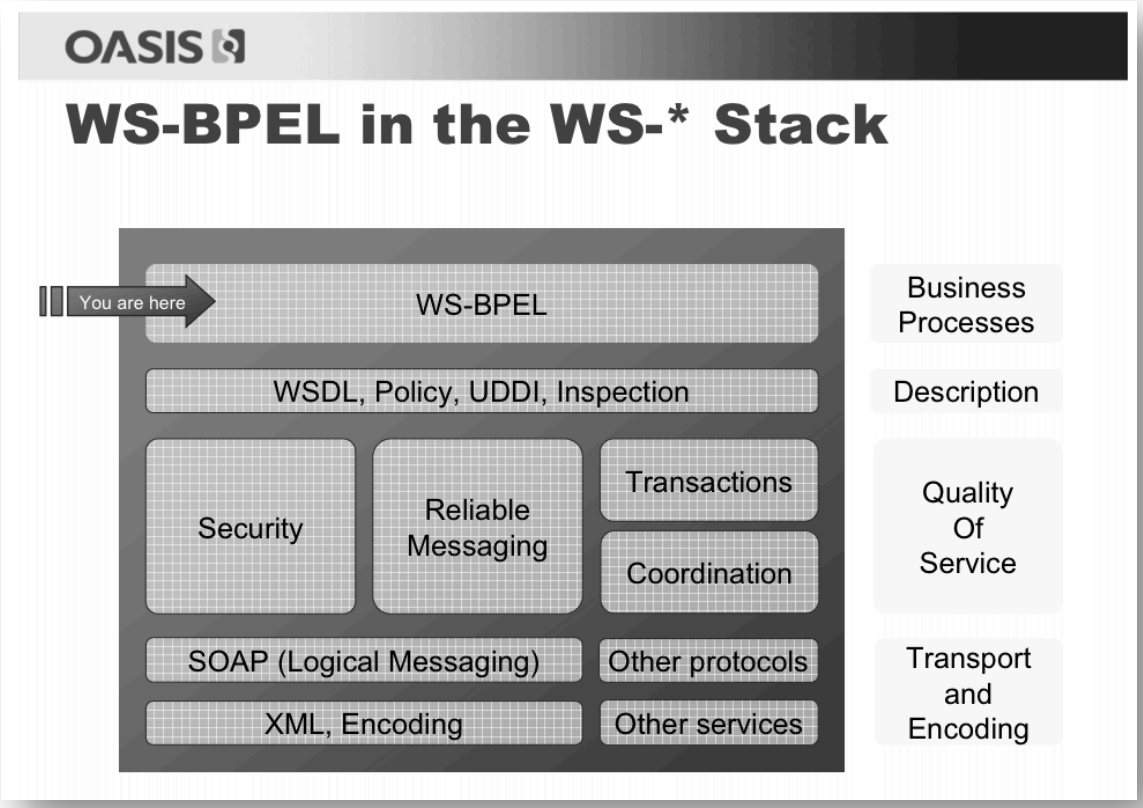
> *„domänenspezifische, imperative Programmiersprache zur Definition von ausführbaren Geschäftsprozessen“* [DPunkt]

- Aber BPEL allein ist nicht genug
  - Um einen Dienst in BPEL aufrufen zu können, muss er zunächst extern existieren (~Kritik an ausführbarem BPMN)
  - Theoretisch kann der gesamte Dienst auch in BPEL geschrieben werden
    - Wieder Frage nach Granularität

Eigenschaften

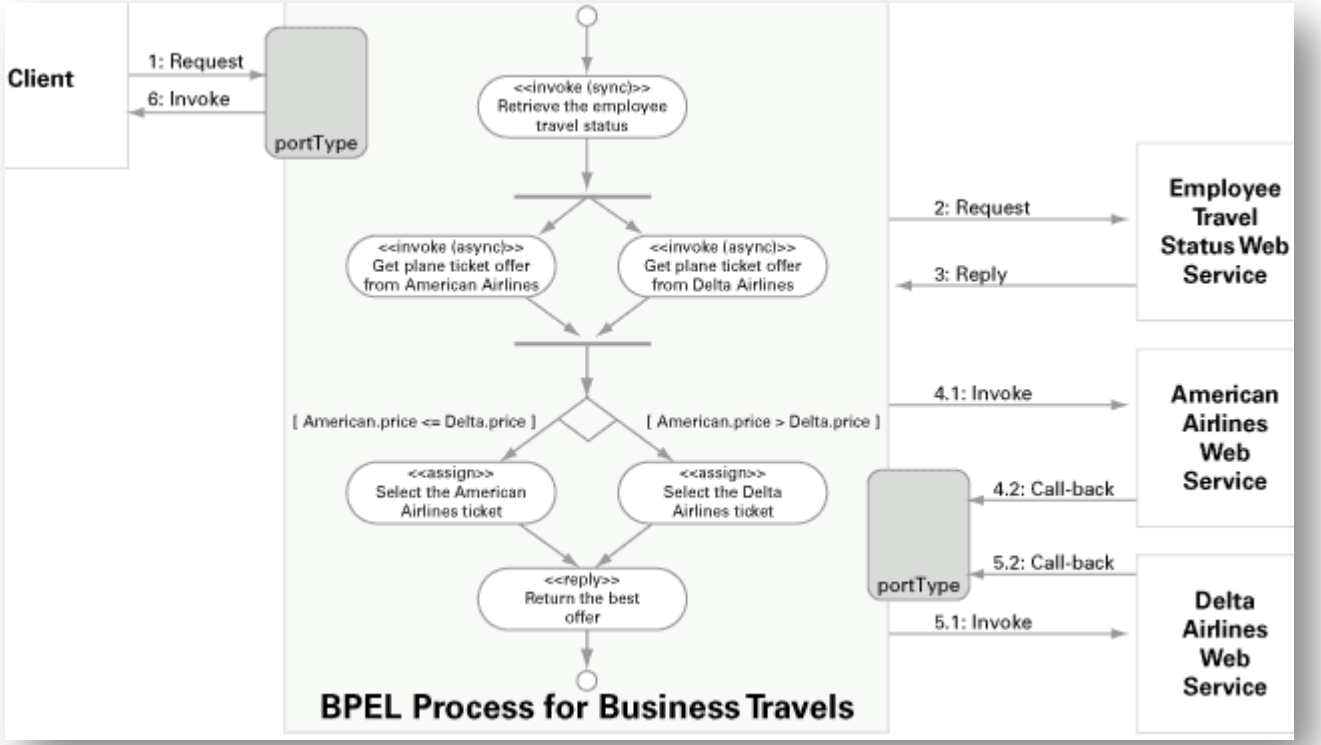
- OASIS-Standard
  - WS-BPEL 2.0
- XML-basiert
- Meist visualisiert, jedoch kein Standard (vgl. BPMN)
  - BPEL wird als XML definiert, kann jedoch der Einfachheit halber grafisch entworfen werden
  - Andere Richtung: BPMN wird grafisch beschrieben und kann als XML gespeichert werden
- Orchestriert SOAP-Webservices

Überblick



https://www.oasis-open.org/committees/download.php/23068/WS-BPEL%20Technical%20Overview%20for%20Developers%20and%20Architects%20-%20Part%201%20(Frank%20Ryan).pdf

Beispiel



http://www.oracle.com/technetwork/articles/matjaz-bpel1-090575.html

BPEL: Grundlegendes

- *Wir beginnen mit ein paar grundlegenden Beschreibungen bevor wir uns mit der BPEL Syntax auseinander setzen*

- *Im Folgenden:*
  - *Partner*
  - *Input/Output-Schema*

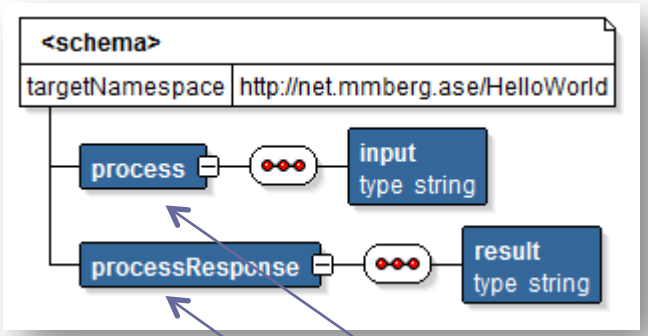
Partner

- Ziel von BPEL ist das Aufrufen von Webservices
- Diese werden als Partner bezeichnet und mit Hilfe eines PartnerLinks adressiert
- Der PartnerLink referenziert die WSDL, die hierfür erweitert wurde
  - Verweis auf PortTypen d.h. Operationen und deren Datentypen

Input-/Output-Schema

- Jeder BPEL-Prozess ist ein WebService und besitzt daher eine WSDL, die automatisch erzeugt wird
  - Jeder Prozess ist eine Operation mit Eingabe- und Ausgabeparametern
  - Die Parameter können auf Grundlage eines bestehenden Schemas erzeugt werden
  - Oder ein neues Schema wird angelegt (mit den Typen „processRequest“ und „processResponse“)
- Jeder Prozess hat somit ein Eingabe- und Ausgabeschema
- Hieraus leiten sich in der WSDL die Messages ab
- Auf Grundlage dieser Typen werden zwei Variablen für jeden Prozess angelegt
  - inputVariable (z.B. vom Typ *processRequest*)
  - outputVariable (z.B. vom Typ *processResponse*)

Input-/Output-Schema



XSD

```
<wsdl:message name="HelloWorldRequestMessage">
           <wsdl:part name="payload" element="client:process"/>
</wsdl:message>
<wsdl:message name="HelloWorldResponseMessage">
           <wsdl:part name="payload" element="client:processResponse"/>
</wsdl:message>
```

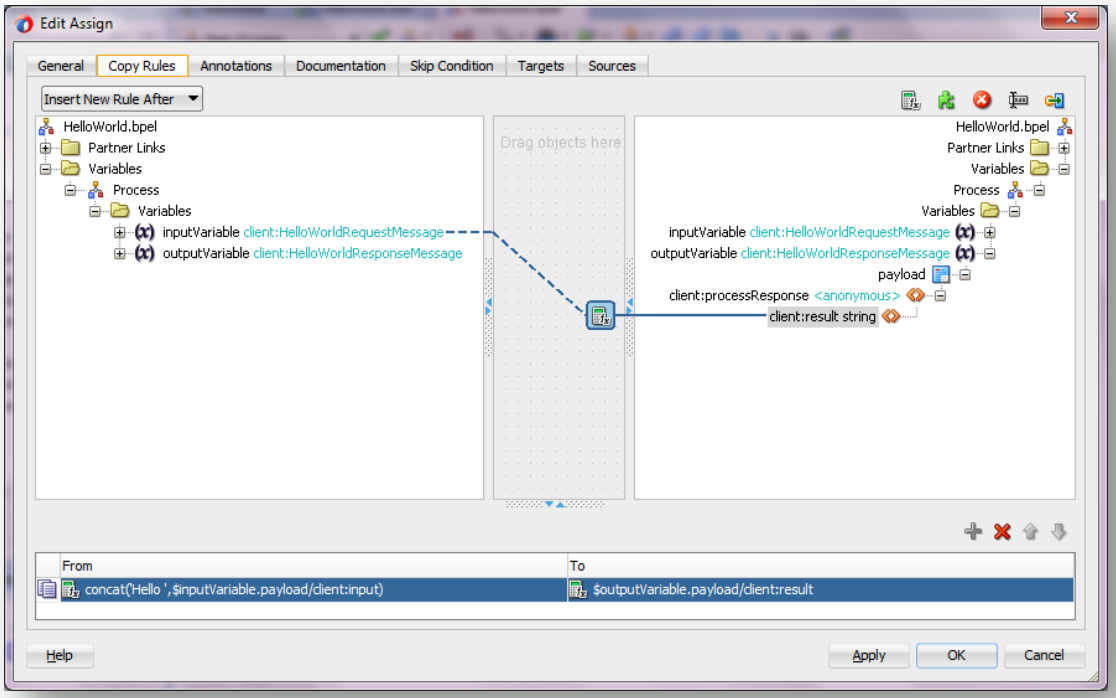
WSDL

```
<variables>
    <!-- Reference to the message passed as input during initiation -->
    <variable name="inputVariable" messageType="client:HelloWorldRequestMessage"/>

    <!-- Reference to the message that will be returned to the requester-->
    <variable name="outputVariable" messageType="client:HelloWorldResponseMessage"/>
  </variables>
```

BPEL

Input-/Output Schema

- Häufiger Anwendungsfall: Mappen der Daten von Input (links) auf Output (rechts)
- In der Mitte: Transformation (XPath-Funktion)



BPEL: Syntax

- Das Root-Element eines BPEL-Prozesses ist `<process>`
- Ein Prozess besteht aus Aktivitäten
  - Diese sind in Sequences oder Scopes zusammengefasst (zur Strukturierung)

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<process name="MeinErsterProzess"
     targetNamespace="http://mmberg.net/ase/meinersterprozess"
     xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
     xmlns:ora="http://schemas.oracle.com/xpath/extension"
     xmlns:ui="http://xmlns.oracle.com/soa/designer"
     xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
     xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable">

<!- Aktivitäten, z.B. innerhalb einer Sequence -->

</process>
```

BPEL: Syntax

- Innerhalb von `<process>` vor Beginn der Aktivitäten (in der Sequenz) werden definiert:
  - Dokumentation (optional)
  - Referenzierte Webservices (PartnerLinks: min. der eigene Service, zusätzlich externe)
  - Variablen (min. Input und Output)
  - FaultHandler (optional)

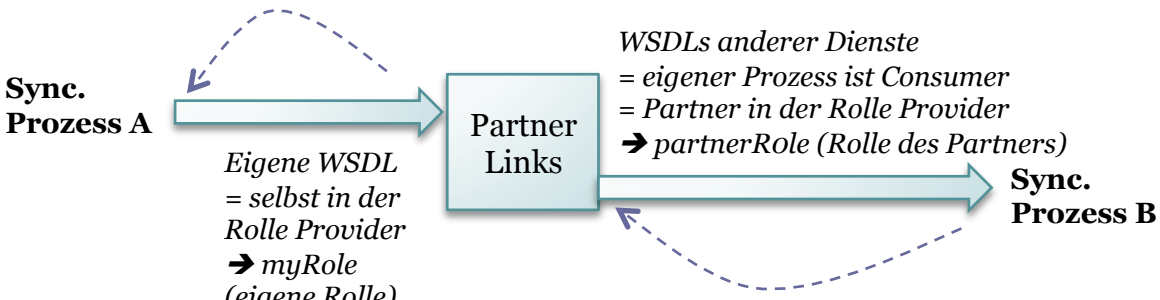
```
<process name="meinErsterProzess" targetNamespace=…>
<partnerLinks>
...
</partnerLinks>
<variables>
...
</variables>
<faultHandlers>
  <catchAll>
  ...
  </catchAll>
</faultHandlers>
<sequence name="main">
...
</sequence>
</process>
```

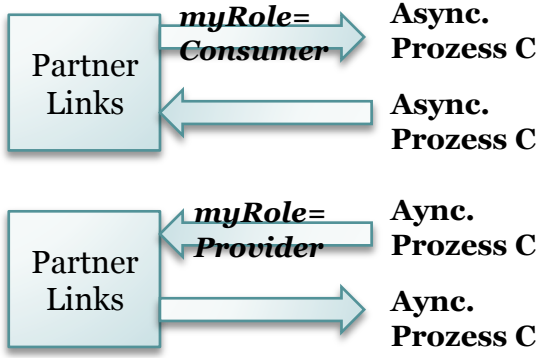
PartnerLinks / PartnerLinkTypes

- PartnerLinks (in BPEL) führen alle bereitgestellten und genutzten Webservices auf
- PartnerLinkTypes werden in WSDL definiert und über PartnerLinks in BPEL referenziert
  - Die WSDL wurde um PartnerLinkType-Informationen angereichert
  - WSDL-PartnerLinkTypes spezifizieren eine Rolle und referenzieren WSDL-PortTypes
- Rollen: Es wird unterschieden zwischen
  - Provider
    - Als anbietender Dienst wird die entsprechende Rolle in eigener WSDL angeboten
  - Consumer
    - Als konsumierender Dienst wird die Provider-Rolle in der WSDL des Partners gesucht
- Partner referenzieren diese Rollen (in BPEL angegeben)
  - myrole (in welcher Rolle befinde ich mich?)
  - partnerRole (in welcher Rolle befindet sich der Partner?)

PartnerLinks

- PartnerLinks erzeugen eine Beziehung zwischen den PortTypes zweier WS
  - Rolle entscheidet darüber welcher PortType der WSDL genutzt wird (Provider oder Consumer) → Wer ist wer?
  - Bei asynchronen Services hat die WSDL zwei PortTypes (bei synchronen nur eine, da die Beziehung klar ist → request/response, d.h. Aufruf nur in eine Richtung)
    - Somit sind Rollen vor allem bei asynchronen Prozessen wichtig
    - Allerdings sind sie auch bei synchronen Prozessen verpflichtend (dort muss allerdings nur eine der beiden Rollen angegeben werden)





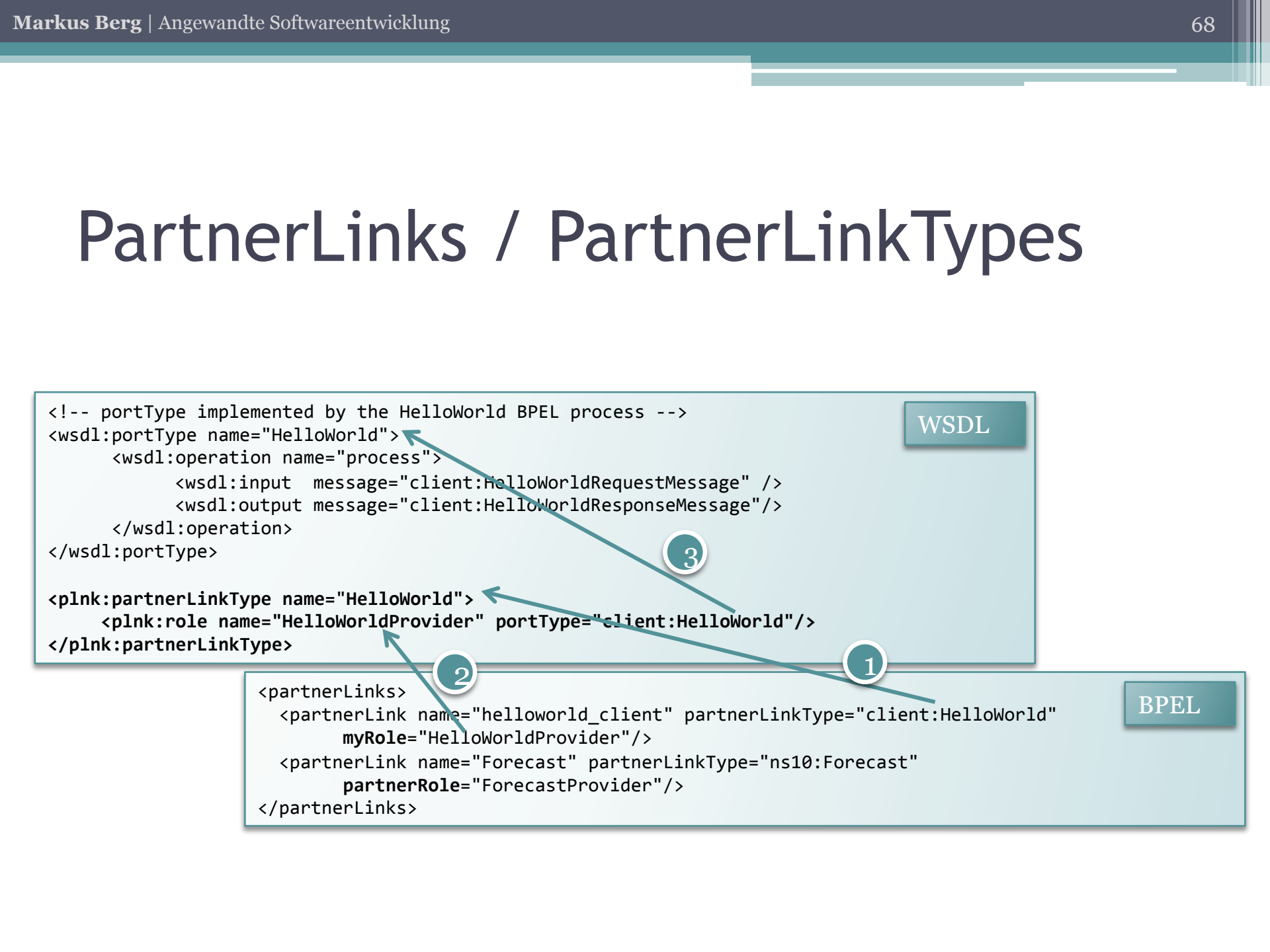
PartnerLinks / PartnerLinkTypes

WSDL

```
<!-- portType implemented by the HelloWorld BPEL process -->
<wsdl:portType name="HelloWorld">
      <wsdl:operation name="process">
            <wsdl:input  message="client:HelloWorldRequestMessage" />
            <wsdl:output message="client:HelloWorldResponseMessage"/>
      </wsdl:operation>
</wsdl:portType>

<plnk:partnerLinkType name="HelloWorld">
     <plnk:role name="HelloWorldProvider" portType="client:HelloWorld"/>
</plnk:partnerLinkType>
```

BPEL

```
<partnerLinks>
  <partnerLink name="helloworld_client" partnerLinkType="client:HelloWorld"
        myRole="HelloWorldProvider"/>
  <partnerLink name="Forecast" partnerLinkType="ns10:Forecast"
        partnerRole="ForecastProvider"/>
</partnerLinks>
```



Variablen

- Drei Typen
  - Message (aus WSDL)
  - Element (aus XSD)
  - SimpleType (z.B. xsd:String)

```
<variables>
  <variable name="temp" messageType="ns3:requestTemperatureMsg"/>
  <variable name="person" element="client:Person"/>
  <variable name="counter" type="xsd:int”/>
</variables>
```

- Können direkt instantiiert werden

```
<variable name="counter" type="xsd:int">
   <from><literal>0</literal></from>
</variable>
```

FaultHandler

- Reaktion auf Fehler eines bestimmten Typs bzw. auf alle Fehler (catchAll)

```
<faultHandlers>
  <catchAll>
    <!-- Definition was im Fehlerfall zu tun ist -->
   </catchAll>
</faultHandlers>
```

Aktivitäten

- <receive>
- <reply>
- <invoke>
- <assign>
- <throw>
- <exit>
- <wait>
- <empty>
- <sequence>
- <if>
- <while>
- <repeatUntil>
- <forEach>
- <pick>
- <flow>
- <scope>
- <compensate>
- <compensateScope>
- <rethrow>
- <validate>
- <extensionActivity>

Sequence, Scope & Flow

- Sequence: Zur Strukturierung
  - Führt enthaltene Aktivitäten genau in der angegebenen Reihenfolge aus

```
<sequence name="Sequence1">
…
</sequence>
```

- Scope dient ebenfalls der Strukturierung, darüber hinaus verfügt er jedoch u.a. über:
  - Lokale Variablen
    - Überdecken globale Variablen
  - Lokales Fehlerhandling
- D.h. definiert neuen Gültigkeitsbereich (im Gegensatz zum Standardgültigkeitsbereich von `<process>`)

```
<scope name="Scope1">
…
</scope>
```

```
<process>
<!-- varA=3 -->
      <scope name="Scope1">
      <!-- varA=4 -->
      </scope>
<!– getVarA -> 3 -->
</process>
```

- Flow: parallele Ausführung von Aktivitäten

Receive

- Warten auf das Eintreffen einer Nachricht (blockierend)
  a) Markiert somit u.a. den Einstiegspunkt der Anwendung
  - Prozessinitiierung (**createInstance**=„yes“)
    - „soll eine neue Instanz erzeugt werden, wenn ich eine Nachricht empfange?“

  b) Wartet auf Antwort (Callback) eines asynchron aufgerufenen Dienstes
- Spezifizieren des Aufrufs:
  - **PartnerLink**: Wer ruft mich auf?
  - **PortType**: Welchen Porttyp bietet der eigene Dienst an, d.h. auf welchem Porttyp wird die Nachricht empfangen?
  - **Operation**: Auf welche Operation reagiere ich, d.h. mit welcher Operation wird der Dienst aufgerufen?
- **Variable**: Definiert den Namen der für den Inhalt des Requests zu verwendenden Variable

```
<receive name="receiveInput" partnerLink="wetter_client" portType="client:Wetter"
operation="process" variable="inputVariable" createInstance="yes"/>
```

```
<receive name="receiveInput" partnerLink="helloworld_client"
portType="client:HelloWorld" operation="process" variable="inputVariable"
createInstance="yes"/>
```

Reply

- Synchrone Antwort
  - Nachricht senden als Antwort auf ein Receive
  - Vollendet damit die Request-Response-Sequenz
- (asynchrone Antwort wird per Invoke realisiert)

```
<reply name="replyOutput" partnerLink="Wetter_client" portType="client:Wetter"
operation="process" variable="outputVariable"/>
```

```
<reply name="replyOutput" partnerLink="helloworld_client"
portType="client:HelloWorld" operation="process" variable="outputVariable"/>
```

Assign

- Lesen/Schreiben von Variablen
  - D.h. updaten der Werte von Variablen
- Beliebig viele Zuweisungen verschiedener Variablen
  - als *copy…from…to* definiert
- InsertMissingTo: Anlegen von (Zwischen)Elementen falls sie nicht existieren

```
<assign name="incFailedCounter">
<copy>
 <from>$errCount+1</from>
 <to>$errCount</to>
</copy>
```

```
<assign name="assignMAname">
<copy>
 <from>$invokeGetMitarbeiter_OutputVariable/ns1:ma[1]/ns1:nachname</from>   <to>
 $outputVariable.payload/ns0:person/ns0:name</to>
</copy>
```

- XPath:

```
<assign name="assignGreetings">
      <copy>
        <from>concat('Hello ',$inputVariable.payload/client:input)</from>
        <to>$outputVariable.payload/client:result</to>
      </copy>
</assign>
```

Assign: XSLT

- XSL Transformation wird über ein *assign* aufgerufen
  - `doXSLTransformForDoc()`
    - XSL Datei
    - Input-Variable(n)

```
<assign name="prepareMessage">
    <bpelx:annotation>
       <bpelx:pattern patternName="bpelx:transformation"></bpelx:pattern>
    </bpelx:annotation>
    <copy>
     <from>ora:doXSLTransformForDoc("../Transformations/AlwinProExporter.prepareMessage.xsl",
$outputVariable.payload)</from>
    <to variable="invoke_sendMessage_InputVariable" part="payload"/>
    </copy>
</assign>
```

Invoke

- Aufrufen eines als Partner definierten Webservices
- Arten:
  - Synchron (request/response)
  - Ohne Antwort (fire-and-forget)
  - Asynchron (one-way mit Callback)
- Input- und Output-Variable
  - Input-Variable vom Typ des Input-Schemas des aufzurufenden Webservices
  - Output-Variable analog vom Typ des Output-Schemas
    - Nur bei synchronen Aufrufen (sonst keine Outputvariable)

```
<invoke name="invoke_getNotExportedChanges" bpelx:invokeAsDetail="no"
partnerLink="Wetter"
portType="ns5:WetterPort"
operation"getWeather"
inputVariable="invoke_getWeather_InputVariable”
outputVariable="invoke_getWeather_OutputVariable"/>
```

Fallunterscheidung: If/else

- Ausführen einer von mehreren Möglichkeiten je nach Eintreten der angegebenen Bedingung(en)

```
<if name="ifExistsPerson">
 <condition>ns0:PersonCollection/ns0:Person</condition>
 <sequence>
  ...
 </sequence>
 <elseif>
 <sequence>
  ...
 </sequence>
 </elseif>
</if>
```

Schleifen: While & RepeatUntil

- While:
  - Wiederholung der angegebenen Aktivitäten solange Bedingung wahr ist

```
<while condition="bool-expr">
   <!-- Aktivitäten -->
</while>
```

- RepeatUntil
  - Wiederholung der angegebenen Aktivitäten bis die angegebene Bedingung wahr ist
  - Wird mindestens einmal ausgeführt

Schleifen: For-each

- Besitzt einen Startwert und einen Endwert
- Führt die Aktivitäten N+1 mal aus
  - N=Endwert-Startwert
  - z.B.
    - 1..3 → (3-1)+1 = 3
    - 2..5 → (5-2)+1 = 4
- Nicht vergleichbar mit einem for-each (Iterator) in Java
  - Eher eine gewöhnliche for-Schleife
    - for(i=1; i<=3;i++)
    - for(i=2; i<=5;i++)

Fehlerbehandlung

- Throw
  - „Werfen“ eines Fehlers aus dem Prozess heraus
  - Besteht aus Name und Fehlervariable
- Fehler werden vom FaultHandler gefangen
  - Im Handler wird entsprechend auf den Fehler reagiert
  - z.B. mit Fehlernachricht antworten

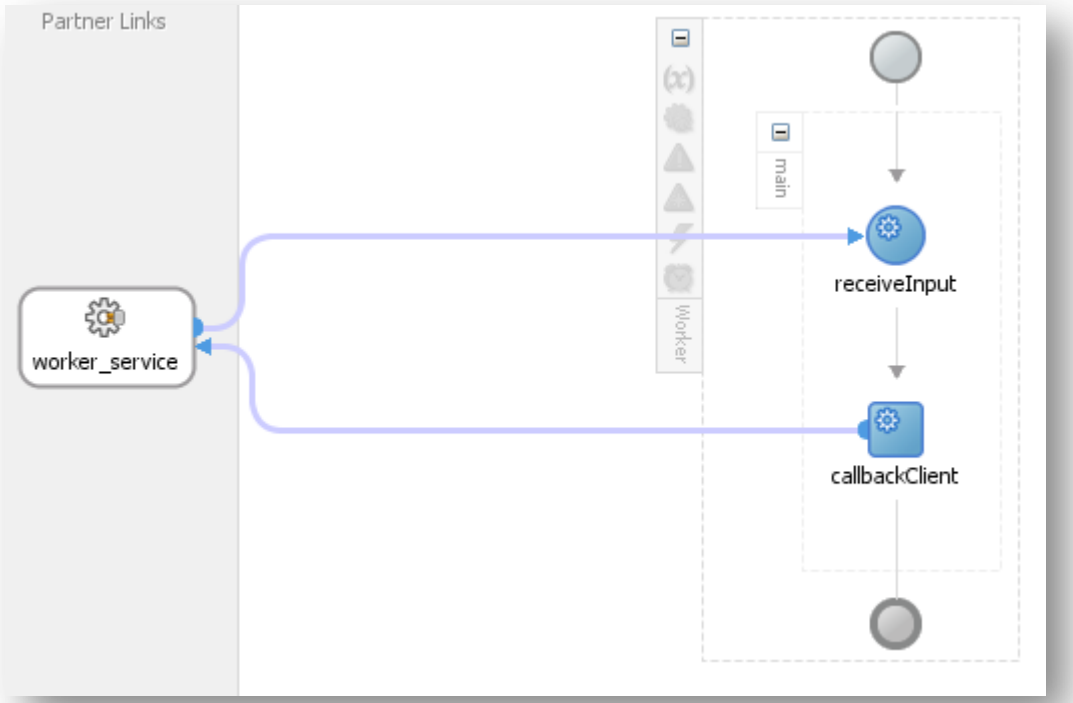
Compensate

- Geschäftsprozess besteht aus mehreren eigenständigen Aktivitäten
  - z.B. aufrufen externer Dienste
- Bei einem Fehler müssen eventuell bereits abgeschlossene Aktivitäten rückgängig gemacht werden
  - Reise buchen:
    - Flug buchen: ok
    - Hotel buchen: Fehler
      - Compensate: Flug stornieren
- Definieren eines **CompensationHandlers**
  - Definiert wie der Vorgang rückgängig gemacht werden kann
    - z.B. Flug stornieren oder Kreditkarte mit entgegengesetztem Wert belasten
  - Auf Scope-Ebene
  - **Compensate** auslösen im FaultHandler

PartnerLinks

- Woher kommt die Rolle bei einem externen Service?
  - JDeveloper legt eine lokale Wrapper-WSDL an, wo der PartnerLink definiert wird
  - Dieser Wrapper importiert die „echte“ (externe) WSDL des referenzierten Services
  - Auf diese Weise wird die ursprüngliche WSDL um PartnerLink-Informationen erweitert

PartnerLinks: Asynchroner Service



WSDL (Asynchroner Service)

- Zwei PortTypes (one-way)

```
<wsdl:portType name="Worker">
  <wsdl:operation name="process">
    <wsdl:input message="client:WorkerRequestMessage"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="WorkerCallback">
  <wsdl:operation name="processResponse">
    <wsdl:input message="client:WorkerResponseMessage"/>
  </wsdl:operation>
</wsdl:portType>
```

- Service (Provider) nutzt PortType „Worker“
- Consumer muss ebenfalls einen Service anbieten, den der Provider aufrufen kann, um die Antwort zu übermitteln
  - Hierzu wird der PortType „WorkerCallback“ genutzt
  - Beide PortTypes stehen in gleicher WSDL

Partner Links (Asynchroner Service)

- WSDL: Zwei Rollen definiert, die auf die beiden PortTypes verweisen

```
<plnk:partnerLinkType name="Worker">
  <plnk:role name="WorkerRequester" portType="client:WorkerCallback"/>
  <plnk:role name="WorkerProvider" portType="client:Worker"/> </
plnk:partnerLinkType>
```

- BPEL: Beide Rollen referenziert

```
<partnerLink name="worker_service" partnerLinkType="client:Worker"
myRole="WorkerProvider" partnerRole="WorkerRequester"/>
```

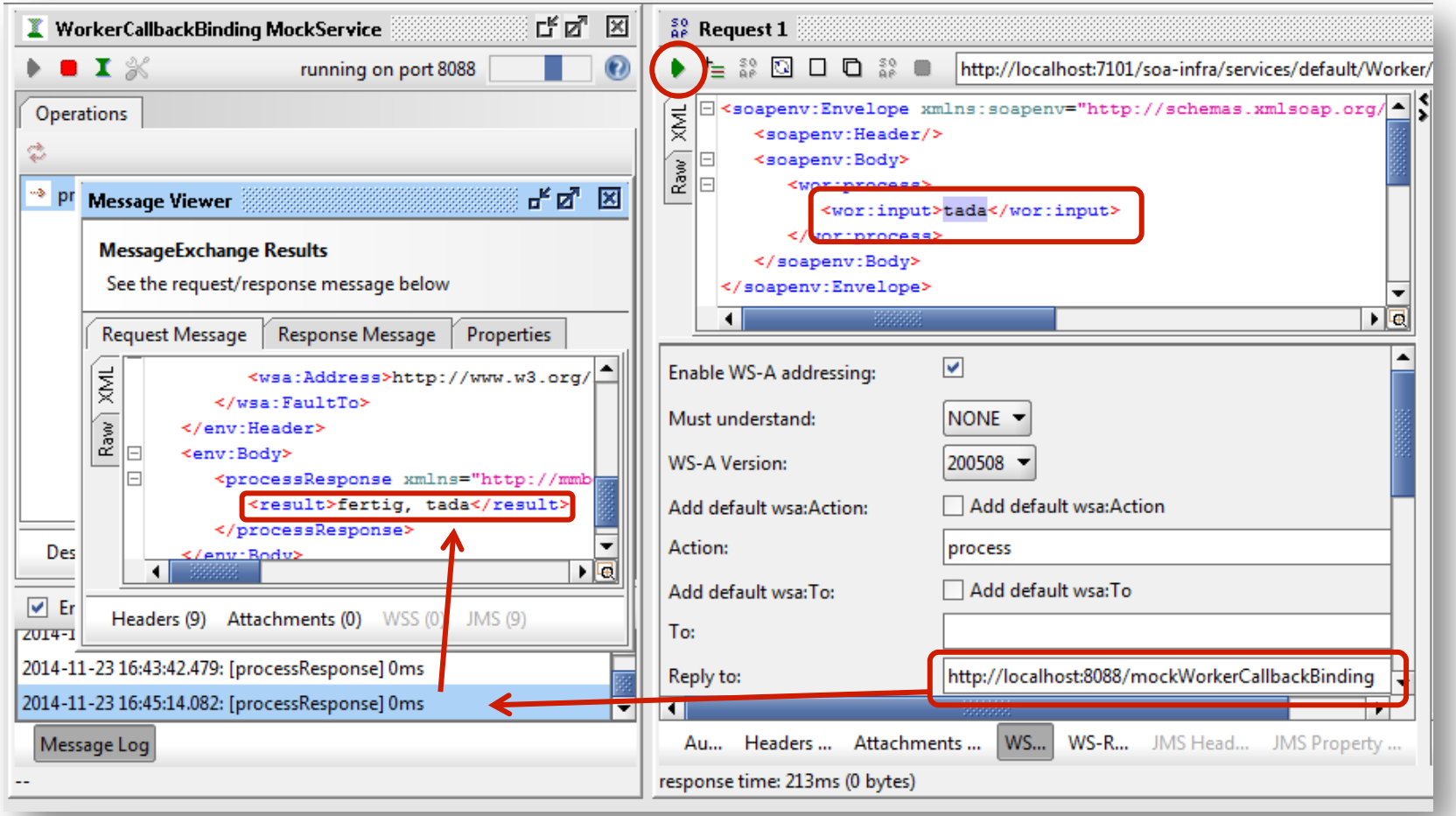
  - Receive:

```
<receive name="receiveInput" partnerLink="worker_service"
portType="client:Worker" operation="process" variable="inputVariable"
createInstance="yes"/>
```

  - Invoke (Aufrufen des Callbacks):

```
<invoke name="callbackClient" partnerLink="worker_service"
portType="client:WorkerCallback" operation="processResponse"
inputVariable="outputVariable"/>
```

Asynchrone Antwort: SOAP UI

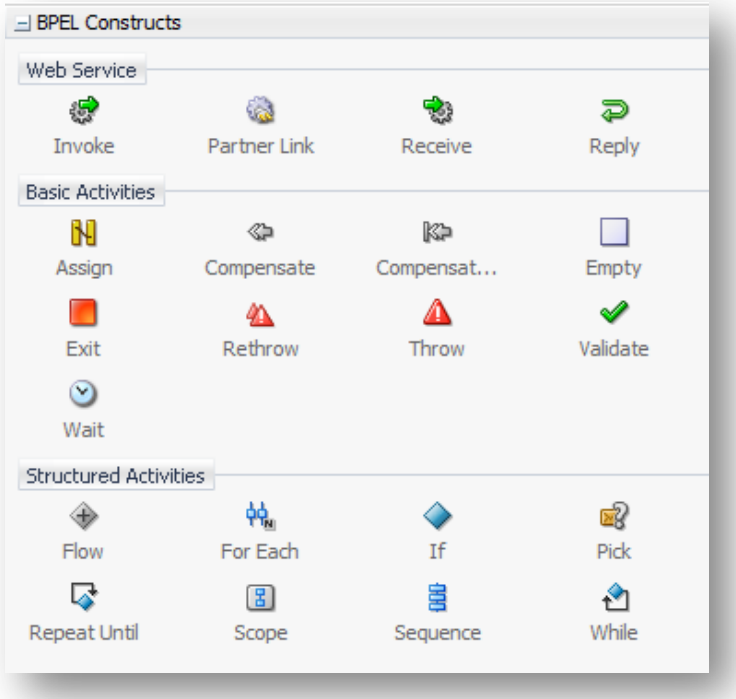


Synchroner oder Asynchroner BPEL Prozess?

- Abhängig von der Laufzeit des Dienstes
- Kurze Laufzeit → synchron
- Lange Laufzeit → asynchron

Grafische Modellierung

- Oracle SOA Suite



→ Mehr in der Übung

Quellen und weiterführende Literatur

- https://dpunkt.de/leseproben/3340/4_Business%20Process%20Execution%20Language%20(BPEL).pdf
- https://www.packtpub.com/sites/default/files/downloads/7948_AppendixA.pdf
- http://www.radikalfx.com/bpel/language.html
- http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
- http://www.cs.hs-rm.de/~linn/fachsem0910/bernti/b.pdf
- https://blogs.oracle.com/gopalan/entry/bpel_what_are_partnerlinktypes_roles
- http://www.oracle.com/technetwork/articles/matjaz-bpel1-090575.html
- http://www.oracle.com/technetwork/articles/matjaz-bpel2-082861.html

Übungen

BPEL

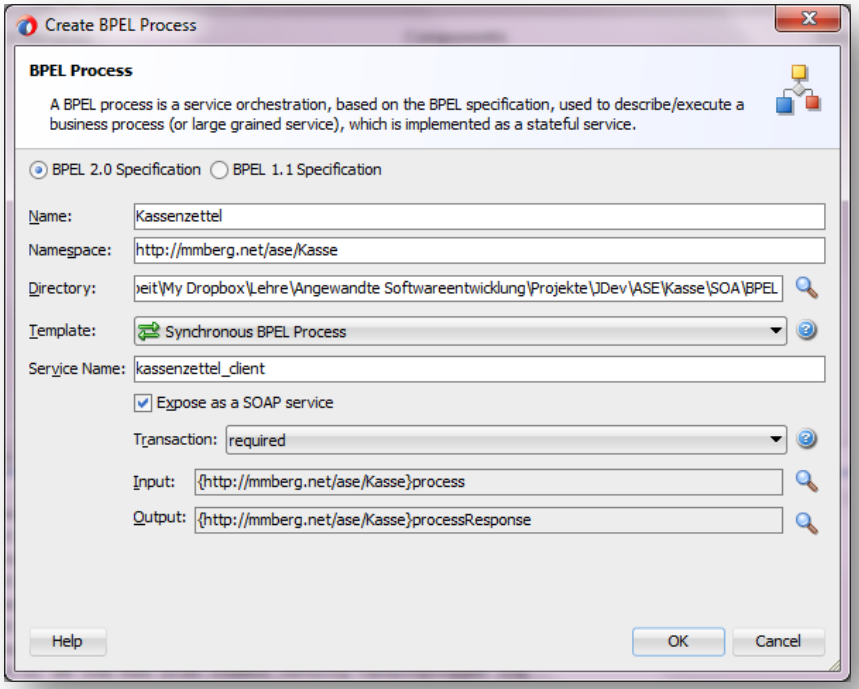
Warm Up!

- Hello World von letzter Woche gemeinsam zum Laufen bringen…
- Fragen?

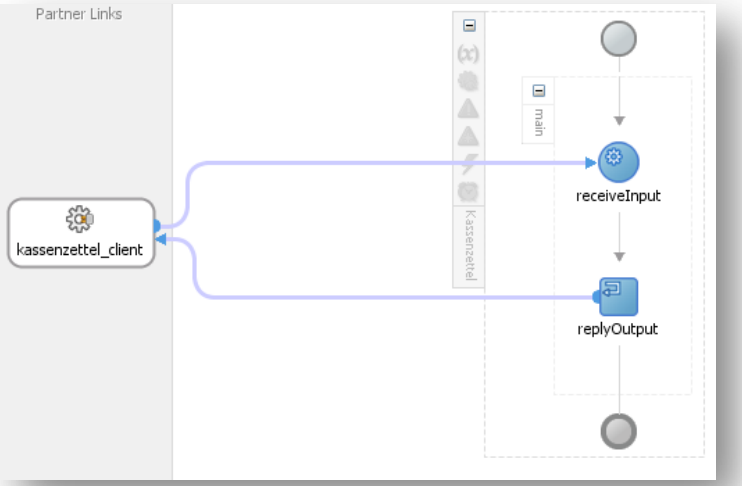
Übung: Kassenzettel

- Teil I
  - Es gibt verschiedene Produkte mit unterschiedlicher MwSt
    - 7% (Milchprodukte, Lebensmittellieferung, Bücher,…)
    - 19%
  - Eingabe: Liste von Produkten mit Nettopreis und Typ
  - Ausgabe: „Kassenzettel“ mit Bruttopreisen
- Teil II
  - Nutzung des Converter Service und umrechnen der Bruttopreise in andere Währung

Neue Applikation bzw. Projekt

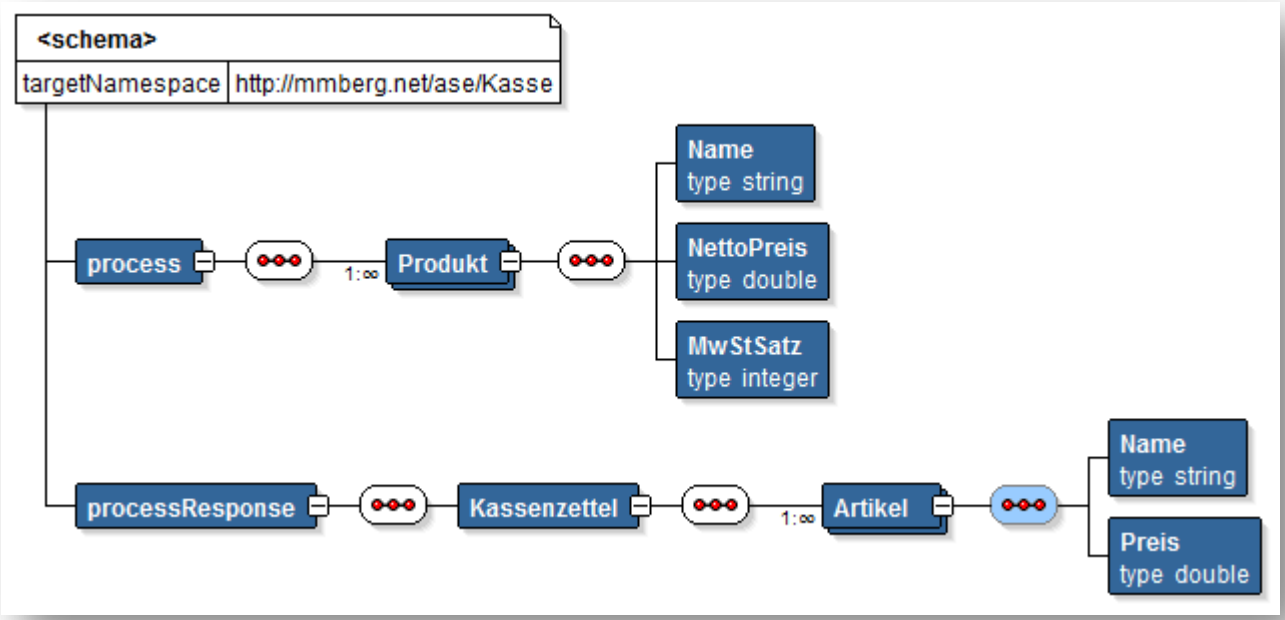


- Neues Projekt „Kasse“
- SCA wird angezeigt
- Neuer BPEL Prozess



Schema bearbeiten

- (Alternative: Schema zuerst erstellen)



WSDL wurde erzeugt

```
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MESSAGE TYPE DEFINITION - Definition of the message types used as
part of the port type defintions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<wsdl:message name="KassenzettelRequestMessage">
        <wsdl:part name="payload" element="client:process"/>
</wsdl:message>
<wsdl:message name="KassenzettelResponseMessage">
        <wsdl:part name="payload" element="client:processResponse"/>
</wsdl:message>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PORT TYPE DEFINITION - A port type groups a set of operations into
a logical service unit.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->

<!-- portType implemented by the Kassenzettel BPEL process -->
<wsdl:portType name="Kassenzettel">
        <wsdl:operation name="process">
                <wsdl:input  message="client:KassenzettelRequestMessage" />
                <wsdl:output message="client:KassenzettelResponseMessage"/>
        </wsdl:operation>
</wsdl:portType>
```

```
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PARTNER LINK TYPE DEFINITION
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<plnk:partnerLinkType name="Kassenzettel">
        <plnk:role name="KassenzettelProvider" portType="client:Kassenzettel"/>
</plnk:partnerLinkType>
```

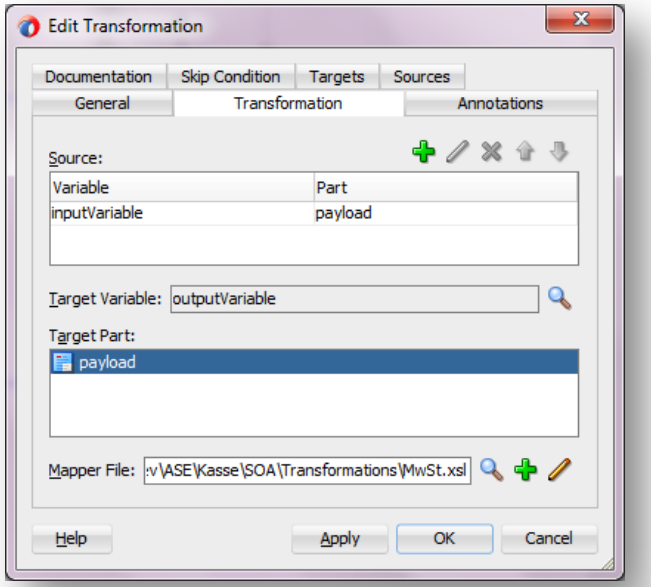
Berechnen des Bruttopreises

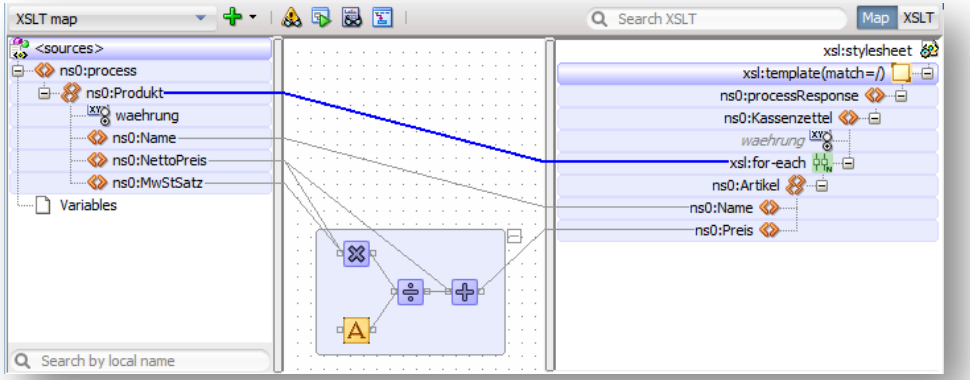
- Rechnen: mit XPath möglich

```
K:NettoPreis + (k:NettoPreis*k:MwStSatz div 100)
```

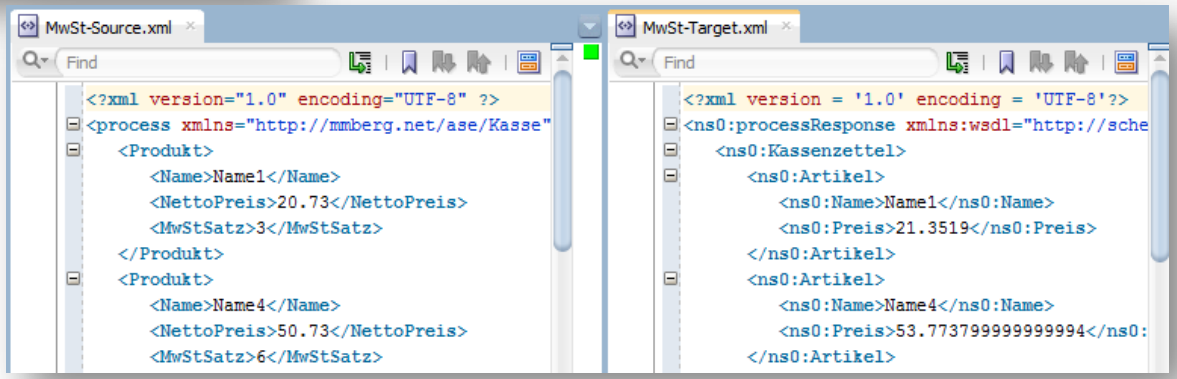
- Varianten: die Preise aller Produkte müssen berechnet werden
  a) Schleife und Assign
  b) XSLT

XSLT



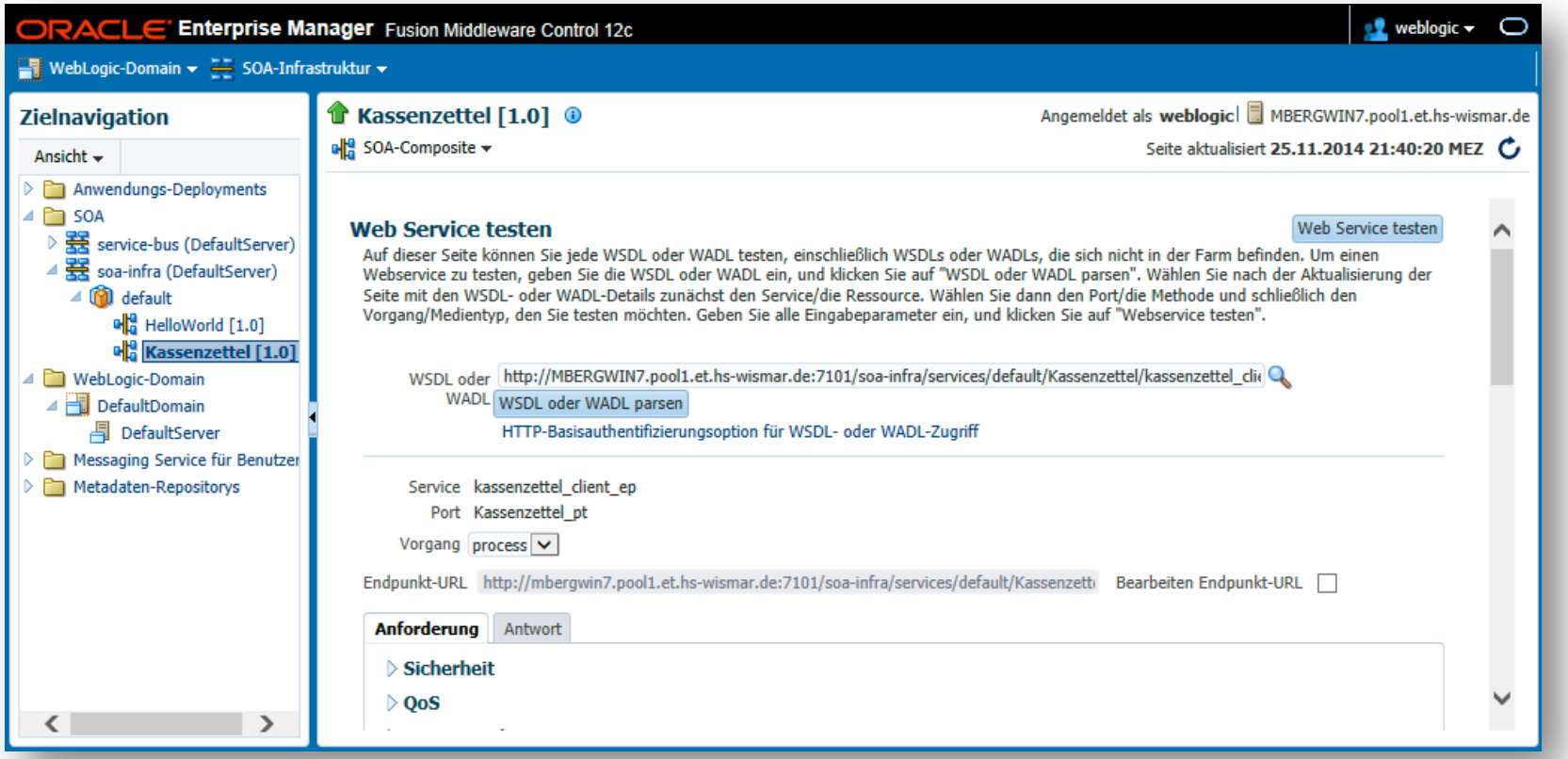



```
<xsl:template match="/">
  <ns0:processResponse>
    <ns0:Kassenzettel>
      <xsl:for-each select="/ns0:process/ns0:Produkt">
        <ns0:Artikel>
          <ns0:Name>
            <xsl:value-of select="ns0:Name"/>
          </ns0:Name>
          <ns0:Preis>
            <xsl:value-of select="ns0:NettoPreis + ((ns0:NettoPreis * ns0:MwStSatz) div 100)"/>
          </ns0:Preis>
        </ns0:Artikel>
      </xsl:for-each>
    </ns0:Kassenzettel>
  </ns0:processResponse>
</xsl:template>
```



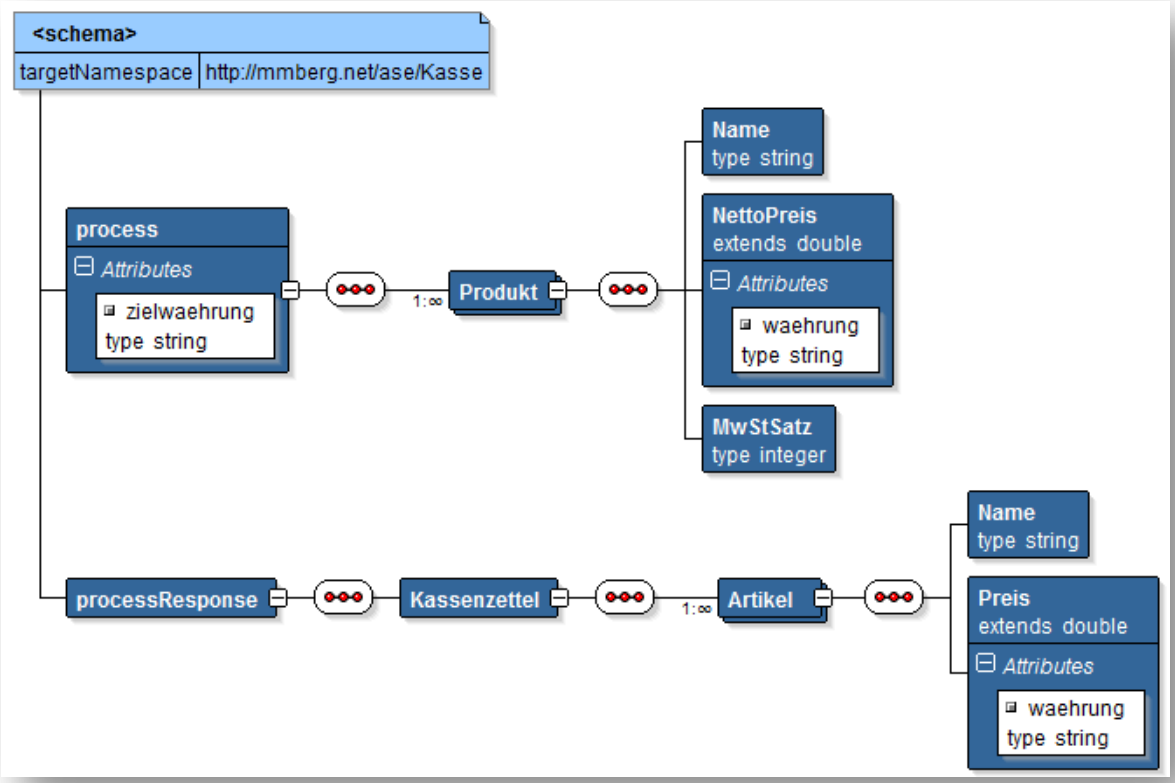
Testen mit Enterprise Manager oder Soap UI

- EM: http://localhost:7101/em



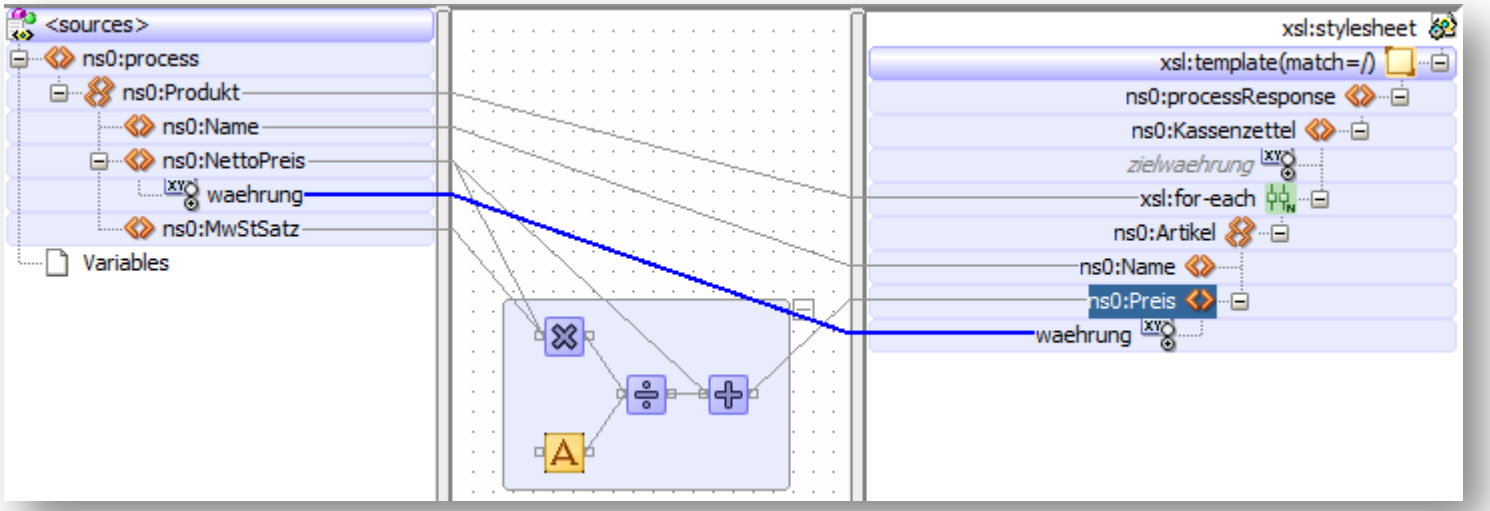
Schema anpassen (für Teil II)

- Währung



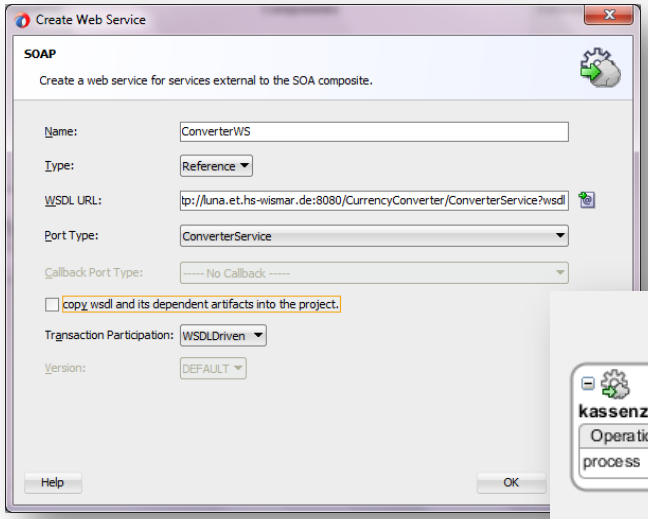
XSLT anpassen

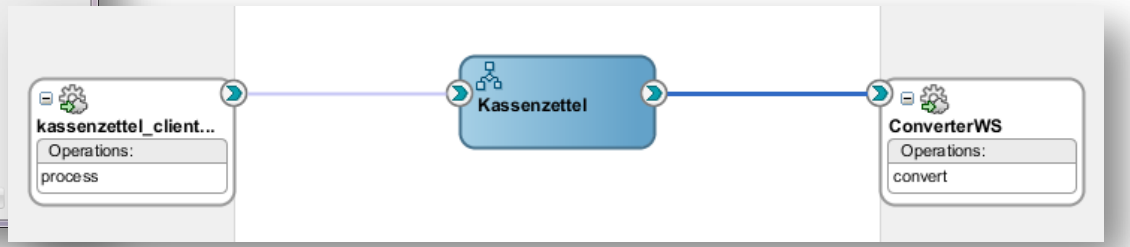
- Währung übernehmen



Umrechnen

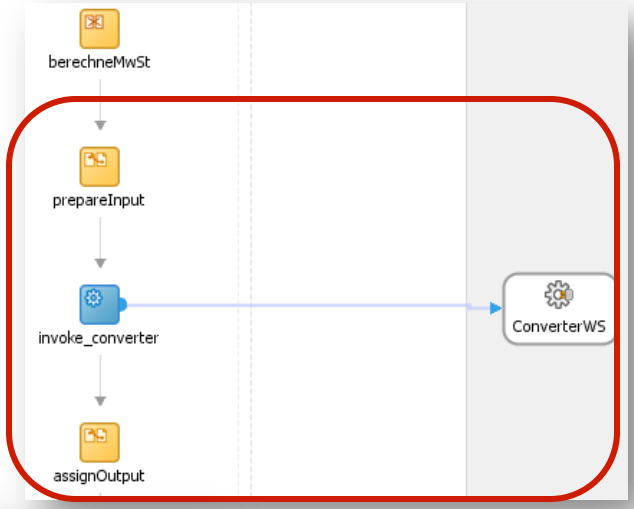
- Nutzen unseres CurrencyConverter-Beispiels
  - WSDL:
    - http://luna.et.hs-wismar.de:8080/CurrencyConverter/ConverterService?wsdl
  - Hinzufügen einer neuen SOAP-Referenz im SCA
    - Und „verkabeln“....



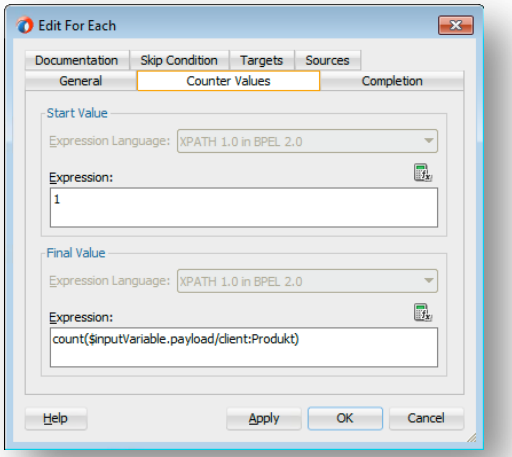


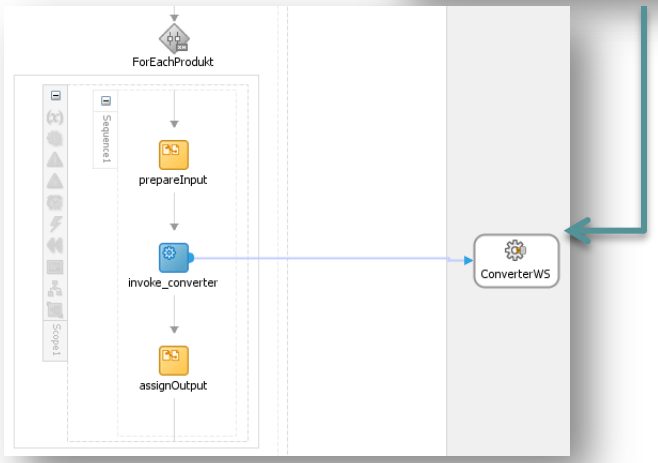
Umrechnung durchführen

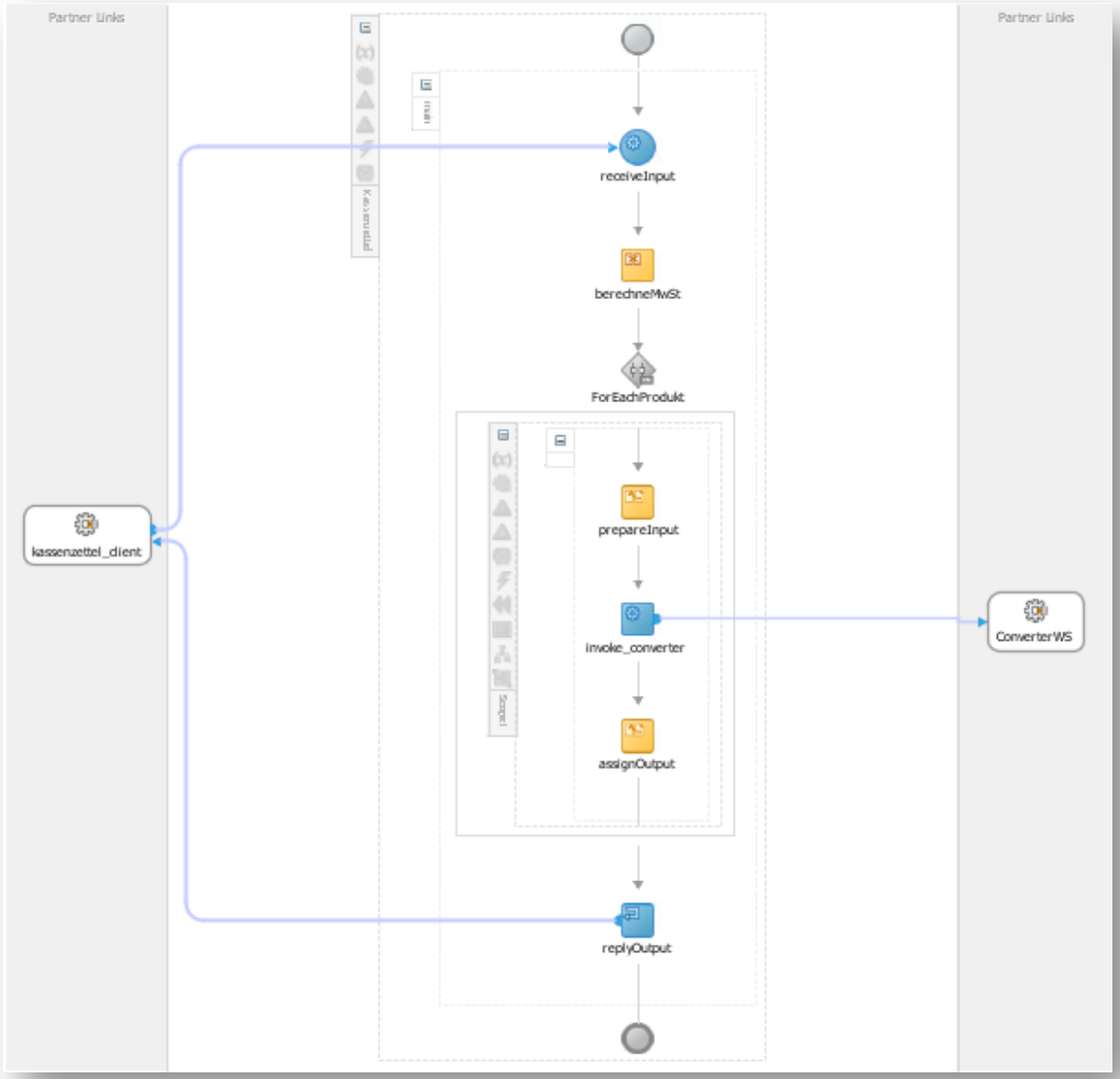
- Invoke
  - Assign der Input und Output Variablen



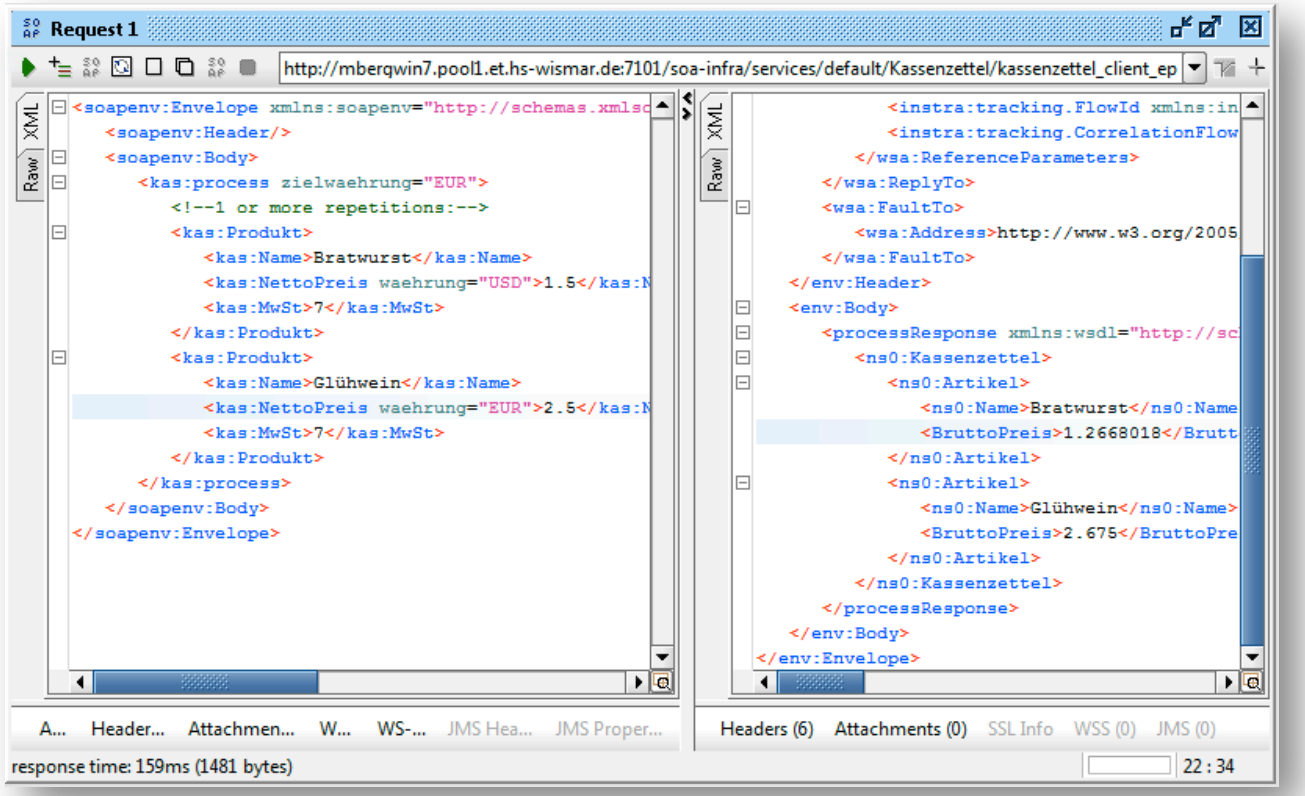
- Schleife (for-each):






Partner Links
Partner Links
main
receiveInput
berechneMwSt
ForEachProdukt
prepareInput
invoke_converter
assignOutput
replyOutput
kassenzettel_client
ConverterWS

Test: Soap UI


Request 1
http://mberqwin7.pool1.et.hs-wismar.de:7101/soa-infra/services/default/Kassenzettel/kassenzettel_client_ep
XML
Raw
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsc
<soapenv:Header/>
<soapenv:Body>
<kas:process zielwaehrung="EUR">
<!--1 or more repetitions:-->
<kas:Produkt>
<kas:Name>Bratwurst</kas:Name>
<kas:NettoPreis waehrung="USD">1.5</kas:N
<kas:MwSt>7</kas:MwSt>
</kas:Produkt>
<kas:Produkt>
<kas:Name>Glühwein</kas:Name>
<kas:NettoPreis waehrung="EUR">2.5</kas:N
<kas:MwSt>7</kas:MwSt>
</kas:Produkt>
</kas:process>
</soapenv:Body>
</soapenv:Envelope>
<instra:tracking.FlowId xmlns:in
<instra:tracking.CorrelationFlow
</wsa:ReferenceParameters>
</wsa:ReplyTo>
<wsa:FaultTo>
<wsa:Address>http://www.w3.org/2005
</wsa:FaultTo>
</env:Header>
<env:Body>
<processResponse xmlns:wsdl="http://sc
<ns0:Kassenzettel>
<ns0:Artikel>
<ns0:Name>Bratwurst</ns0:Name
<BruttoPreis>1.2668018</Brutt
</ns0:Artikel>
<ns0:Artikel>
<ns0:Name>Glühwein</ns0:Name>
<BruttoPreis>2.675</BruttoPre
</ns0:Artikel>
</ns0:Kassenzettel>
</processResponse>
</env:Body>
</env:Envelope>
A... Header... Attachmen... W... WS-... JMS Hea... JMS Proper...
Headers (6) Attachments (0) SSL Info WSS (0) JMS (0)
response time: 159ms (1481 bytes)
22 : 34

Teil III: Integration

Oracle SOA Suite

Motivation

- Letzte Übung: Aufrufen von SOAP-Webservices
  → Orchestrierung

- Einbinden von anderen Schnittstellen, z.B. Datenbanken
  - Kommunikation zwischen heterogenen Programmen
  - Daten- und Protokolltransformation

  → Integration

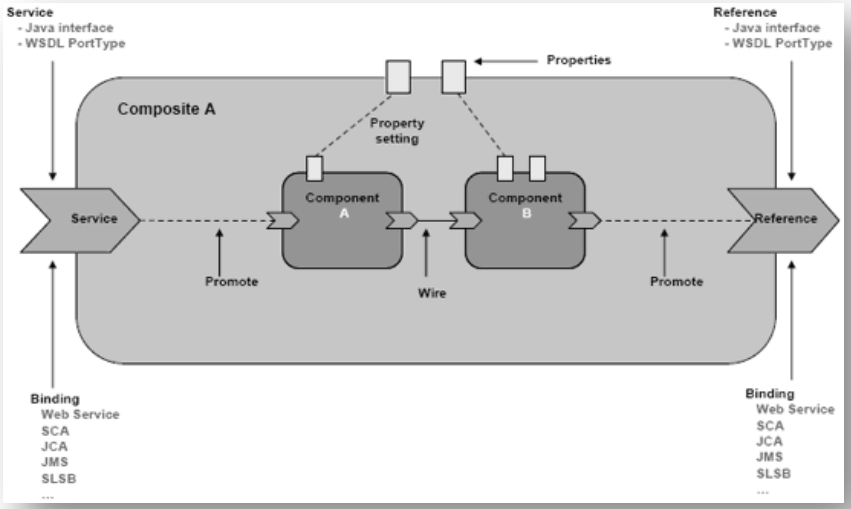
Schaffen von Schnittstellen

- Direkter Zugriff auf Datenbank, Datei, LDAP, etc. und Bereitstellung der Daten als Webservice

- Bsp.:
  - Vorhandene Software zur Reisekostenabrechnung soll stets die aktuellen Mitarbeiterdaten aus dem LDAP nutzen können (zentrale Datenpflege)
  - Job (zeitgesteuert), der in der DB der Software regelmäßig Mitarbeiter anlegt/löscht entsprechend der Informationen im LDAP

SCA (Service Component Architecture)

- Komponenten eines SCA Composites sind z.B. BPEL-Prozesse, Adapter, etc.
  - Realisierung einer SOA
  - Ziel: Entkopplung durch einheitliche Schnittstelle
  - Zusammenfassung von Services zu einer Anwendung



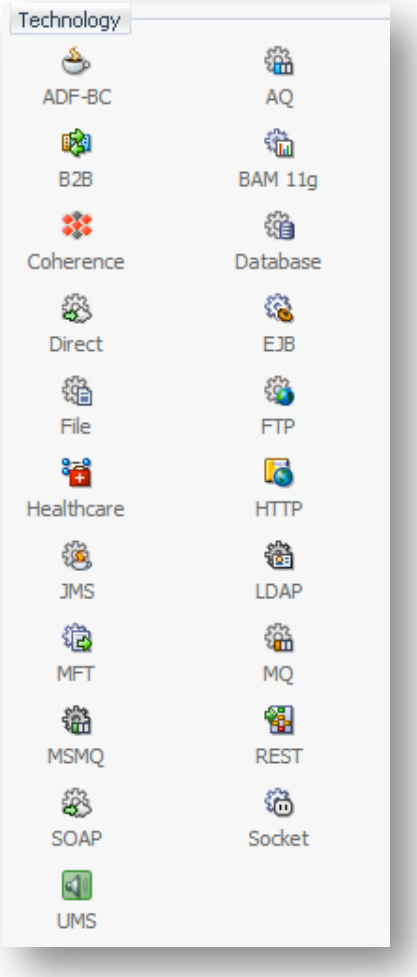
http://www.theenterprisearchitect.eu/blog/2009/03/11/what-every-architect-should-now-know-about-the-service-component-architecture-sca/

„A SOA composite is an assembly of services, service components, and references designed and deployed in a single application."

https://docs.oracle.com/cd/E23943_01/dev.1111/e10224/fod_intro1.htm#SOASE1444

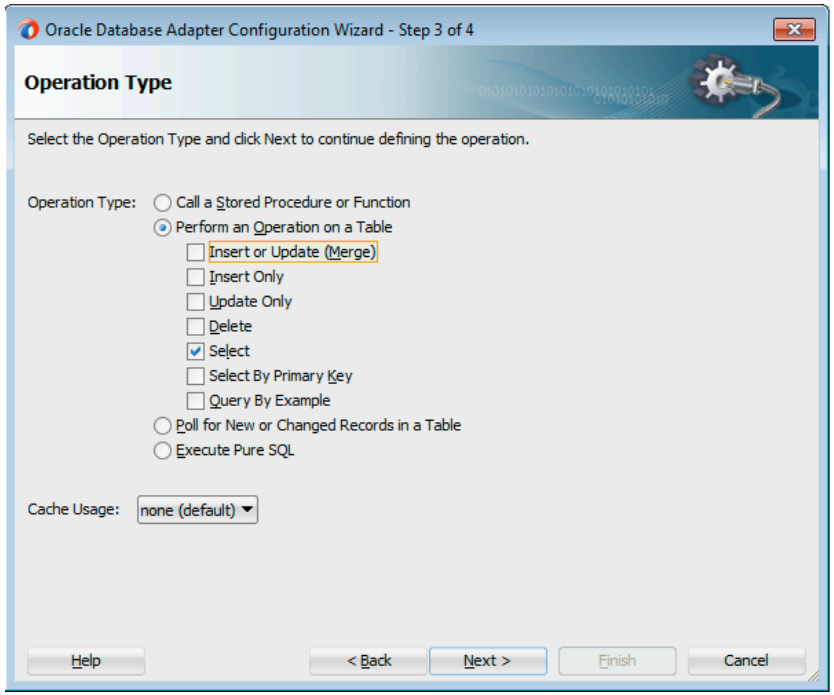
Adapter

- Adapter stellen Schnittstellen zwischen dem Application Server und externen Systemen (z.B. Datenbanken) dar
  - Mehr dazu: JCA (Java EE Connector Architecture)
- Diese werden im SCA als Webservice gekapselt, sodass sie im BPEL-Prozess genutzt werden können



DB Adapter

- Zugriff auf die Datenbank ohne selbst SQL schreiben zu müssen
- Datenquelle muss auf dem Application Server konfiguriert sein
  → JDBC (+ JNDI)
- Abbilden der gewünschten Operation als Webservice
- D.h. Datentransfer über XML
  - DB-Adapter erzeugt Schema



JDBC

- Java Database Connectivity
  - Datenbankschnittstelle für Java (Treiber für unterschiedliche Hersteller, die jeweils den Zugriff auf die DB über JDBC-API abbilden)
    - Treiber laden:

```
Class.forName("org.postgresql.Driver");
```

oracle.jdbc.OracleDriver

com.mysql.jdbc.Driver

    - Verbindung herstellen:

```
Connection db = DriverManager.getConnection(url,
   username, password);
```

jdbc:postgresql://host:port/database

jdbc:oracle:thin:@//[HOST][:PORT]/SERVICE

jdbc:mysql://localhost/test

    - SQL Befehl erstellen und ausführen:

```
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM mytable WHERE columnfoo = 500");
```

    - Über Ergebnismenge iterieren:

```
while (rs.next()) {
    System.out.print("Column 1 returned ");
    System.out.println(rs.getString(1));
}
```

    - Verbindung schließen:

```
rs.close();
st.close();
```

http://jdbc.postgresql.org/documentation/80/

JNDI

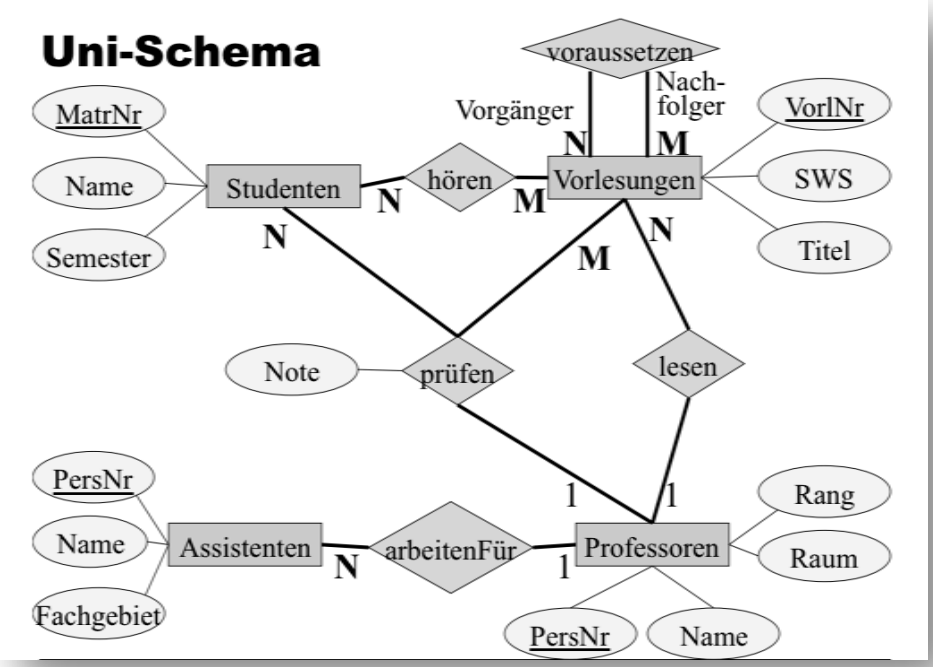
- Java Naming and Directory Interface
  - API zum Zugriff auf Verzeichnisdienste
    - z.B. LDAP
  - Lookup: Anhand eines Namens Informationen zu einem Objekt erhalten
- Anwendung:
  - z.B. zentrales Speichern der JDBC-Verbindungsinformationen (z.B. in Application Server) und Abrufen über ein JNDI-Lookup
    - z.B. über den String `jdbc/myDatabase`

DB Adapter „Magic“

- Select
  - Verwendete Tabellen angeben
  - Relationen werden automatisch ermittelt
  - Abfrage wird erzeugt (kein manuelles Erzeugen von Joins notwendig)
- Query by Example
  - Es wird nicht festgelegt nach welchem Attribut gesucht werden soll
  - Das Erzeugen der Abfrage geschieht dynamisch, je nachdem welche Elemente im Anfrageschema gefüllt sind
- Merge
  - Je nachdem ob der Datensatz (identifiziert über Primärschlüssel) schon existiert oder nicht, wird ein Update oder Insert durchgeführt

Beispiel

- Bekannt aus Datenbanken I (Prof. Raab-Düsterhöft)
  - Schema von A. Kemper



[A. Kemper, A. Eickler: Datenbanksysteme – Eine Einführung]

Daten

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesen von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

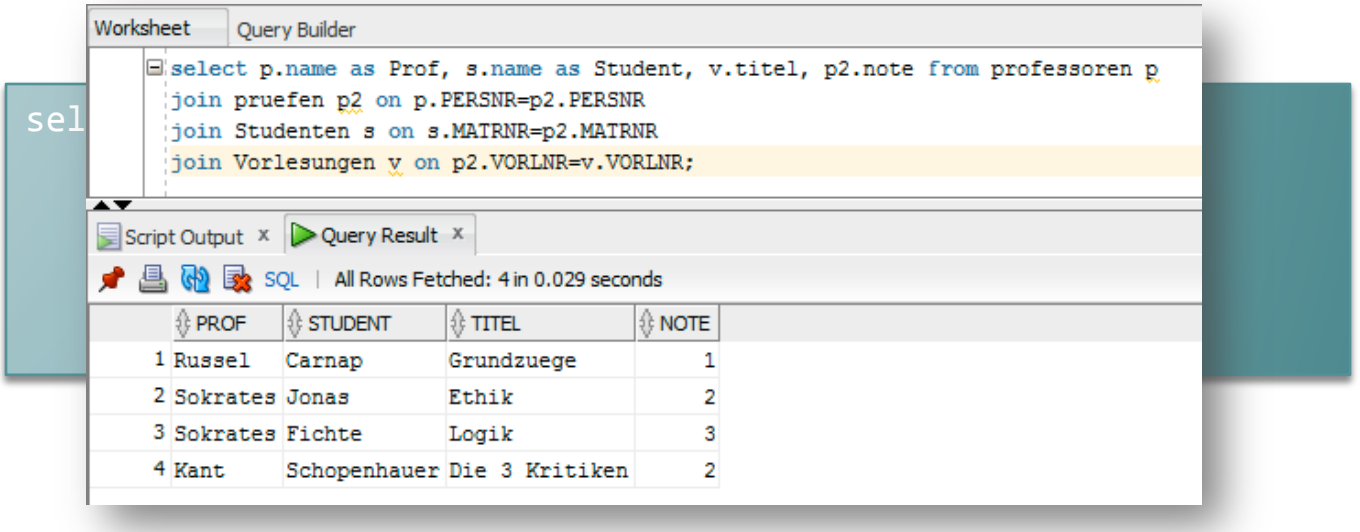
Assistenten			
PerslNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

[A. Kemper, A. Eickler: Datenbanksysteme – Eine Einführung]

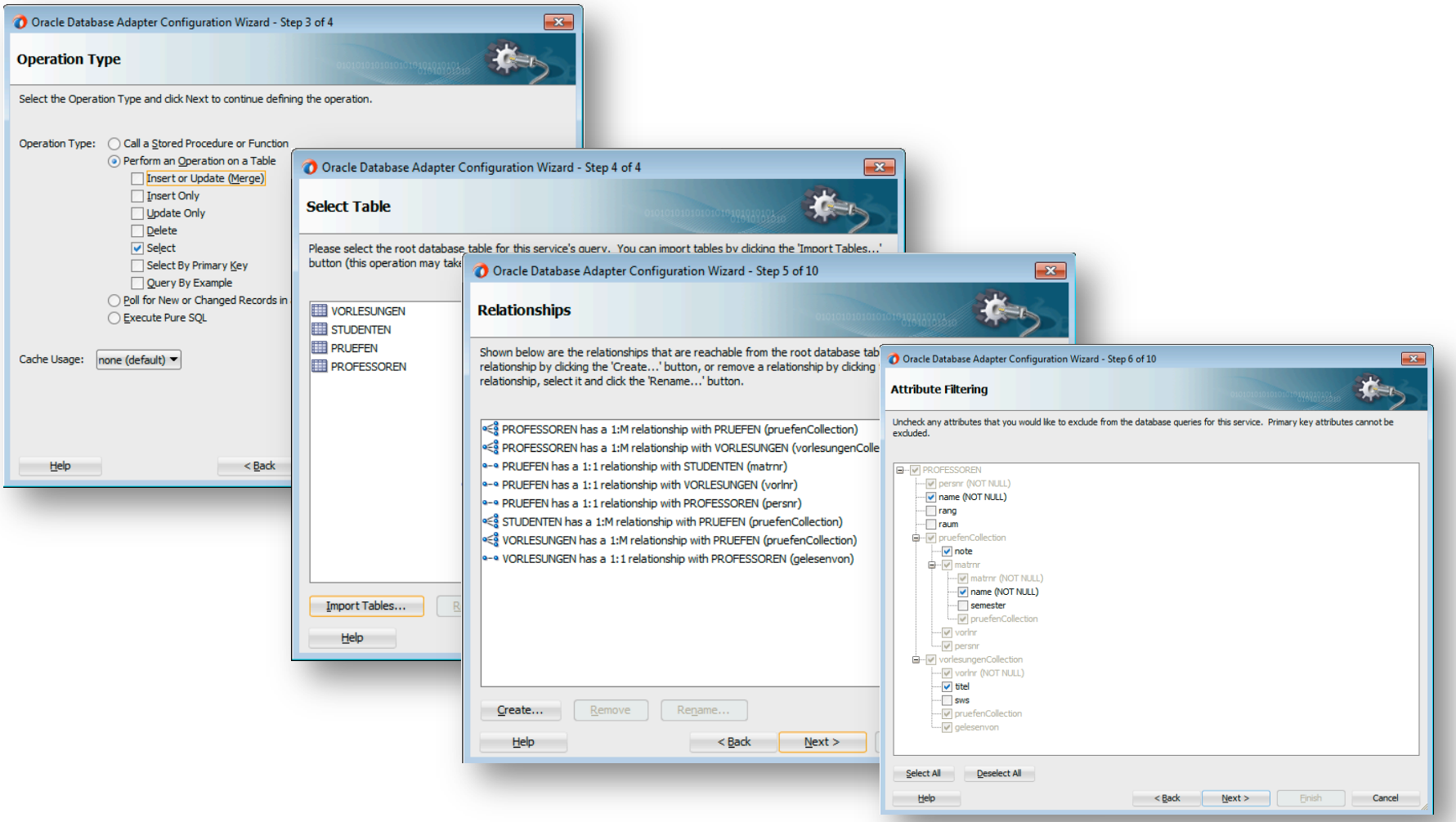
SQL

- *Ermitteln welcher Student welches Fach bei welchem Professor mit welcher Note abgelegt hat…*

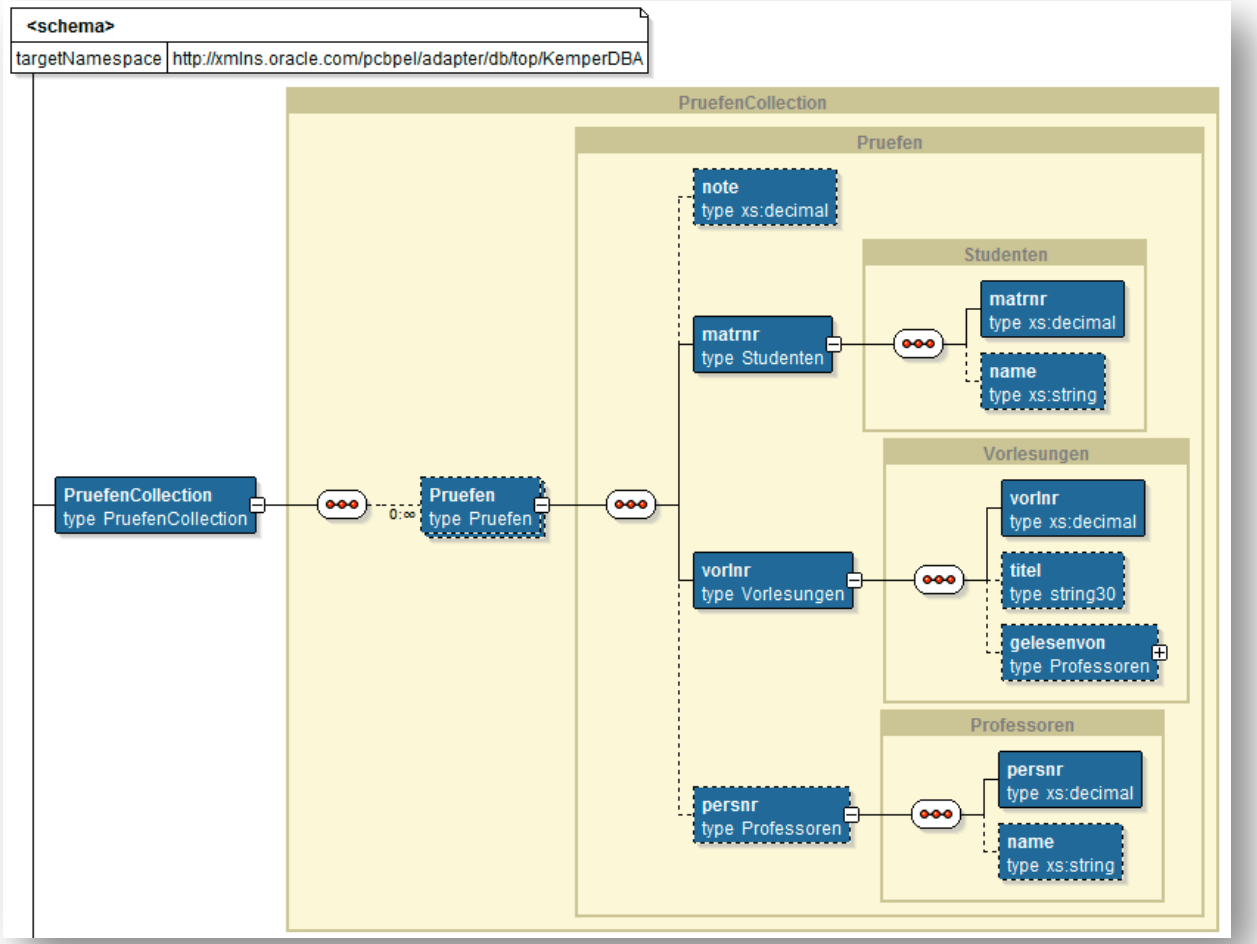


→ Nun über den Datenbank-Adapter…

DB Adapter konfigurieren

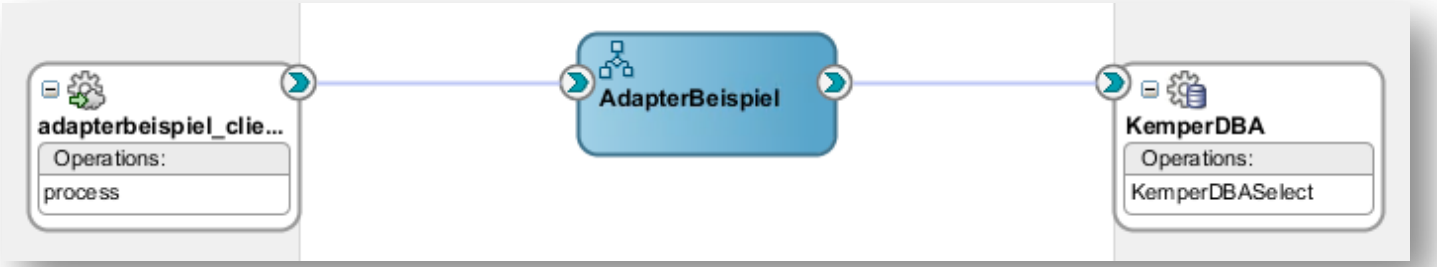


DB Resultset als XML Schema



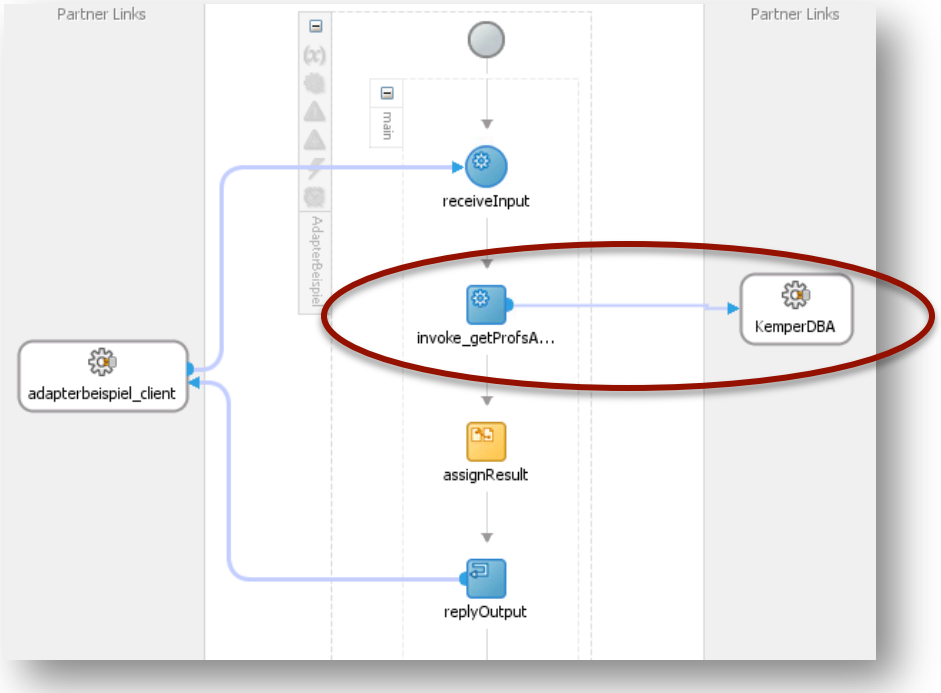
SCA

- BPEL Prozess mit DB-Adapter verbunden
  - DB als Webservice gekapselt



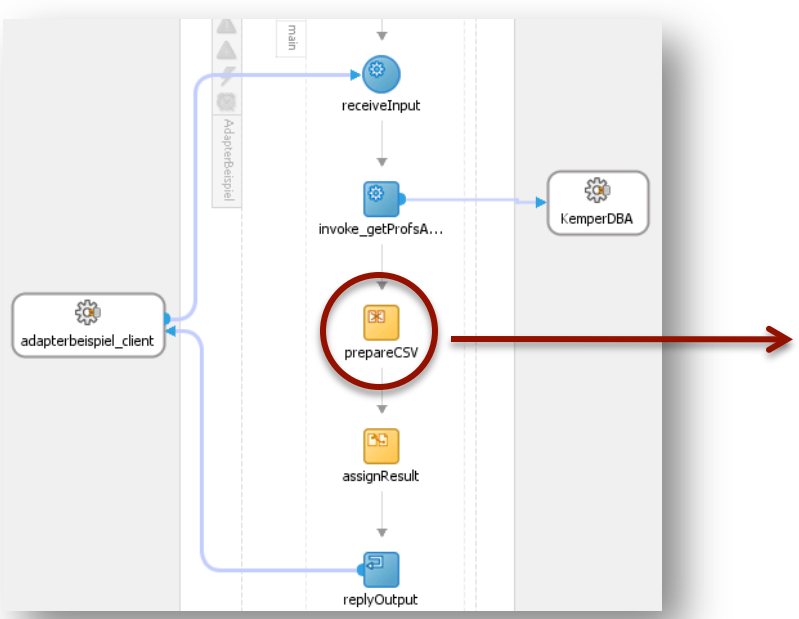
BPEL

- Invoke des Datenbankadapters



Transformierte Ausgabe

- XSLT, um Daten weiter zu filtern




```
<processResponse xmlns="http://mmberg.net/ase/AdapterBeispiel">
   <result xmlns:client="http://mmberg.net/ase/AdapterBeispiel"
      <tns:Pruefung>
         <tns:Prof>Russel</tns:Prof>
         <tns:Student>Carnap</tns:Student>
         <tns:Fach>Grundzuege</tns:Fach>
         <tns:Note>1</tns:Note>
      </tns:Pruefung>
      <tns:Pruefung>
         <tns:Prof>Sokrates</tns:Prof>
         <tns:Student>Jonas</tns:Student>
         <tns:Fach>Ethik</tns:Fach>
         <tns:Note>2</tns:Note>
      </tns:Pruefung>
      <tns:Pruefung>
         <tns:Prof>Kant</tns:Prof>
         <tns:Student>Schopenhauer</tns:Student>
         <tns:Fach>Die 3 Kritiken</tns:Fach>
         <tns:Note>2</tns:Note>
      </tns:Pruefung>
   </result>
</processResponse>
```

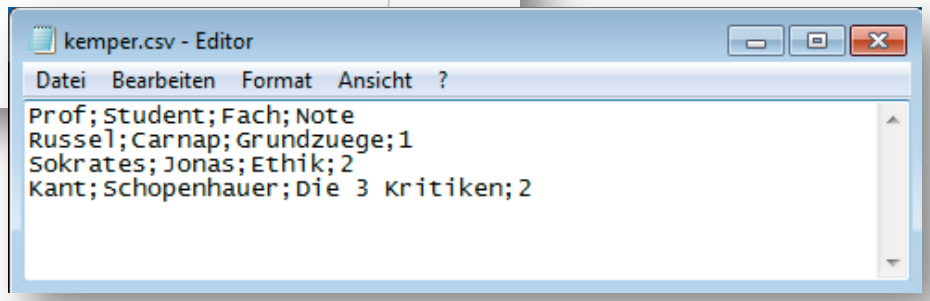
Beispiel (Schritt II)

- Das Beispiel geht weiter…
  - Ergebnisse sollen nun zur weiteren Verwendung als CSV-Datei gespeichert werden (Anwendungsfall: externes Programm benötigt die Daten und versteht nur CSV)

File Adapter

- Erzeugen von Dateien, z.B. CSV
- Grundlage ist ein XML Schema

```
<xsd:element name="Pruefungen">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Pruefung" minOccurs="1" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Prof" type="xsd:string" nxsd:style="terminated" nxsd:terminatedBy=";" nxsd:quotedBy="&quot;"/>
            <xsd:element name="Student" type="xsd:string" nxsd:style="terminated" nxsd:terminatedBy=";" nxsd:quotedBy="&quot;"/>
            <xsd:element name="Fach" type="xsd:string" nxsd:style="terminated" nxsd:terminatedBy=";" nxsd:quotedBy="&quot;"/>
            <xsd:element name="Note" type="xsd:string" nxsd:style="terminated" nxsd:terminatedBy="${eol}" nxsd:quotedBy="&quot;"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



Beispiel (Schritt III)

- Nun wollen wir weitere Prüfungsergebnisse der Datenbank hinzufügen können

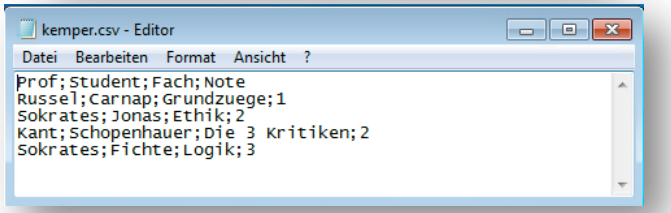
DB-Adapter: Insert

- DB Adapter mit Operation „insert“ erzeugen
- In BPEL aufrufen (assert, invoke)
- Request mit SOAP UI
- Ergebnis in DB kontrollieren
- FileAdapter aus letztem Schritt hat CSV aktualisiert

```
<soapenv:Envelope xmlns:soapenv="http
   <soapenv:Header/>
   <soapenv:Body>
      <adap:addPruefungRequest>
         <adap:Pruefung>
            <kem1:matrnr xmlns:kem1="
            26120
            </kem1:matrnr>
            <kem1:vorlnr xmlns:kem1="
            4052
            </kem1:vorlnr>
            <kem1:persnr xmlns:kem1="
            2125
            </kem1:persnr>
            <kem1:note xmlns:kem1="ht
            3
            </kem1:note>
         </adap:Pruefung>
      </adap:addPruefungRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<addPruefungResponse xmlns="http://mmberg.net/ase/AdapterBeispiel">
   <Response>Pruefung gespeichert</Response>
</addPruefungResponse>
```

	MATRNR	VORLNR	PERSNR	NOTE
1	28106	5001	2126	1
2	25403	5041	2125	2
3	27550	4630	2137	2
4	26120	4052	2125	3

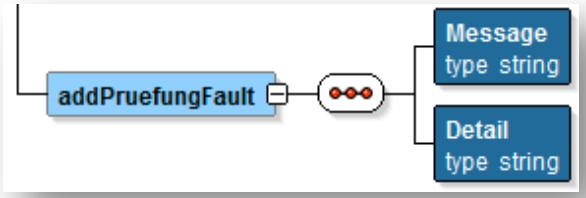


Beispiel (Schritt IV)

- Fehler sollen gefangen und ordnungsgemäß zurückgegeben werden

BPEL: Fehlerbehandlung

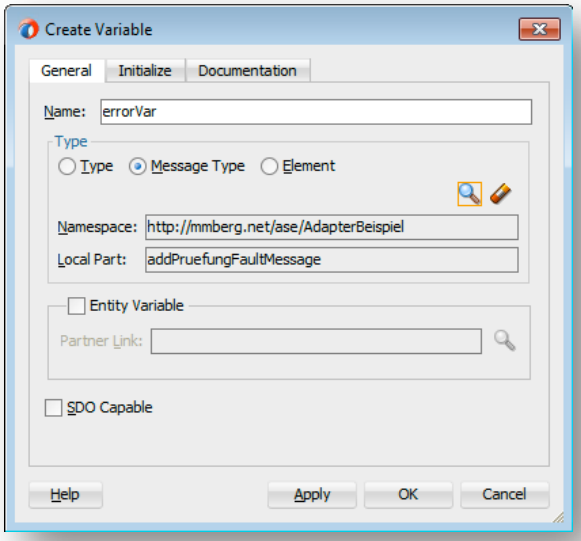
- Falls beim Einfügen Fehler auftreten, soll eine Meldung an den Aufrufer zurückgegeben werden
- Zunächst Schema und WSDL erweitern

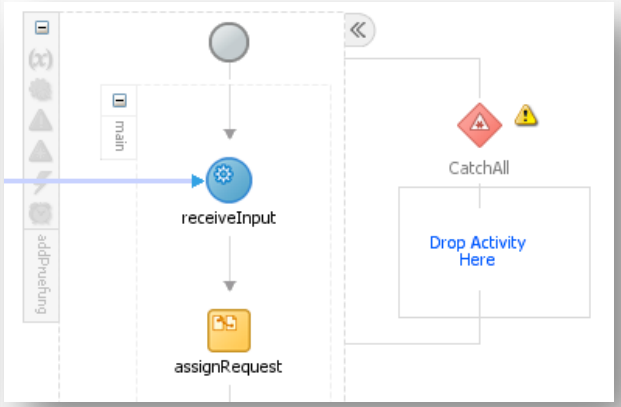


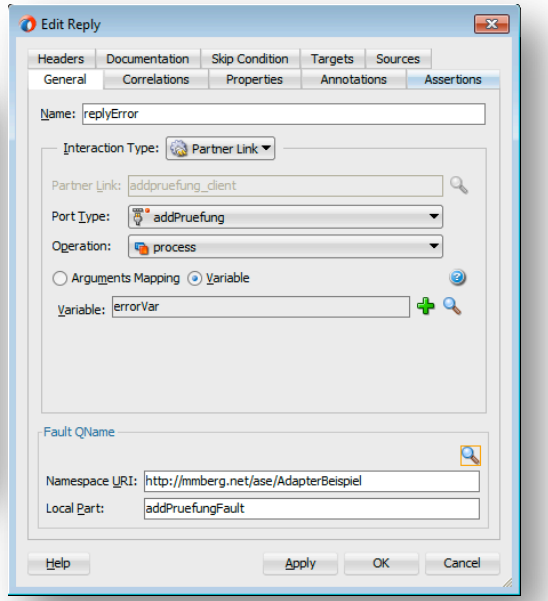

```
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MESSAGE TYPE DEFINITION - Definition of the message types used as
part of the port type defintions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<wsdl:message name="addPruefungRequestMessage">
        <wsdl:part name="payload" element="client:addPruefungRequest"/>
</wsdl:message>
<wsdl:message name="addPruefungResponseMessage">
        <wsdl:part name="payload" element="client:addPruefungResponse"/>
</wsdl:message>
<wsdl:message name="addPruefungFaultMessage">
  <wsdl:part name="payload" element="client:addPruefungFault"/>
</wsdl:message>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PORT TYPE DEFINITION - A port type groups a set of operations into
a logical service unit.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->

<!-- portType implemented by the addPruefung BPEL process -->
<wsdl:portType name="addPruefung">
        <wsdl:operation name="process">
                <wsdl:input  message="client:addPruefungRequestMessage" />
                <wsdl:output message="client:addPruefungResponseMessage"/>
                <wsdl:fault name="addPruefungFault" message="client:addPruefungFaultMessage"/>
        </wsdl:operation>
</wsdl:portType>
```

BPEL: Catch-All

- Fehlervariable erstellen vom Typ der WSDL FaultMessage
- Catch-All hinzufügen
- Reply (synchrone Antwort) erzeugen und Fehlervariable angeben

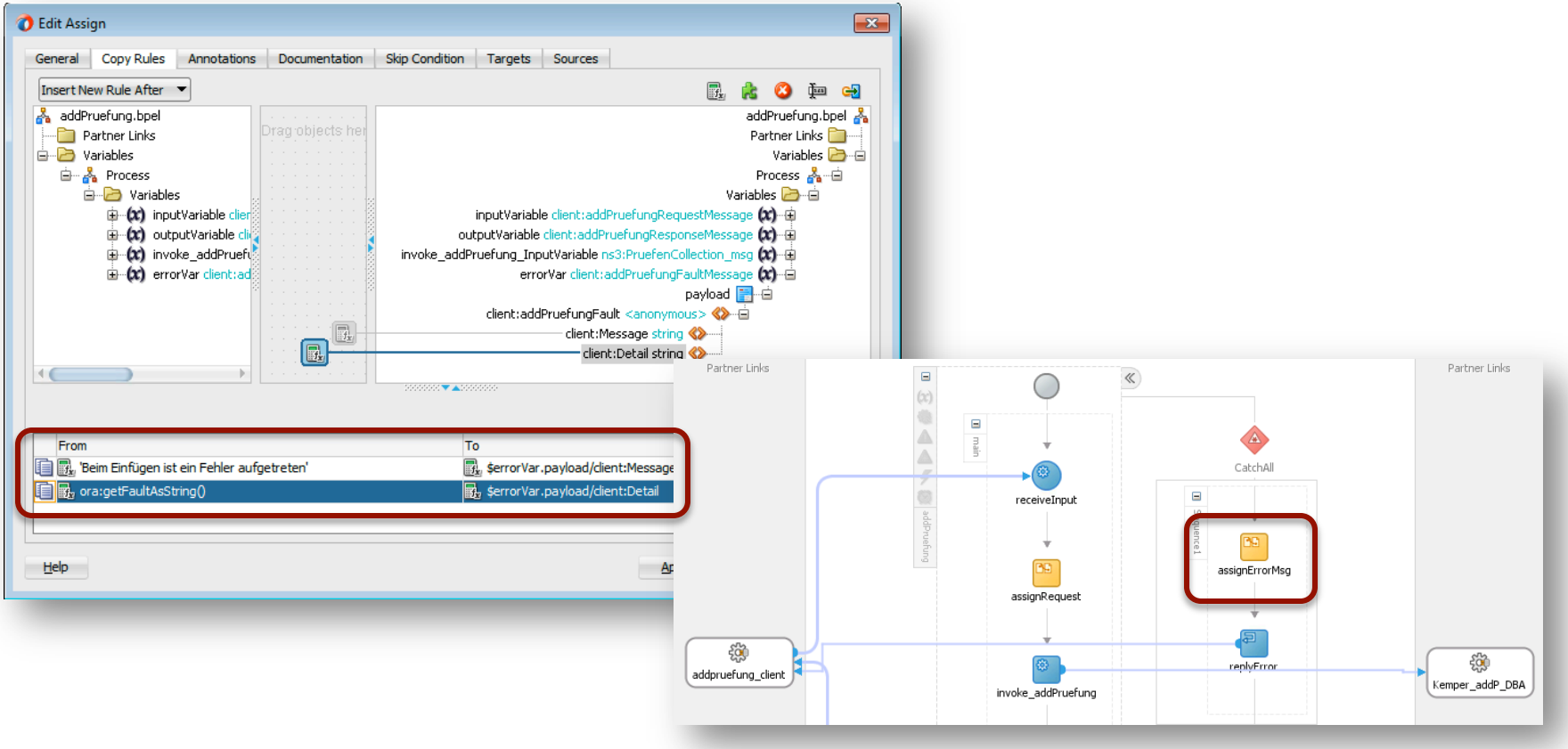






Fehlervariable füllen

- `getFaultAsString()`



Fehlermeldung: Beispiel

- Falscher Primärschlüssel angegeben

```
            <addPruefungFault xmlns="http://mmberg.net/ase/AdapterBeispiel">
               <Message>Beim Einfügen ist ein Fehler aufgetreten</Message>
               <Detail><![CDATA[com.oracle.bpel.client.BPELFault: faultName: {{http://schemas.oracle.
messageType: {{http://schemas.oracle.com/bpel/extension}RuntimeFaultMessage}
parts: {{
summary=<summary>Beim Aufrufen des Bindings ist eine Exception aufgetreten.
Beim Aufrufen des JCA-Bindings ist eine Exception aufgetreten: "JCA Binding execute of Reference oper
merge nicht erfolgreich. Deskriptorname: [Kemper_addP_DBA.Pruefen].
Verursacht von java.sql.SQLIntegrityConstraintViolationException: ORA-02291: integrity constraint (AS
.
Prüfen Sie in den Logs die vollständige DBAdapter-Logging-Ausgabe vor dieser Exception.  Diese Except
".
Der aufgerufene JCA-Adapter hat eine Ressourcen-Exception ausgelöst.
Prüfen Sie die obige Fehlermeldung sorgfältig, um eine Lösung zu finden.
</summary>
,detail=<detail>ORA-02291: integrity constraint (ASE.SYS_C007138) violated - parent key not found
</detail>
,code=<code>2291</code>}]]></Detail>
            </addPruefungFault>
```

Beispiel (Schritt V)

- Zensuren sollen nun nicht mehr als Zahlen (2) sondern als Wörter (z.B. gut) gespeichert werden
- Zur Umwandlung soll eine Java-Klasse genutzt werden

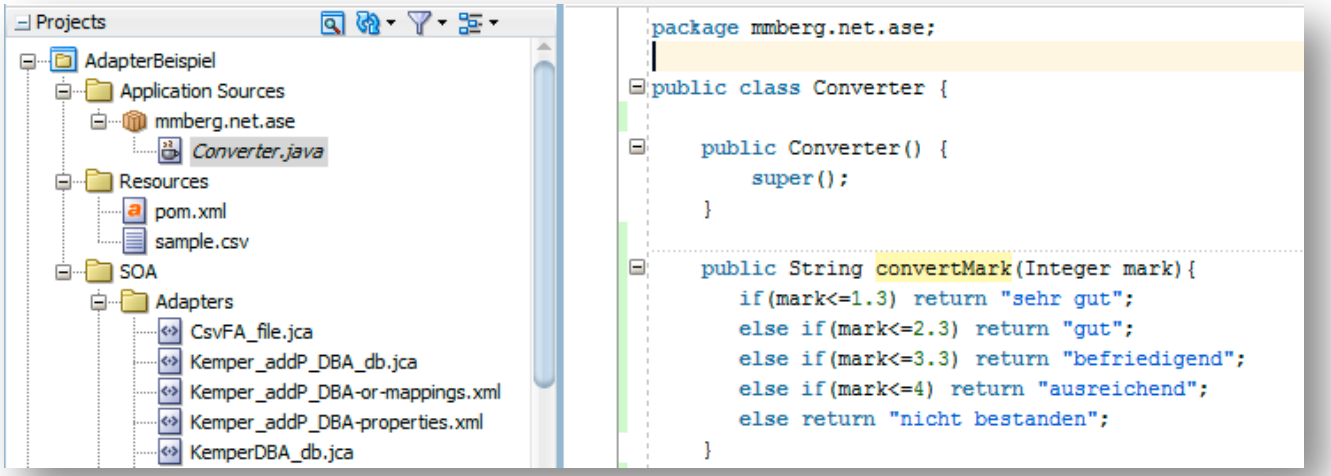
Java-Integration

- Oftmals ist es notwendig, komplexere Vorgänge per Java zu realisieren
  - Möglichkeit 1: externen Java-WS anbieten
  - Möglichkeit 2: Java Spring Bean in Composite integrieren

Hauptmerkmal von Spring ist Dependency Injection. Wir können über eine Konfigurationsdatei bestimmen, welche Klasse zur Instantiierung eines Beans genutzt wird. Für diese Vorlesung reicht es zu wissen, dass wir einfach ein POJO implementieren (öffentlicher Konstruktor, Getter/Setter).

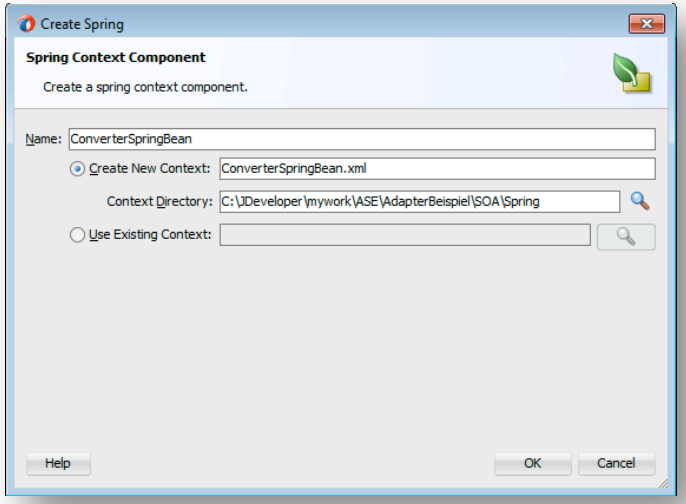
Java Spring Beans

- Java-Klasse (Bean) erzeugen und Interface extrahieren



Spring Bean Context erzeugen (SCA)

- Wizard ausführen

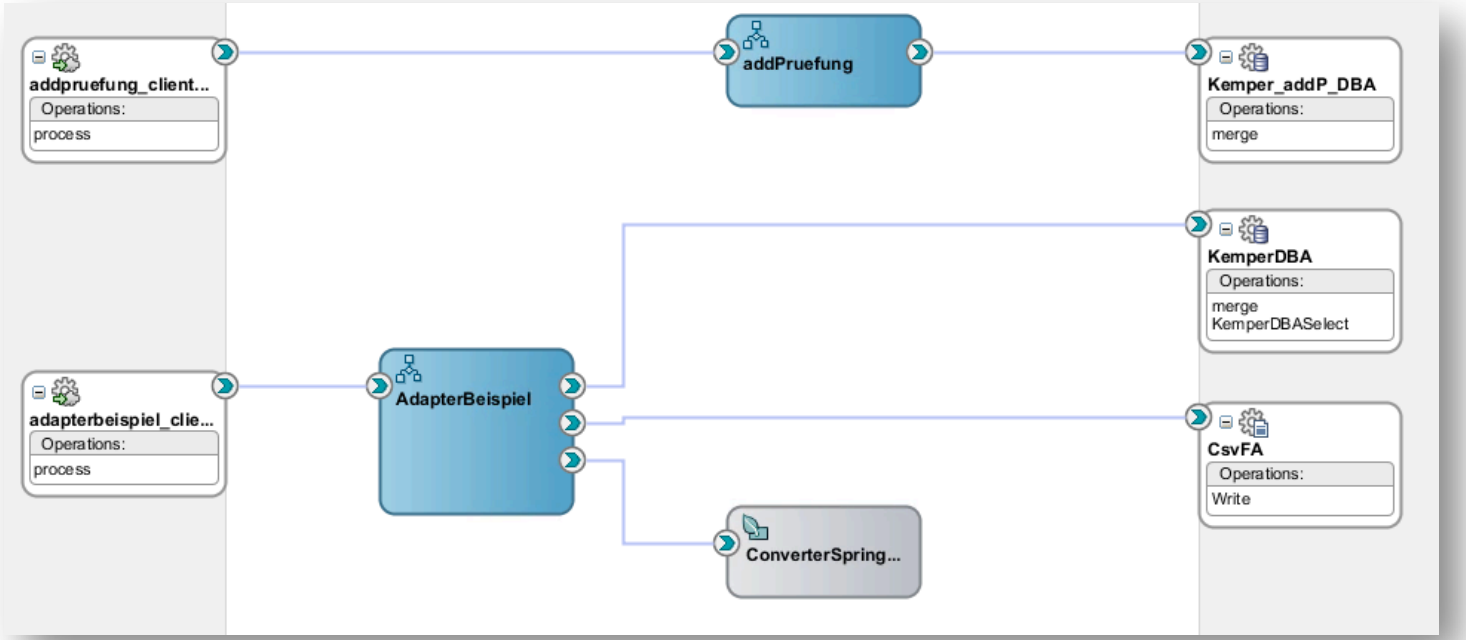


- Anschließend XML ergänzen

```
<!--Spring Bean definitions go here-->
<bean name="impl" class="mmberg.net.ase.Converter"/>
<sca:service name="Converter" target="impl" type="mmberg.net.ase.IConverter"/>
```

SCA Composite Ansicht

- Spring Bean mit BPEL-Prozess verbinden



Bean-Aufruf

- Bean ist als WS gekapselt
- Normales Invoke
- Bsp.: Schleife über alle Prüfungen und Spring Bean aufrufen
  - numerische Zensuren in Wörter umwandeln