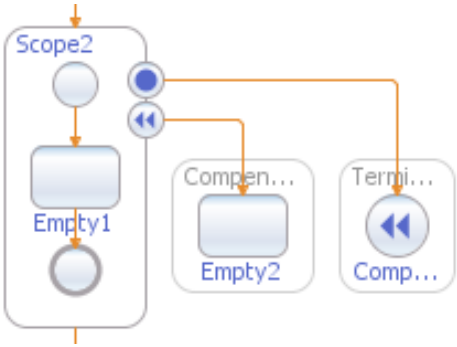
BPEL Schulung

Autoren:
Kaveh Keshavarzi,
Thomas Bayer

predic8 GmbH
Moltkestr. 40
53173 Bonn

www.predic8.de
info@predic8.de



Inhalt

- Einführung
- Partnerlinks
- Variablen
- Aktivitäten
- Fehlerbehandlung
- Compensation
- Correlation
- Message Exchange Pattern
- EventHandling
- Synchronisation
- Bussines Activity Monitoring

Einführung

Business Prozesse

- Long runners (Long-running Transaction LRT)
  - Von Zehn Minuten bis mehreren Monaten
- Alternative Flüsse für Ausnahmen
- Change
- Oft gegenseitige Abhängigkeiten auf Service-Ebene

Orchestration und Choreography

- Steuern von Partnern, die in einem Prozess beteilig sind.
- Orchestration
  - Fokus auf einen Teilnehmer der orchestirert
- Choreography
  - Globaler Ansatz

PI-Calculus

- By Robin Milner (späte 1980er)
- Mathematische Grundlage von BPML und XLANG
- Modellierung von konkurrierenden kommunizierenden Systemen

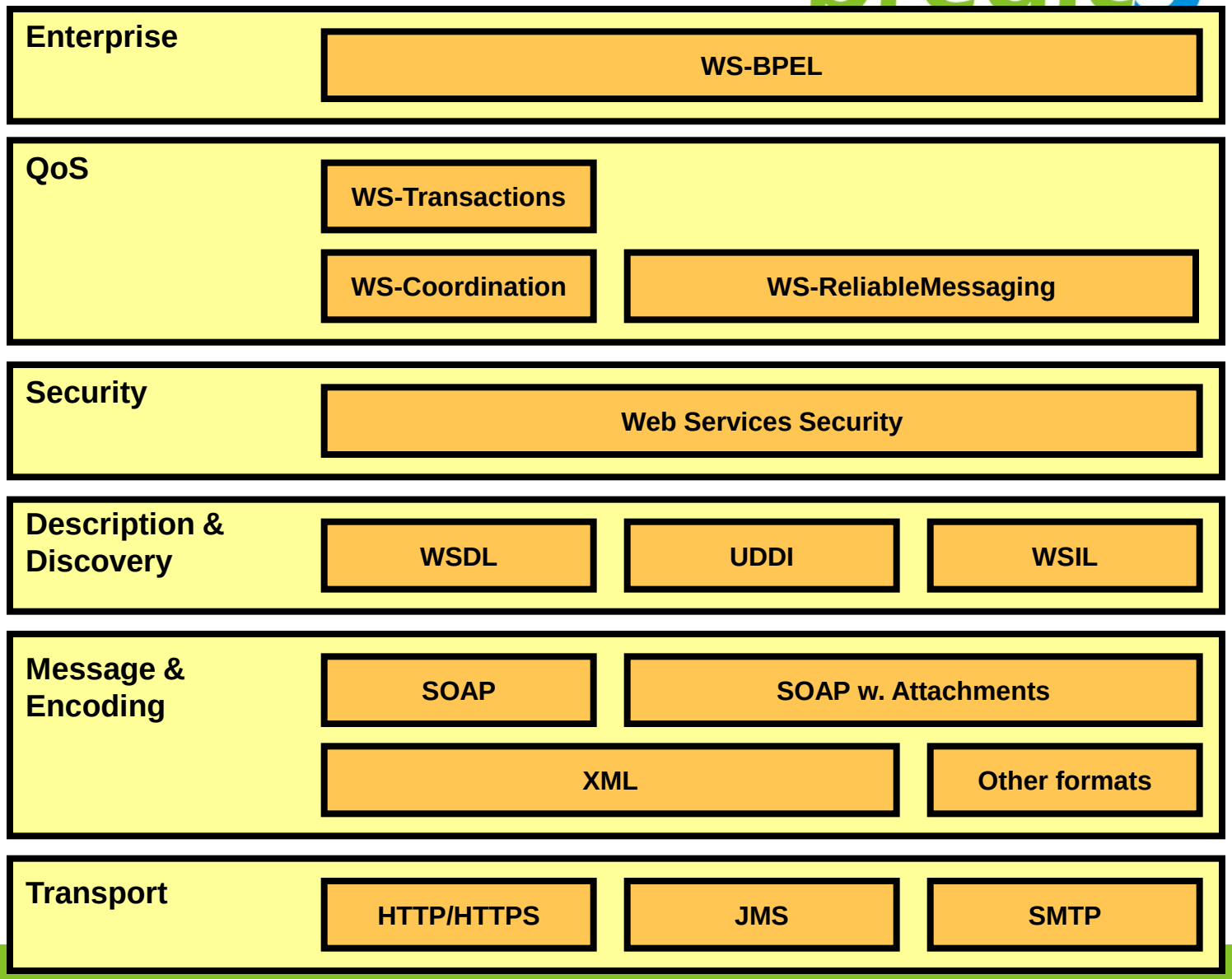
Web Services Business Process Execution Language WS-BPEL

- XML Sprache für auf Web Services basierende Business Prozesse
- Früher bekannt unter BPEL4WS
- Von BEA, IBM (WSFL), Microsoft (XLANG), SAP
- OASIS Standard
- Basiert auf: XML,Schema, XPath, WSDL, XSLT

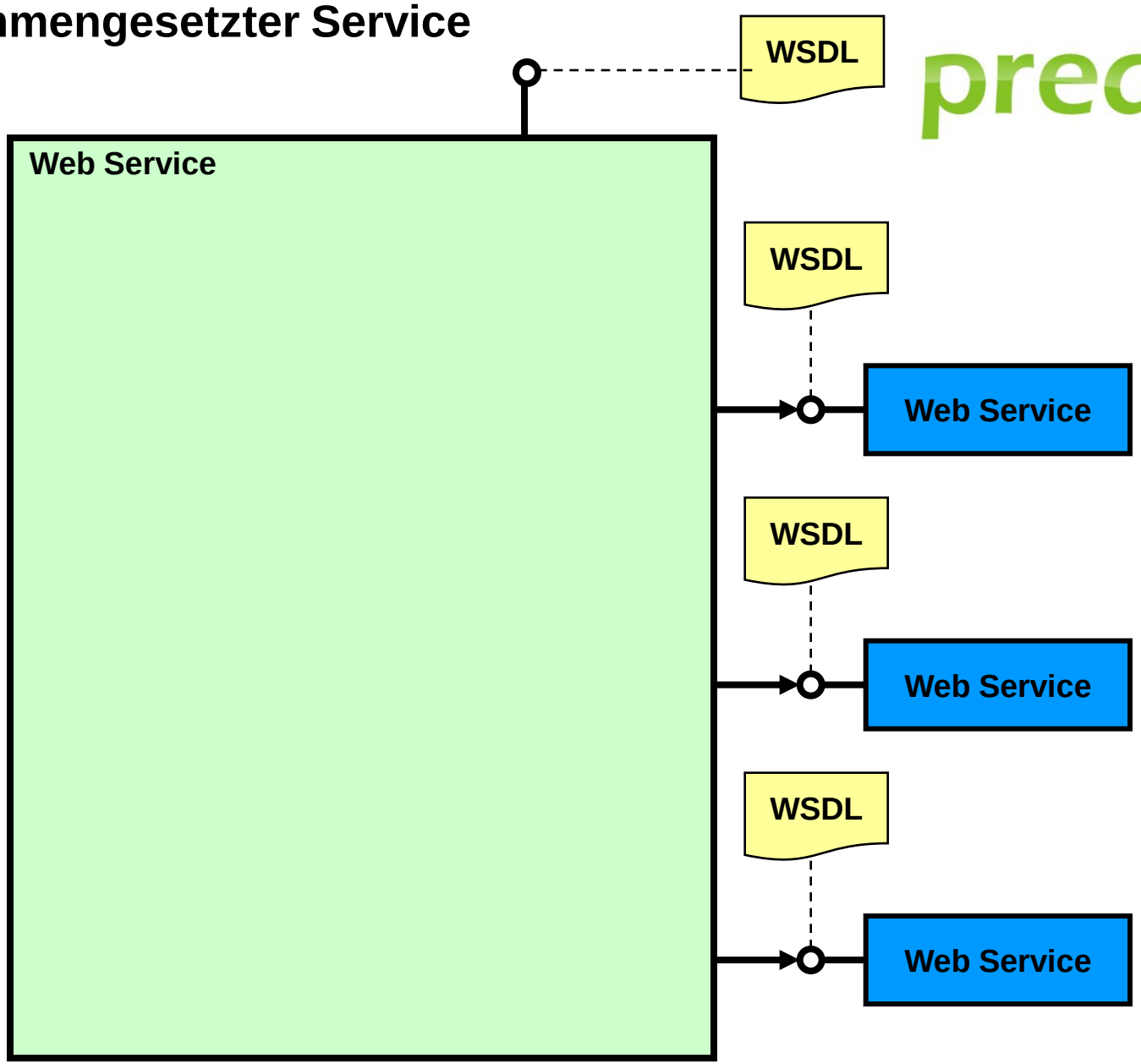
Beziehungen zu anderen Spezifikationen

- XML
- **WSDL 1.1**
- Schema 1.0
- XPath 1.0
- XSLT 1.0
- WS-I Basic Profile 1.1

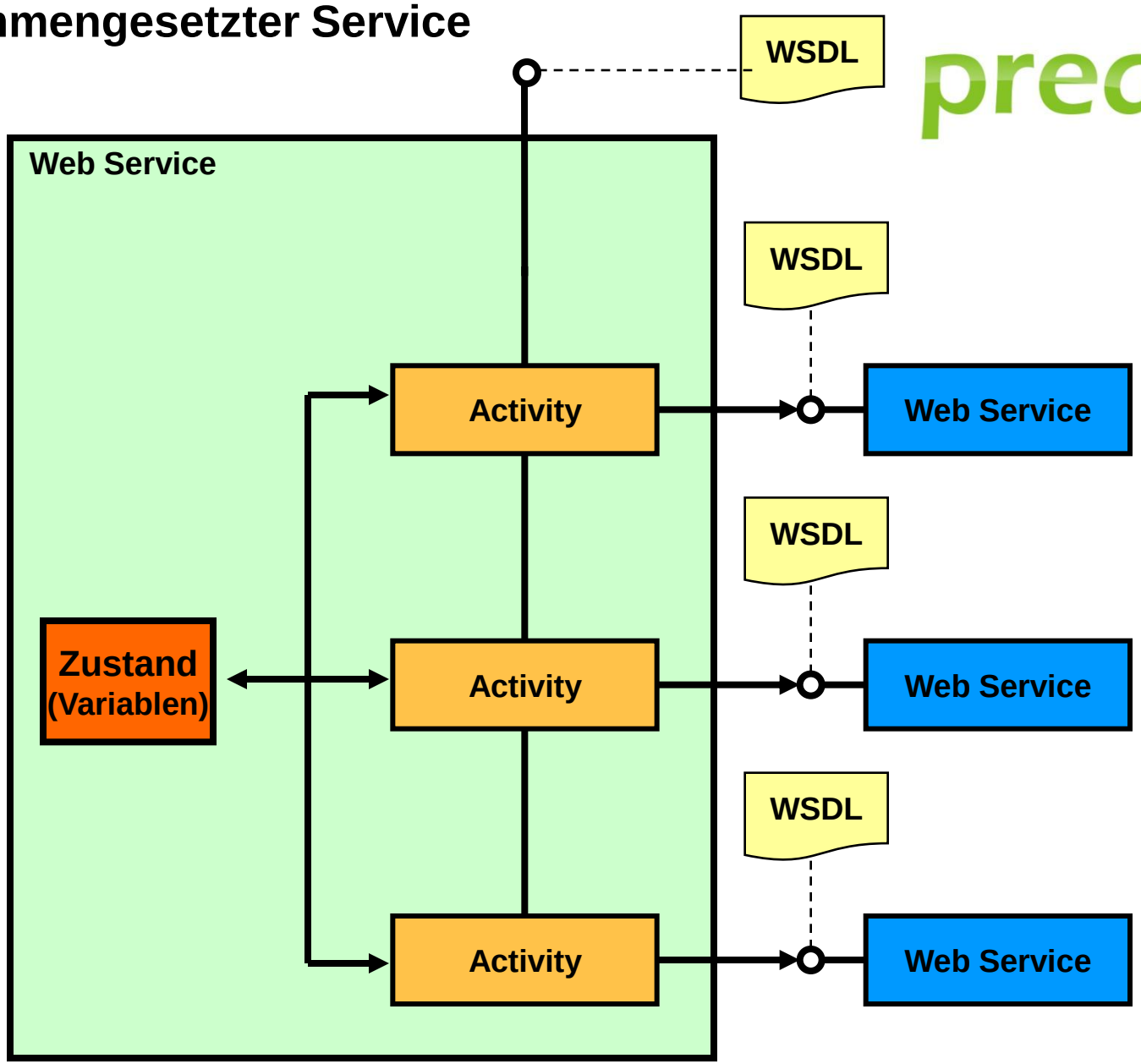
WS Stack



Zusammengesetzter Service



Zusammengesetzter Service



Ausführbare und Abstrakte Prozesse

- Executable Process
  - Beschreibt das Verhalten eines Teilnehmers vollständig
- Business Protocols
  - Beschreibt den Nachrichtenaustausch zw. Den Teilnehmern

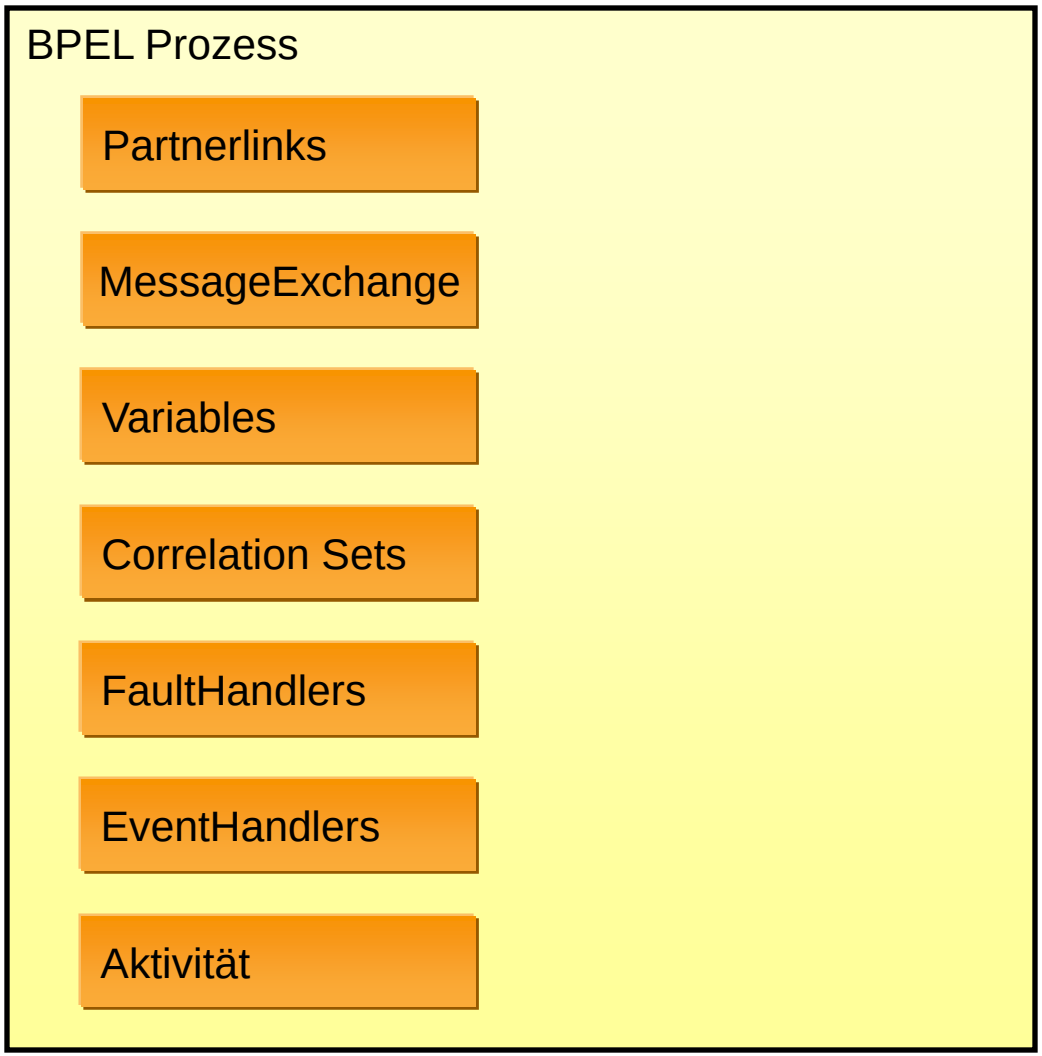
Abstrakter Prozess

- Beschreibung des beobachteten Verhaltens eines Prozesses
- Standardisierte Form
- „Programmierung in the Large“
  - Warten auf Nachrichten
  - Senden von Nachrichten
  - Kompensation
- Nicht ausführbar
- Verbirgt Details

BPEL 2.0:

- Abstrakte Prozesse haben eigenen Namespace

Struktur eines ausführbaren BPEL Prozesses



WS-BPEL - Konzepte

- Container speichern Nachrichten und Statusinformationen
- Fault Handling ähnlich zu Java
  - Catch Block innerhalb XML Struktur
- Fehlerfälle können Aktivitäten zugeordnet werden
  - Compensation Handler
- Interaktionen zwischen Diensten mittels Partner Link
  - PLs beschreiben Interaktionen und angebotener Funktionsumfang
  - Rollenkonzept für Definition

BPEL Projekt

- BPEL Datei
  - Beschreibt den Prozess
- WSDL Dateien
  - Beschreiben Schnittstellen
- XML Schema Dateien
  - Beschreiben Datentypen
- JBI Assembly kann BPEL Projekt aufnehmen (Sun)

BPEL Produkte

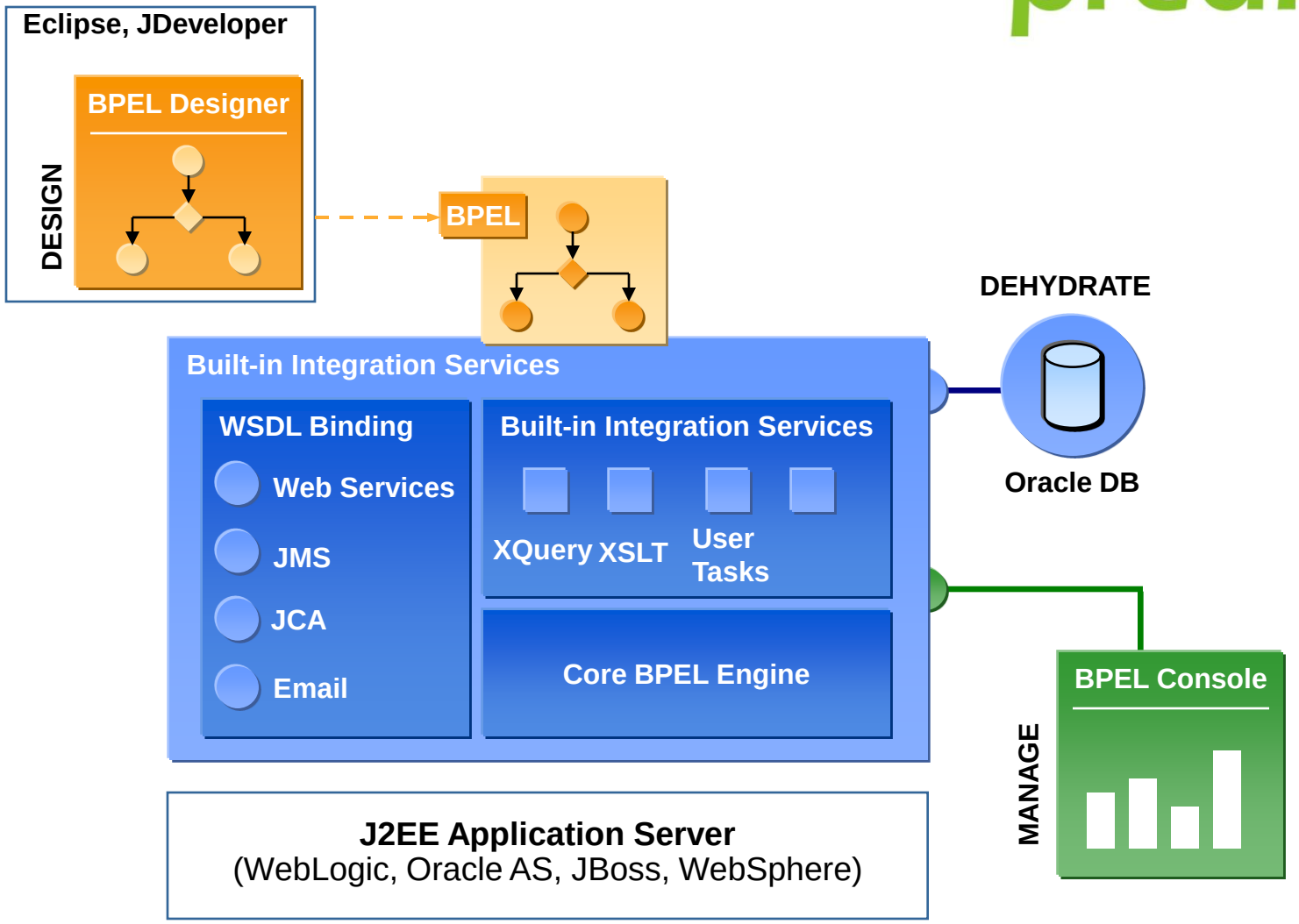
Active BPEL Engine

- Open Source GPL (Nur bis Version 5)
- Läuft in JEE Web Container
- Basis für kommerzielle Produkte von Active Endpoints

Oracle SOA Suite

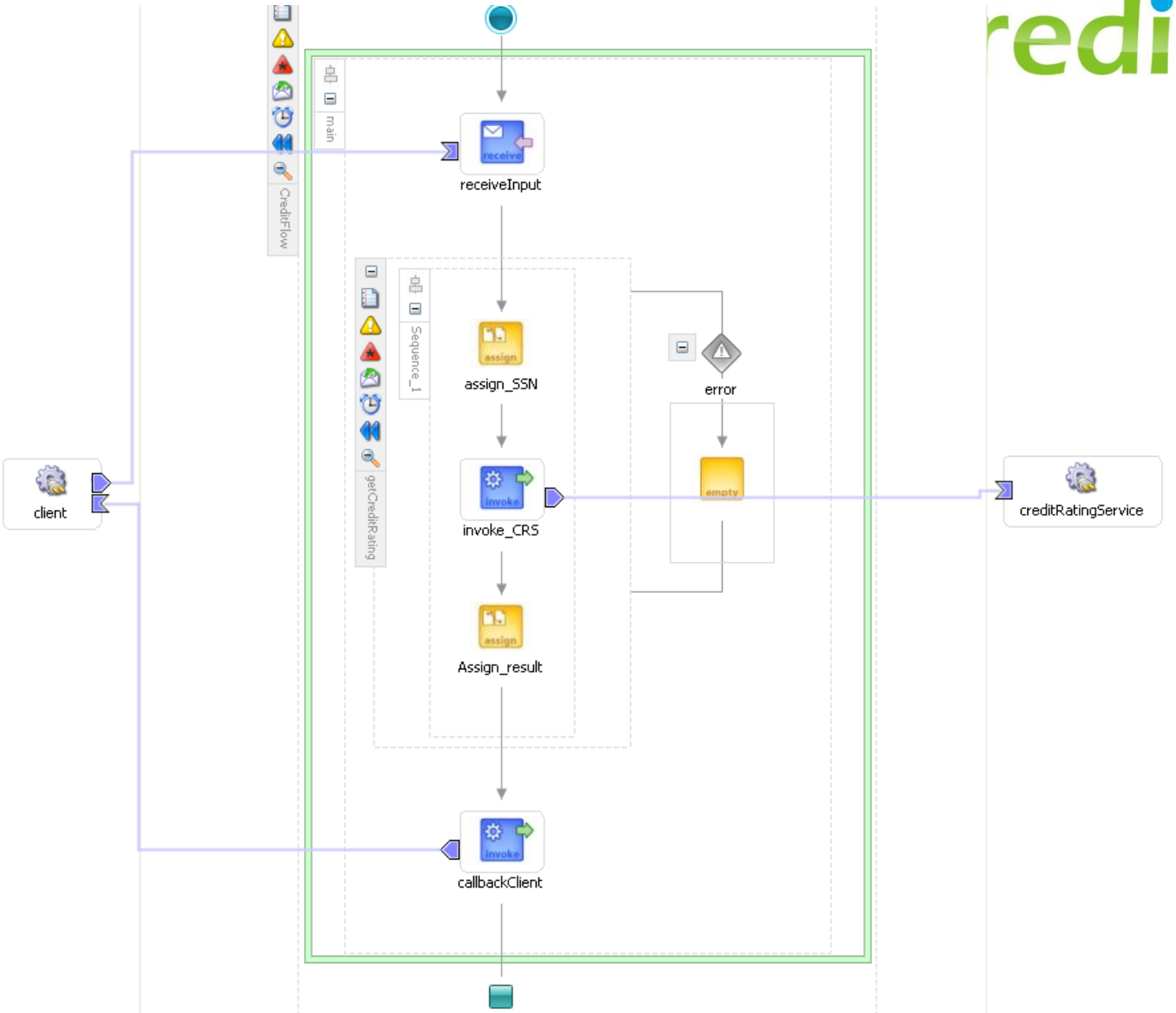
- Basiert auf Collaxa Produkt
- BPEL Designer und Engine
- Business Activity Monitoring (BAM)
- Rules Engine (RETE)
- ESB
- Security
- Registry
- IDE

Oracle BPEL Process Manager

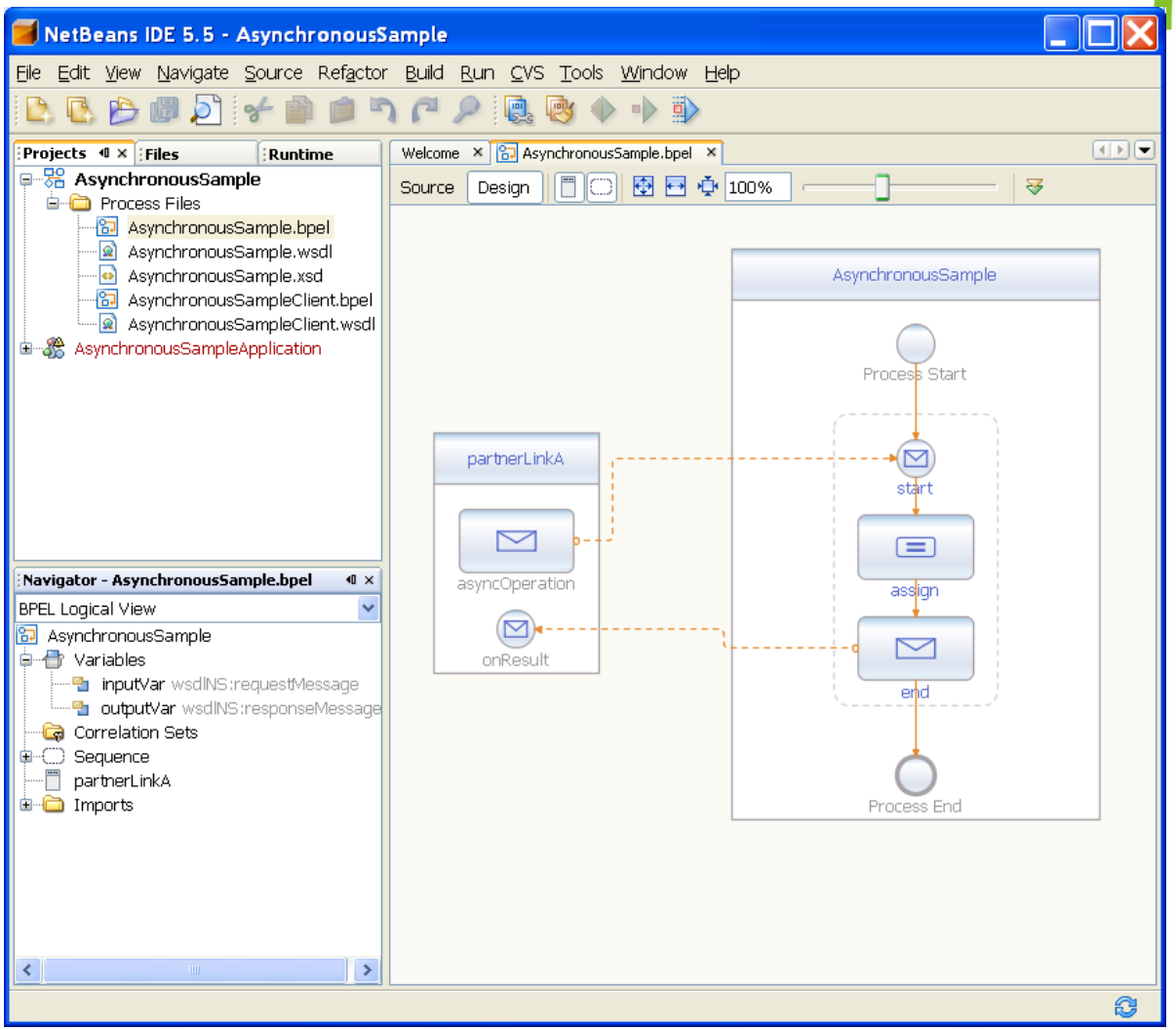


Quelle: http://www.oracle.com/technology/bpel

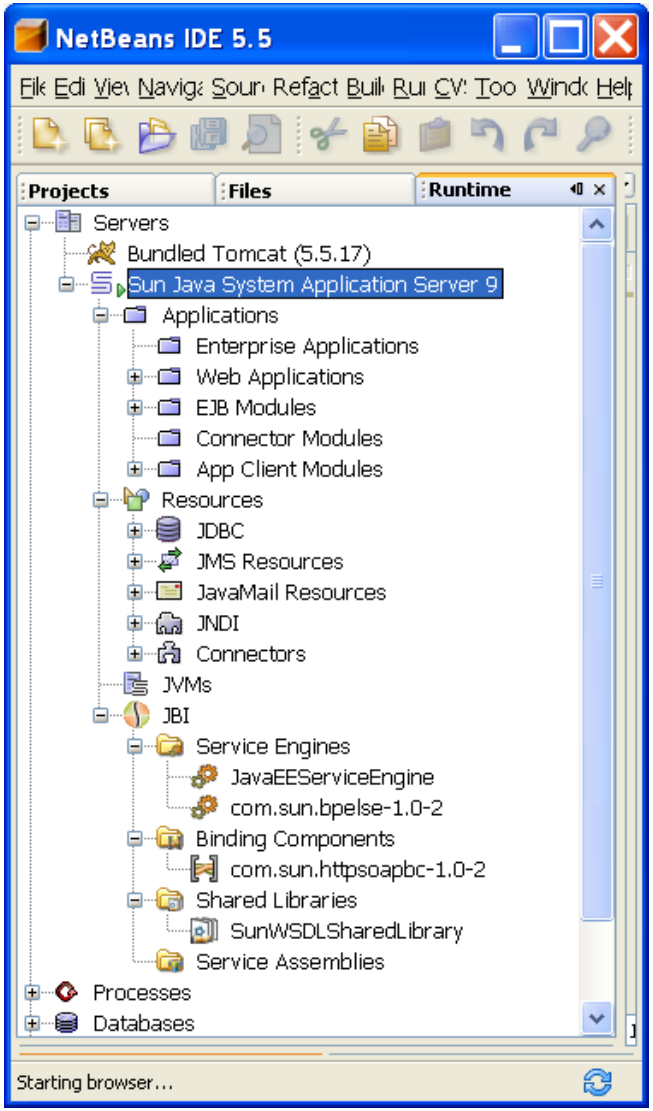
BPEL Prozess



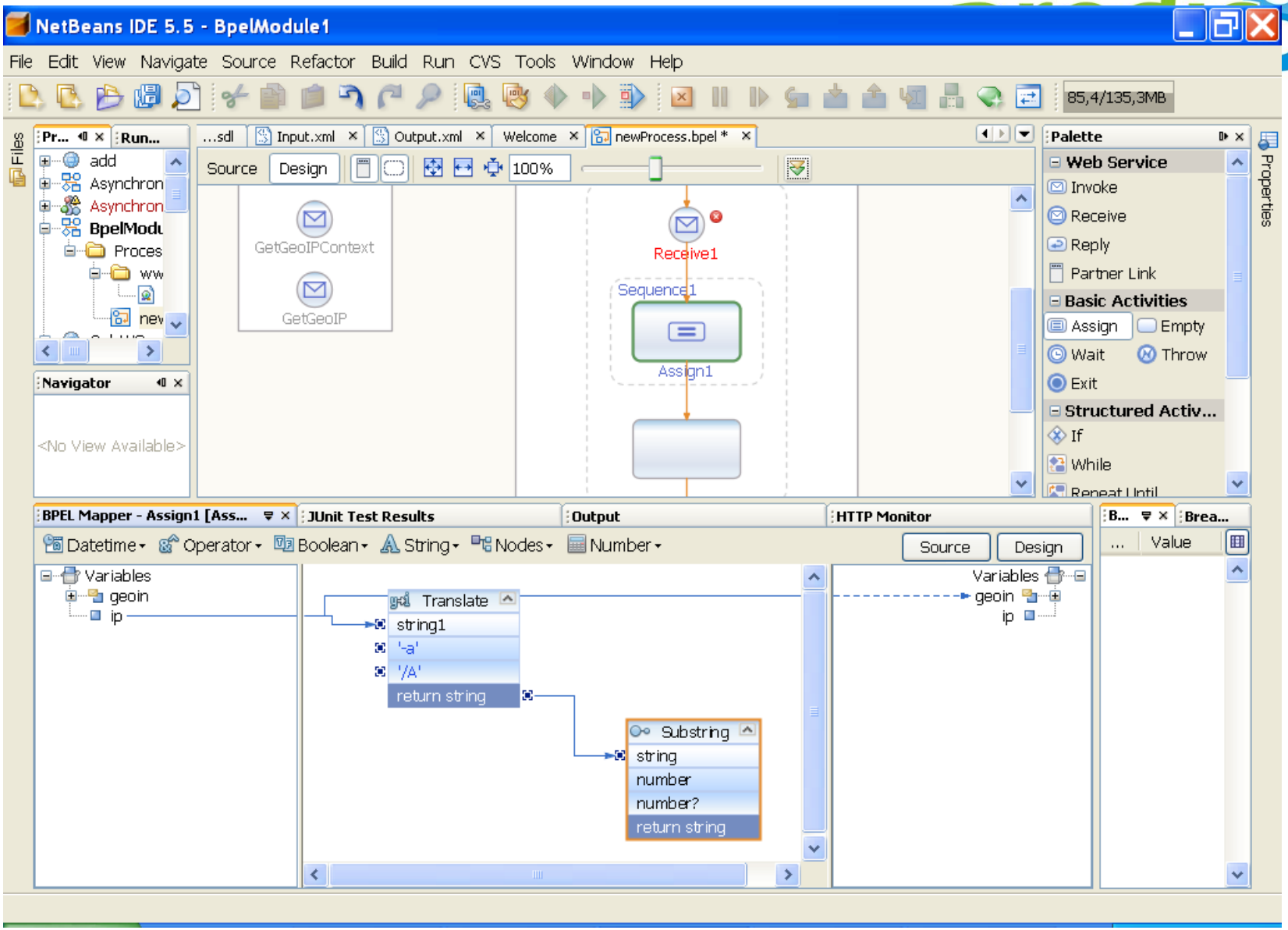
NetBeans BPEL Designer



NetBeans Runtime



Netbeans 5.5 BPEL Designer

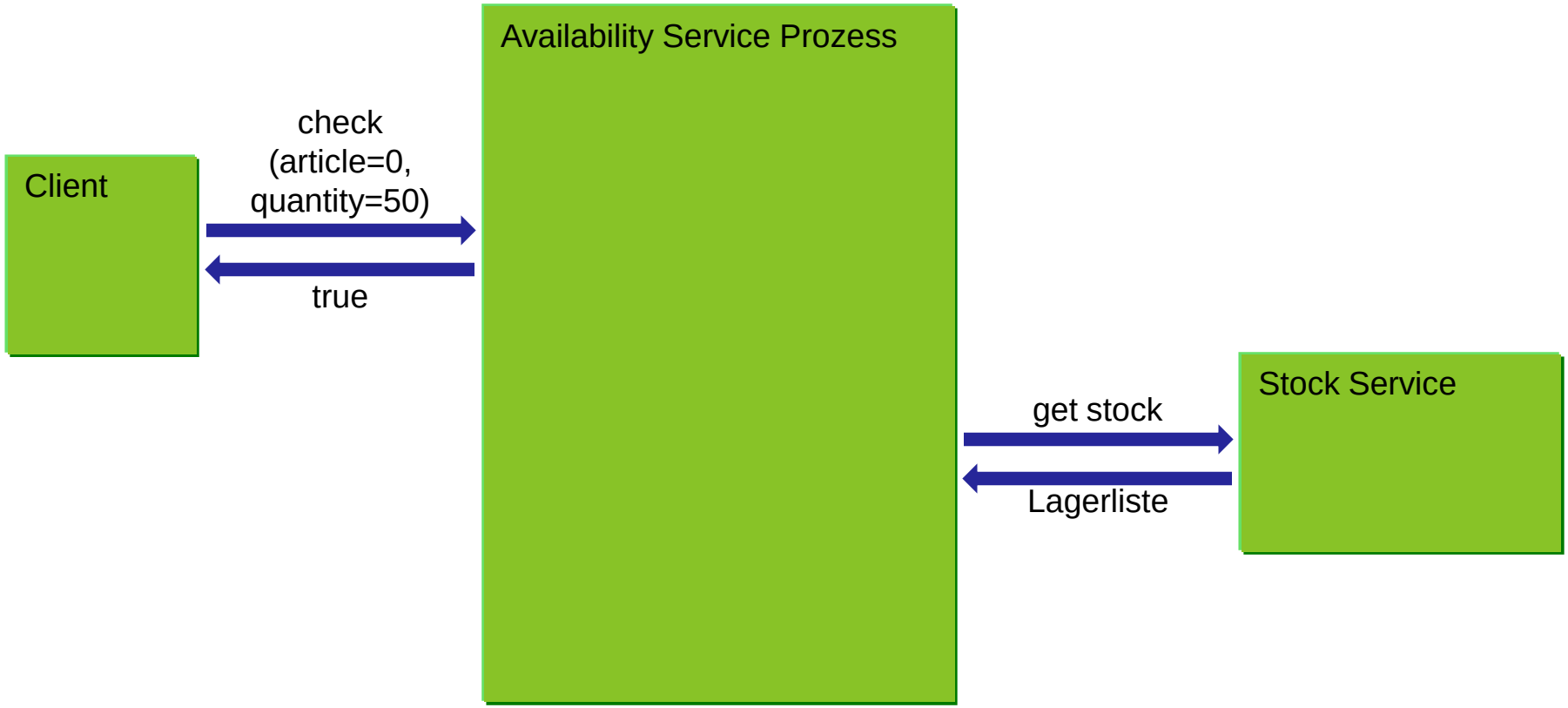


BPEL Server

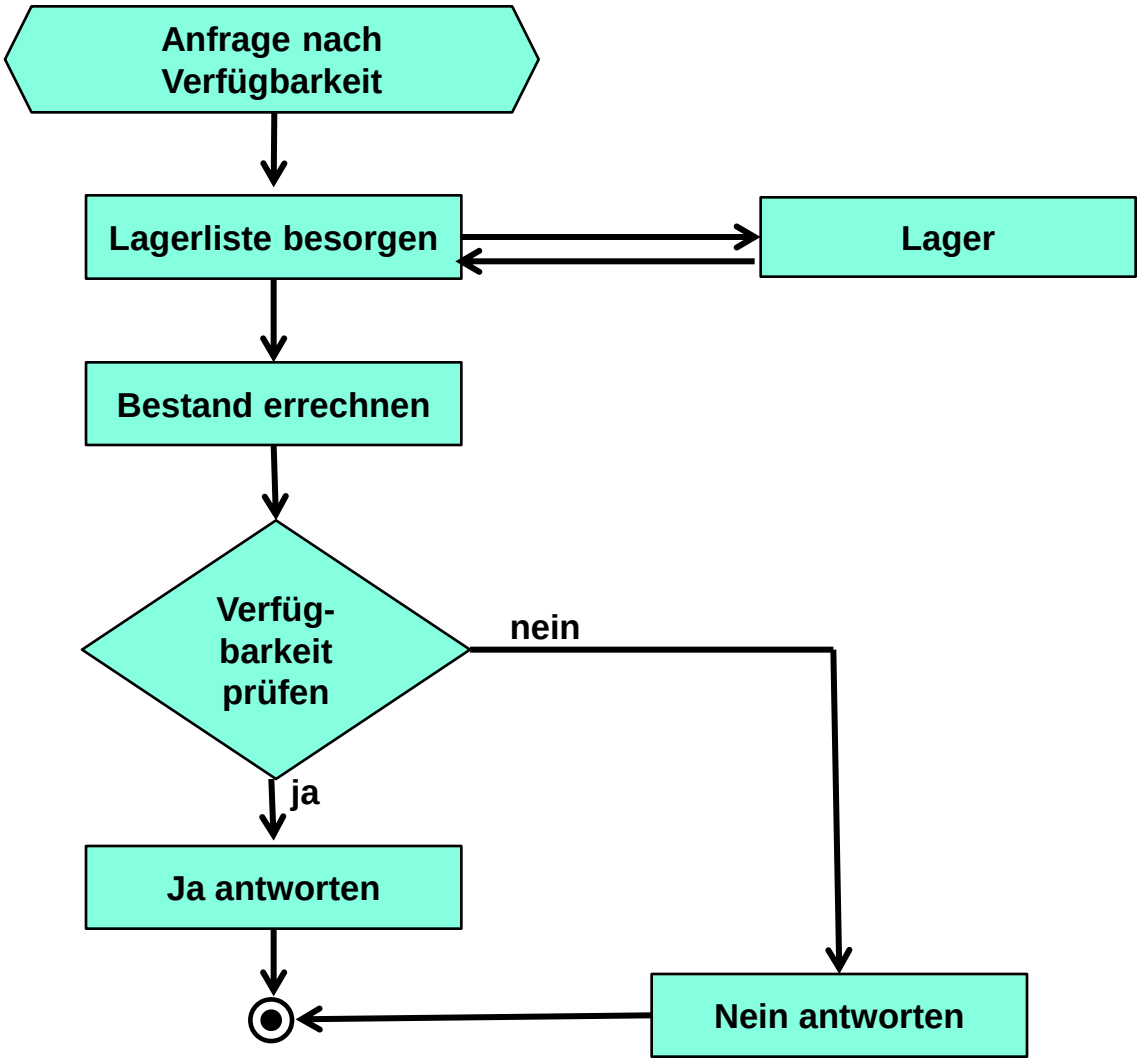
- Sun-Bpel Engine
- ActiveBPEL Engine
  - http://www.activebpel.org
- IBM
- Oracle BPEL Process Manager
  - http://www.oracle.com/technology/products/ias/bpel/index.html
- ActiveWebflow Enterprise
  - http://www.active-endpoints.com/products/activewebflow/awfent/
- Microsoft BizTalk Server
  - http://www.microsoft.com/biztalk/
- FiveSight's Process Execution Engine (PXE)
  - http://www.fivesight.com/pxe.shtml/
- Apache ODE

Übung: AvailabilityProcess

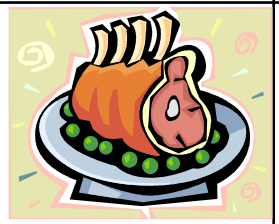
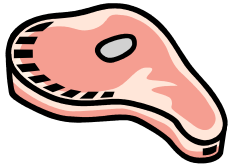
Availability Service Prozess



Availability Prozess



Lager des Schlachthauses

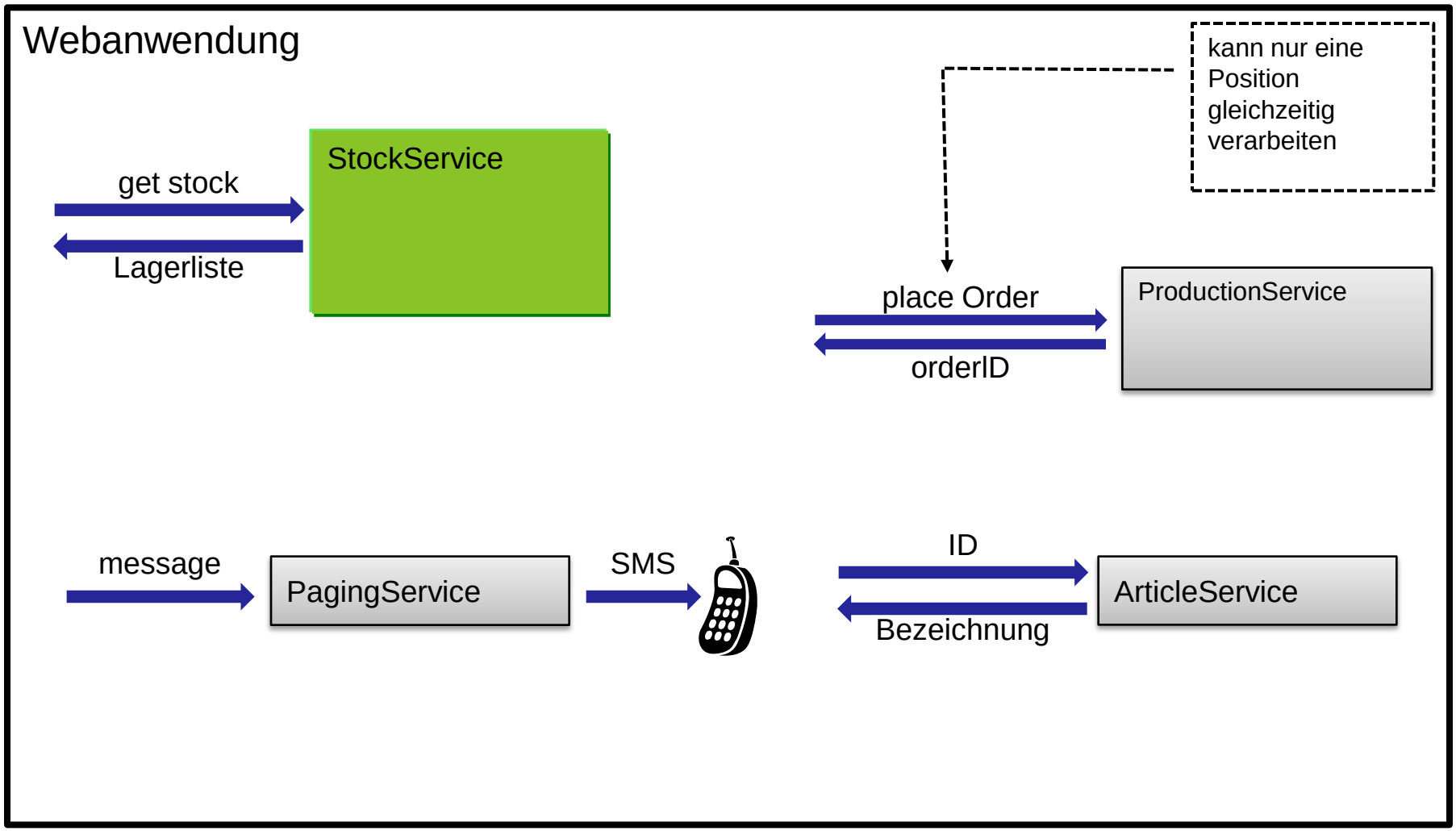
1	2	3  250 Stück
4  50 Stück	5	6  300 Stück
7	8	9  100Stück

① 

② 

③ 

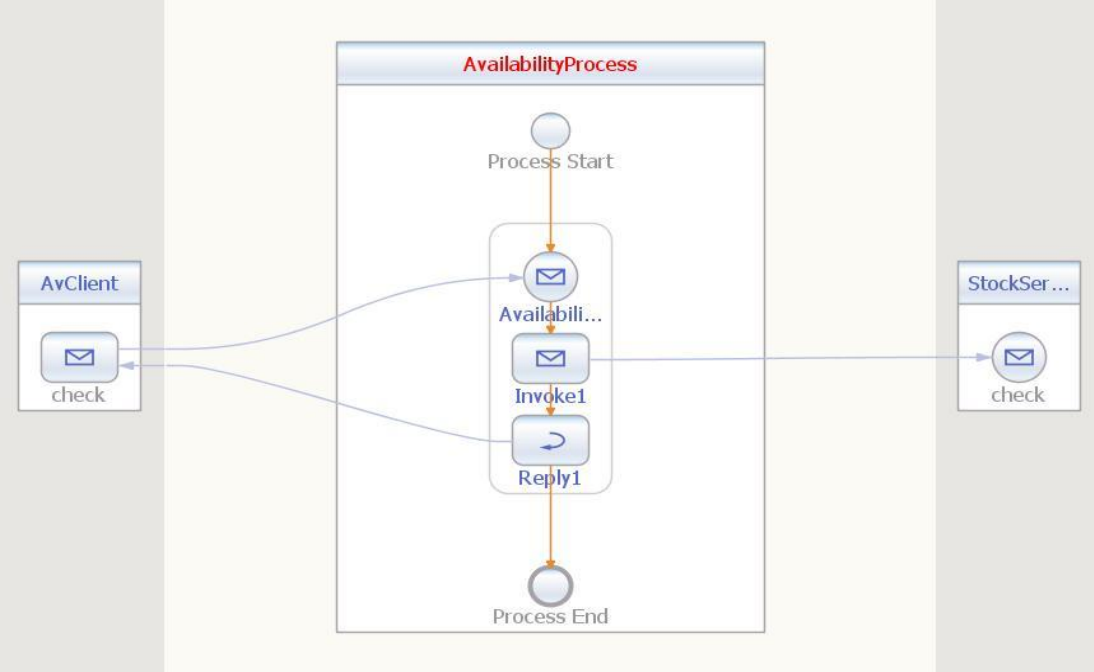
Slaughterhouse Services (Backend)



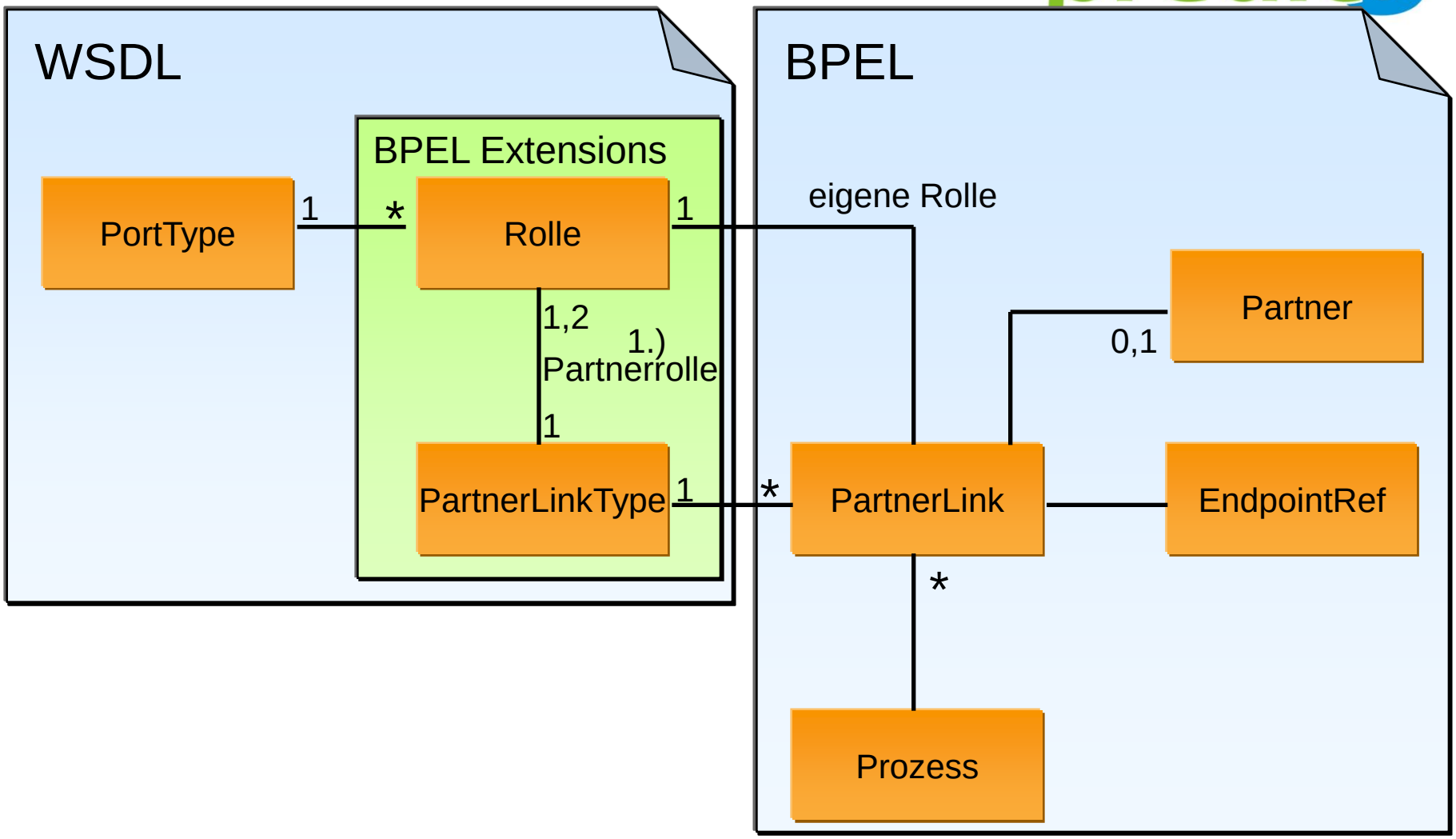
PartnerLinks
oder
Geschäftsbeziehungen

Partner

- Nimmt an Web Service Transaktion teil
- Kommunikation mit Partner über Schnittstellen
  - Ruft Endpoints eines Prozesses auf oder
  - Wird von Prozess aufgerufen



Partner Links



1.) Bei Services, die mit jedem anderen zusammenarbeiten, genügt eine Rolle

Partner Link

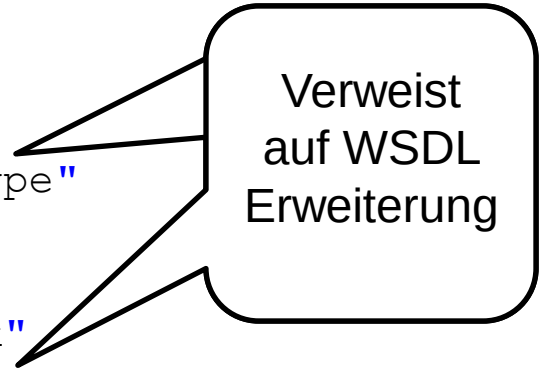
- Definiert am Prozess beteiligte Partei
- „Partner“ müssen Schnittstelle bereitstellen
  - Definition über WSDL
- Jede Aktion verweist indirekt auf den PortType des Partners

```
...
 <partnerLinks>

     <partnerLink name="inboundPartnerLink"
        partnerLinkType="name:ServiceLinkType"
        myRole="inbound" />

     <partnerLink name="outboundPartnerLink"
        partnerLinkType="name:ServiceLinkType"
        partnerRole="outbound" />

 </partnerLinks>
```



BPEL Erweiterungen für WSDL

- PartnerLinkType
  - Verbindet Port-Type mit Partnerlink
  - Wird bei jedem Prozess benötigt
- Property
  - Namens/Wert Paar
  - Wird für Correlation genutzt
- Property Alias
  - Mappt Teil einer Message auf Property
  - Wird für Correlation genutzt

Variablen

Variablen

- Speichern den Status zw. dem Austausch von Nachrichten
- Datentypen
  - WSDL Message Types
  - XML Schema Types (simple, complex)
  - XML Schema Elemente

Variablen

```
<bpel:variables>
  <bpel:variable messageType="ns1:gespraechType"
                 name="Gespraech"/>
  <bpel:variable name="bestellung "
                 type="ns7:Bestellung"/>
  <bpel:variable name="ergebnis"
                 type="xsd:string">
    <bpel:from>''</bpel:from>
  </bpel:variable>
</bpel:variables>
```

Initialisierung von Variablen

```
<bpel:variable name="message"
                    type="xsd:string">
  <bpel:from>'ok!'</bpel:from>
</bpel:variable>
```

Java:

```
String message ="ok!";
```

Initialisierung von Variablen 2

```
<bpel:variable name="message" type="m:message">
  <bpel:from>
      <bpel:literal>
        <m:message
      xmlns:m="http://predic8.com/bpel">
          <m:code>7</m:code>
        </m:message>
      </bpel:literal>
    </bpel:from>
</bpel:variable>
```

Grundlegende Aktivitäten

Grundlegende Aktivitäten

- Receive
- Reply
- Invoke
- Assign
- Throw
- Wait
- Empty

Standard Attribute für Aktivitäten

Können bei jeder Aktivität angegeben werden:

`name="ncname"?` Name für Dokumentation

`joinCondition="bool-expr"?`
`suppressJoinFailure="yes|no"?>`

} Siehe Kapitel Synchronisation

Standard Kindelemente für Aktivitäten

Können jeder Aktivität hinzugefügt werden:

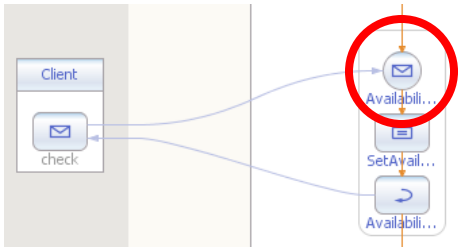
(Bedeutung Siehe Synchronisation)

```
<source linkName="ncname"
         transitionCondition="bool-expr"?>*
<target linkName="ncname"/>*
```

receive - Aktivität

- Wartet bis passende Nachricht von einem Partner eintrifft
- Blockt
- Stellt Partnern Service zur Verfügung
- PortType ist optional (Vorsicht – Inkonsistenz)

```
<bpel:receive createInstance="yes"
              name="BerechneGespraechRequest"
              operation="BerechneGespraech"
              partnerLink="BillingPL"
              portType="ns1:BillingPT"
              variable=" BerechneGespraechRequest "/>
```

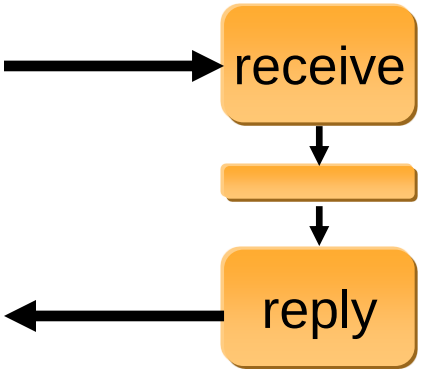


Lebenszyklus eines Prozesses

- Beginnt mit dem Empfang einer Nachricht durch eine receive- oder pick- Aktivität mit
  - createInstance = „yes“
- Normale Beendigung:
  - Die Hauptaktivität und alle Event-Handler terminieren ohne fault
- Abnormale Beendigung:
  - durch exit-Aktivität
  - Fault-Handler auf Prozess Ebene terminiert
  - Fault-Handler auf Prozess-Ebene wirft Fault

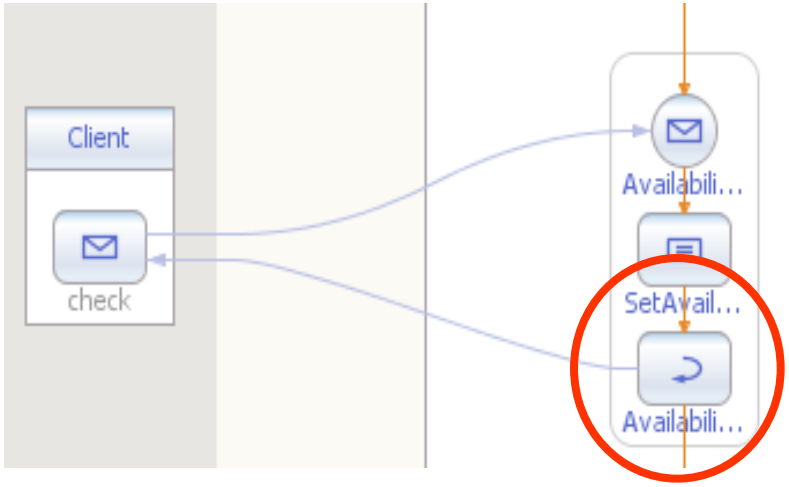
reply - Aktivität

- Sendet eine Antwort auf eine mit receive oder pick empfangene Nachricht
- Nur für synchrone Interaktion
- Bei reveice und reply müssen übereinstimmen
  - PartnerLink
  - PortType
  - Operation
  - ggfls. Correlation Set



reply - Aktivität

```
<bpel:reply operation="BerechneGespraech"
            partnerLink="BillingPL"
            portType="ns1:BillingPT"
            variable="BerechneGespraechResponse"/>
```



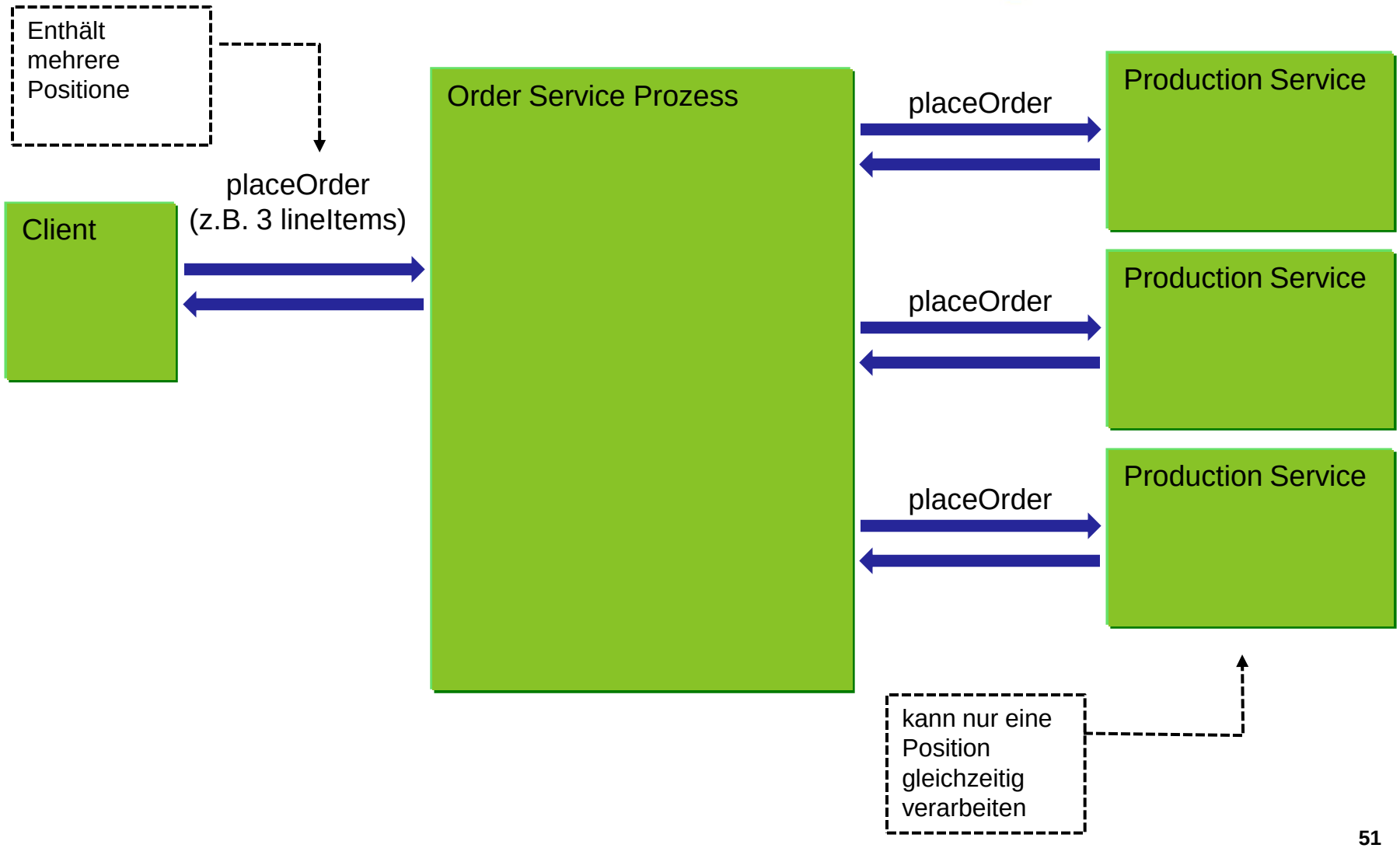
invoke - Aktivität

- Ruft eine one-way oder request-response Operation auf einen PortType eines Partners auf

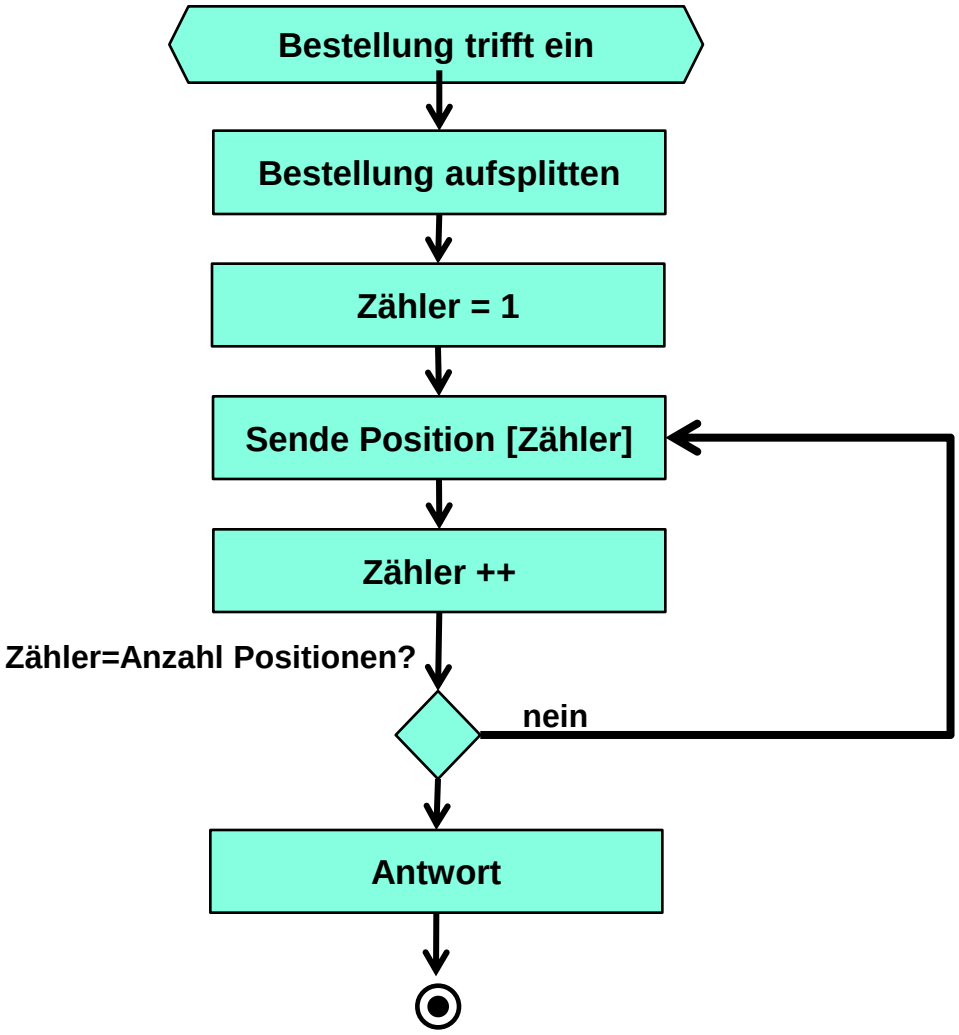
```
<invoke inputVariable="BerechneGespraech“
        operation=“BerechneGespraech“
        outputVariable="BerechneGespraechResponse“
        partnerLink=“BillingPL“
        portType="ns1:BillingPT"/>
```

Übung: OrderProcess

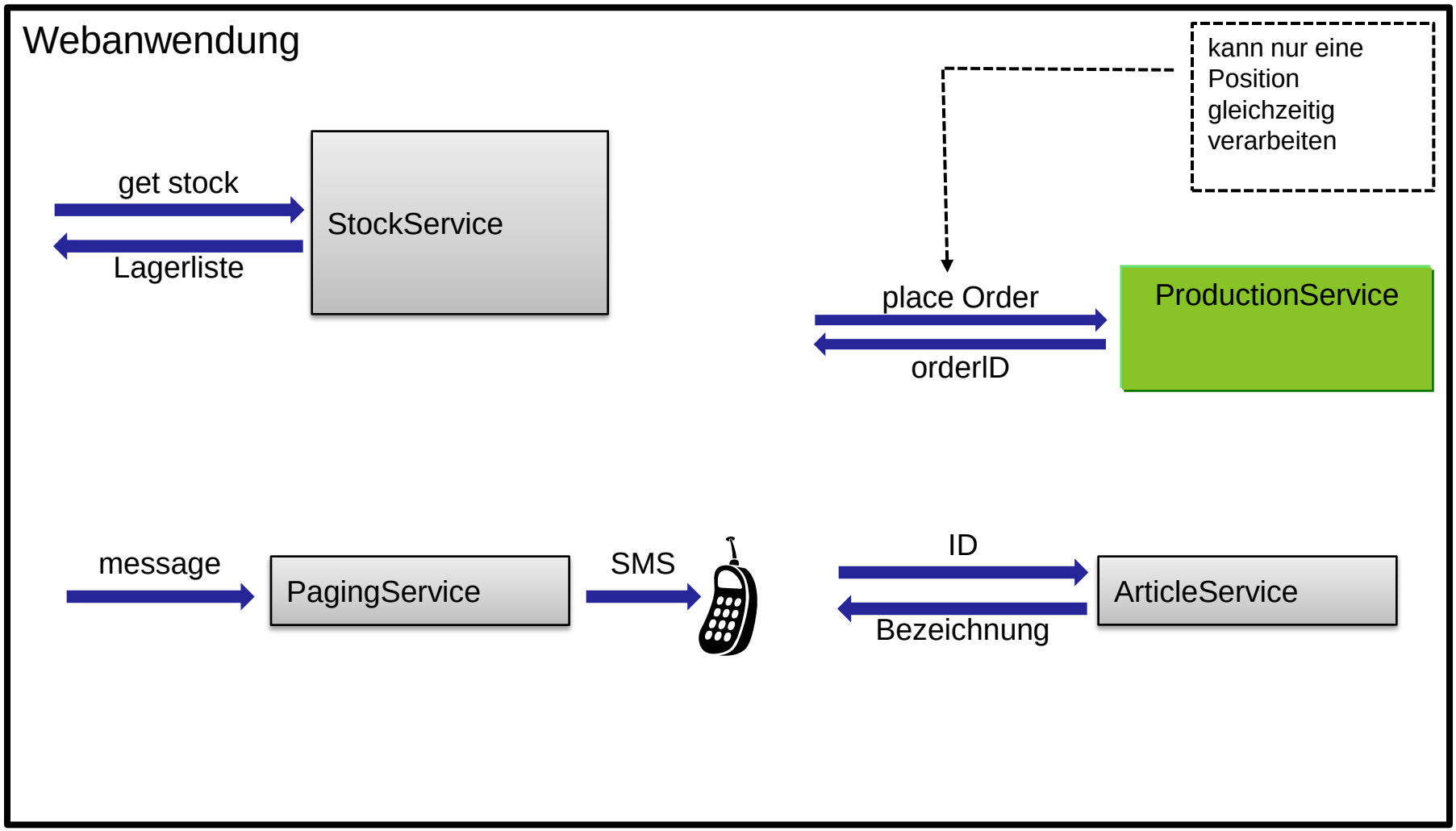
Availability Service Prozess



Aufgabe: Order Prozess



Slaughterhouse Services (Backend)



Strukturierte Aktivitäten

Strukturierte Aktivitäten

- sequence
- switch
- while
- foreach (ab BPEL 2.0)
- pick
- flow

sequence Aktivität

- Führt Kindaktivitäten der Reihe nach aus

switch-Aktivität (nur bis BPEL 1.1)

- Gegensatz zu C:
  - Abbruch nach Verzweigung
  - Kein break notwendig
  - Jedes Case Element enthält eine Bedingung
- In BPEL 2.0 if-Aktivität verwenden

if – Aktivität (ab BPEL 2.0)

```
<bpel:if name="isMondschein">
   <bpel:condition>$requestMessage.Gespraech/zeit
   &gt;=18</bpel:condition>
     ...
   <bpel:else>
     ...
   </bpel:else>
</bpel:if>
```

Vergleich in Java:

```
if(requestMessage.gespräch.zeti > 18){...
} else {...}
```

flow - Aktivität

- Gleichzeitige Ausführung mehrerer Aktivitäten
- Synchronisation
- Ist fertig, wenn alle Kind Aktivitäten fertig sind
- Vergleich: Java erfordert Thread-Erzeugung und Controlle

while Aktivität

- Schleife mit Prüfung am Anfang
- Führt Kindaktivität wiederholt aus
- Bricht ab, wenn Bedingung falsch ist

forEach – Aktivität ab BPEL 2.0

- Iteriert über Aktivitäten
- Kann Aktivitäten parallel ausführen
  - parallel=“yes“
- Reihenfolge ist nicht festgelegt

wait - Aktivität

- Wartet auf
  - Zeitspanne
  - Zeitpunkt

```
<wait until="'2000-04-07T12:00-01:00'"/>
<wait for="'P7DT5H'"/>
```

Fehlerbehandlung

Fehlerbehandlung

- Beim Auftritt eines Fehlers können bereits mehrere ACID Transaktionen in Backends abgeschlossen sein
- Ursachen für Fehler
  - Invoke bekommt Fault zurück
  - Throw Aktivität
  - Standard Faults

Fault Handler

- Was wird bei einem Fehler getan
- Wird durch catch-Aktivität realisiert
- Tritt ein Fault auf, wird keine Compensation durchgeführt

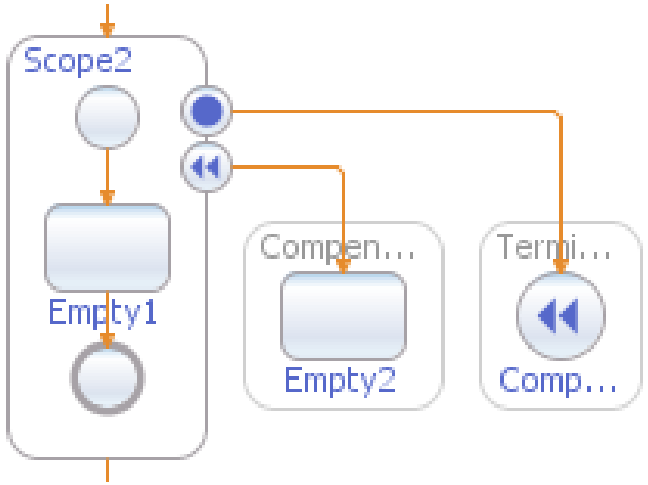
```
<bpel:faultHandlers>
      <bpel:catch faultName="ns1:ungueltigerTarif">
         ...
      </bpel:catch>
      <bpel:catchAll>
         ...
      </bpel:catchAll>
</bpel:faultHandlers>
```

Default Fault Handler

```
<catchAll>
  <sequence>
    <compensate />
    <rethrow />
  </sequence>
</catchAll>
```

Handler für forced Termination

- Wird beim Abbruch eines Scopes aufgerufen z.B. durch Fault in übergeordneten Scope
- Kann keinen Fault werfen (rethrow)
- Kann Compensation auslösen
- Für Aufräumarbeiten



Default Termination Handler

```
<teminationHandler>
  <compensate name="compensate1"/>
</terminationHandler>
```

Event Handler

- Startet Aktivitäten bei Ereignissen und Alarmen
- Ein Compensation Handler darf von einem Event-Handler nicht aufgerufen werden

throw Aktivität

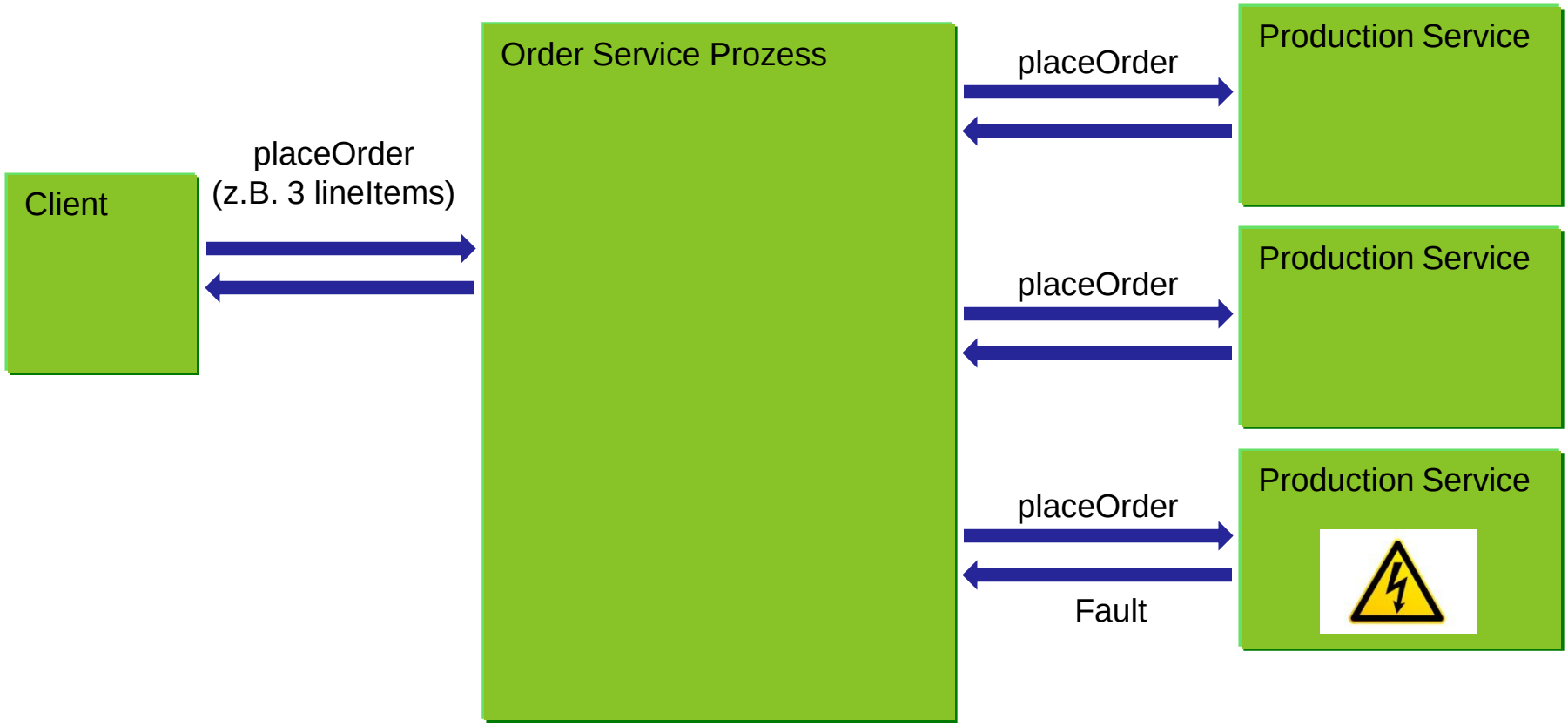
- Wirft einen „Fault“
- Analog zu Java:

```
throw new LimitExceededException();
```

```
<throw faultName="ns1:limitExceeded"/>
```

Übung: Fault Handling

Availability Service Prozess



empty Aktivität

- Macht nichts
- Nützlich beim Debuggen
- Verhinder Fehler wenn Aktivität erwartet wird

repeatUntil (ab BPEL 2.0)

- Kindaktivität wird geführt, bis Beendigung wahr wird
- Bedingung wird nach der Ausführung der Kindaktivität getestet
  - Kinderaktivität wird min. 1 mal ausgeführt

Spezielle Aktivitäten

- scope
- compensate
- terminate
- validate (ab BPEL 2.0)

scope - Aktivität

- Vergleichbar mit { } in Java
- Lokale Umgebung für
  - PartnerLinks, MessageExchanges, Variablen, CorrelationSets, FaultHandlers, Compensation Handlers, TerminationHandlers, EventHandlers
- Kann beinhalten
  - Fault-, Event- und Compensation Handler
  - Variablen
  - Correlation Sets

scope – Aktivität Beispiele

```
<bpel:scope>
  <bpel:faultHandlers>
    <bpel:catchAll>
      ...
    </bpel:catchAll>
  </bpel:faultHandlers>
  <bpel:invoke ... />
</bpel:scope>
```

Serializable Scopes

- Dürfen keine weiteren Scopes enthalten
  - Blätter
- Zugriff auf gemeinsame Variablen wird synchronisiert
- Analog zum Isolation Level „Serializable“ in Datenbanken

BPEL 1.1

```
<scope variableAccessSerialzeable="yes">
   ...
</scope>
```

BPEL 2.0

```
<scope isolated="yes">
   ...
</scope>
```

exit - Aktivität

- Beendet den Business Prozess

```
<exit/>
```

validate Aktivität (ab BPEL 2.0)

- Validiert eine Variable gegen XML oder WSDL Definitionen.

```
<validate variables="Handyvertrag"/>
```

assign Aktivität

- Weist einer Variablen einen Wert zu
- Kopiert Werte

Assign mit XPath Ausdruck

```
<bpel:assign name="...">
  <bpel:copy>
    <bpel:from>normalize-space($token)</bpel:from>
    <bpel:to variable="tag"/>
  </bpel:copy>
</bpel:assign>
```

Assign mit Konstante

```
<bpel:assign>
  <bpel:copy>
    <bpel:from>
      <bpel:literal>
        <foo:bar xmlns:foo="urn:foo">
          <foo:baz/>
        </foo:bar>
      </bpel:literal>
    </bpel:from>
    <bpel:to part="foo" variable="bar"/>
  <bpel:copy>
</bpel:assign>
```

Kompensation

Saga

- Von Hector Gareira-Molina und Kenneth Salem (1987)
- Reihe von kleinen Transaktionen, die in Verbindung stehen
- Koordinator triggert Aktionen zur Kompensation

Compensation

- Von der Anwendung kontrollierte Fehlerbehandlung
  - Das heisst Fehlerbehandlung muss programmiert werden

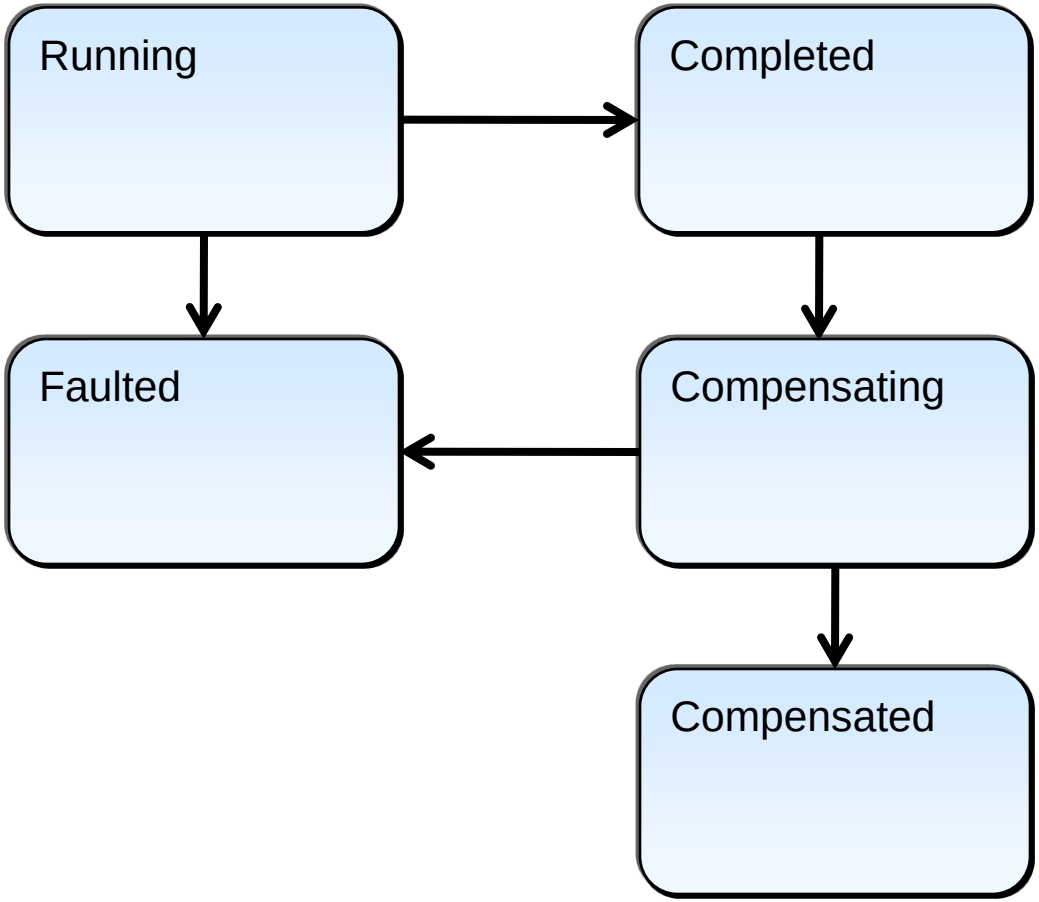
Vergleich ACID Transaktion und Kompensation

ACID Transaktion	Kompensation
Rollback	Ausführen von „Gegentransaktionen“
Rollback macht Transaktionen ungeschehen	Kompensation kann ganz andere Aktionen durchführen z.B. Stornogebühren
Für kurze Transaktionen	Für lange Prozesse geeignet
Atomic	-
Isoliert	-
Resourcen müssen transaktional sein	Nicht transaktionale Resourcen können teilnehmen (z.B. Postversand)

Cross-Service Transactions

- Partner können Antwort verzögern und so Resourcen unnötig blockieren
- Technisch möglich ist XA, Atomic Transactions

Stati eines Scopes



Scope Snapshot

- Zustand des Scopes und aller einschließenden Scopes
- Zustand der Variablen, PartnerLinks, CorrelationSets und MessageExchanges
- Wird von einem ausgeführten, aber noch nicht kompensierten Scope erstellt

Compensation Handler

- Macht die Folgenen eines bereits abgeschlossenen Prozesses „rückgängig“.
- Lebt in einer Snapshot-world
  - Kann keine vom Prozess benutzten Veriablen ändern
  - Bekommt eingefrorenen Snapshot vom Scope nach dessen Beendigung
- Wird von compensate-Aktivität aufgerufen
- Wird nur nach normaler Beendigung des Scopes in dem der Handler definiert ist aufgerufen

compensate-Aktivität

- Ruft einen Compensation-Handler auf
- Kann nur in Fault- und Compensation Handlern ausgeführt werden

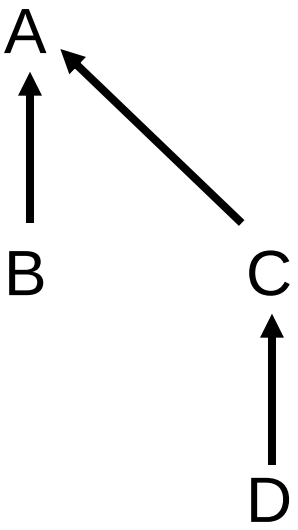
Kompensation eines Scopes:

```
<compensate scope="..."/>
```

Default Kompensation

```
<compensate/>
```

Reihenfolge Compensation



Compensation Order:

B, D, C, A oder

D, B, C, A

→ Abhängigkeit im Kontrollfluß

Compensation Reihenfolge

- In Schleifen
  - Compensation Handler werden für die ausgeführte Iterationen in umgekehrter Reihenfolge ausgeführt
- Default
  - Compensation Handler für eingebettete Scopes werden umgekehrt zur Reihenfolge der Durchführung ausgeführt.

Default Compensation Handler

```
<compensationHandler>
  <compensate />
</compensationHandler>
```

compensateScope (ab BPEL 2.0)

- Start Kompensation für einen eingebetteten Scope
- Scope muss erfolgreich ausgeführt worden sein
- compensateScope kann nur innerhalb von einem fault-, compensation- oder termination- Handler verwendet werden

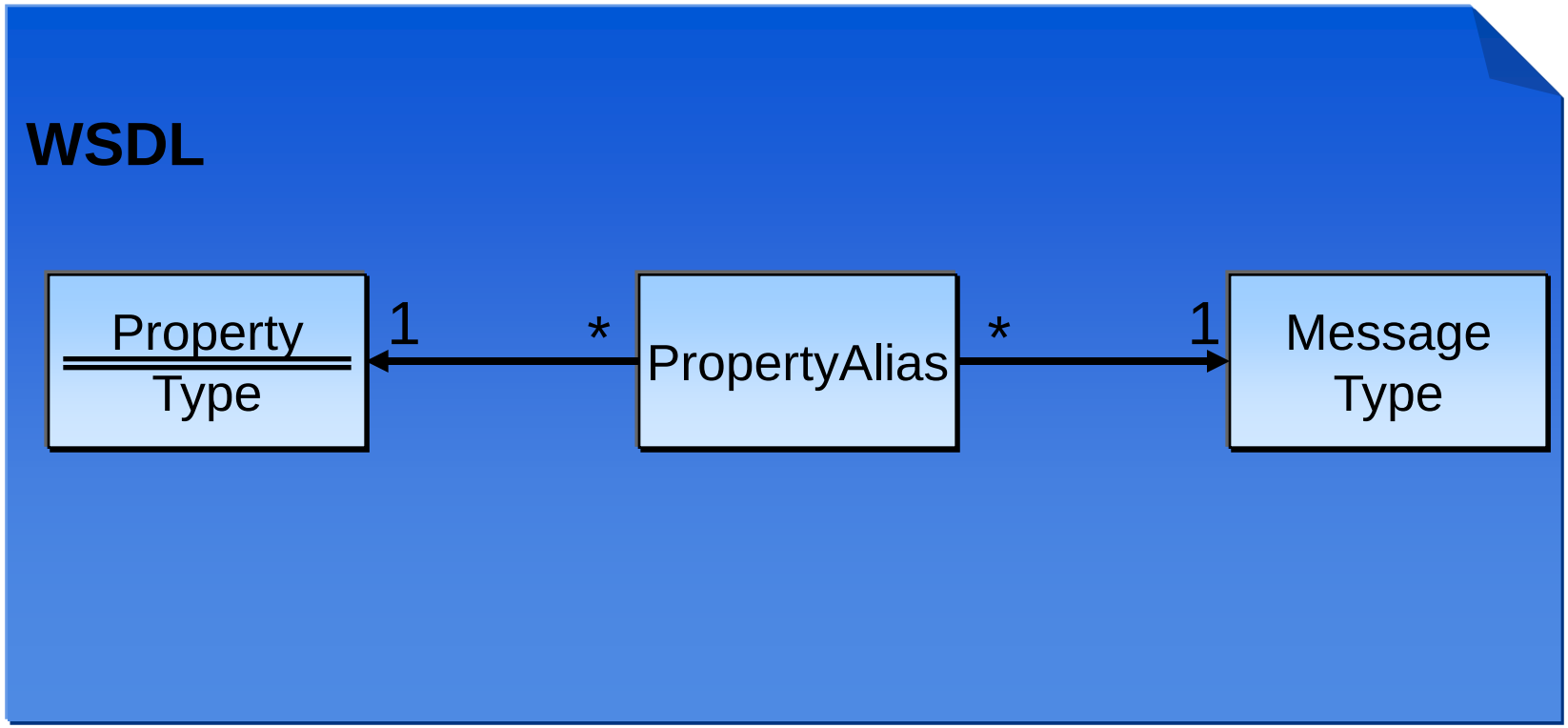
Übung:
Compensation

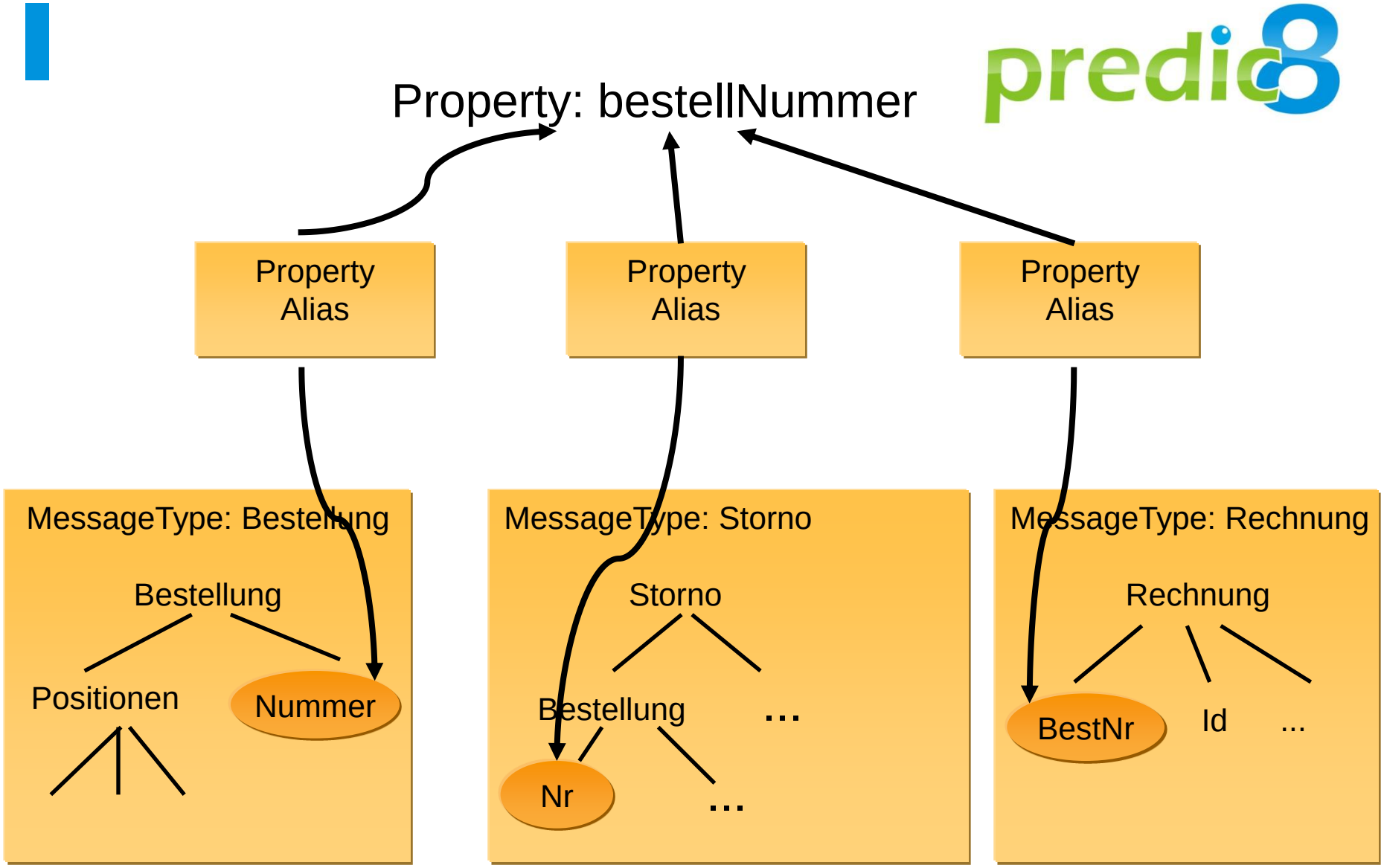
Korrelation - oder was zusammengehört

MessageProperties

- Nützlich in abstrakten Prozessen um zu zeigen, wie Teile einer Nachricht behandelt werden
- Feld innerhalb einer Nachricht, das durch eine Abfrage identifiziert wird
- Sind eine Spezialisierung der allgemeineren „Variable Properties“

Property Alias

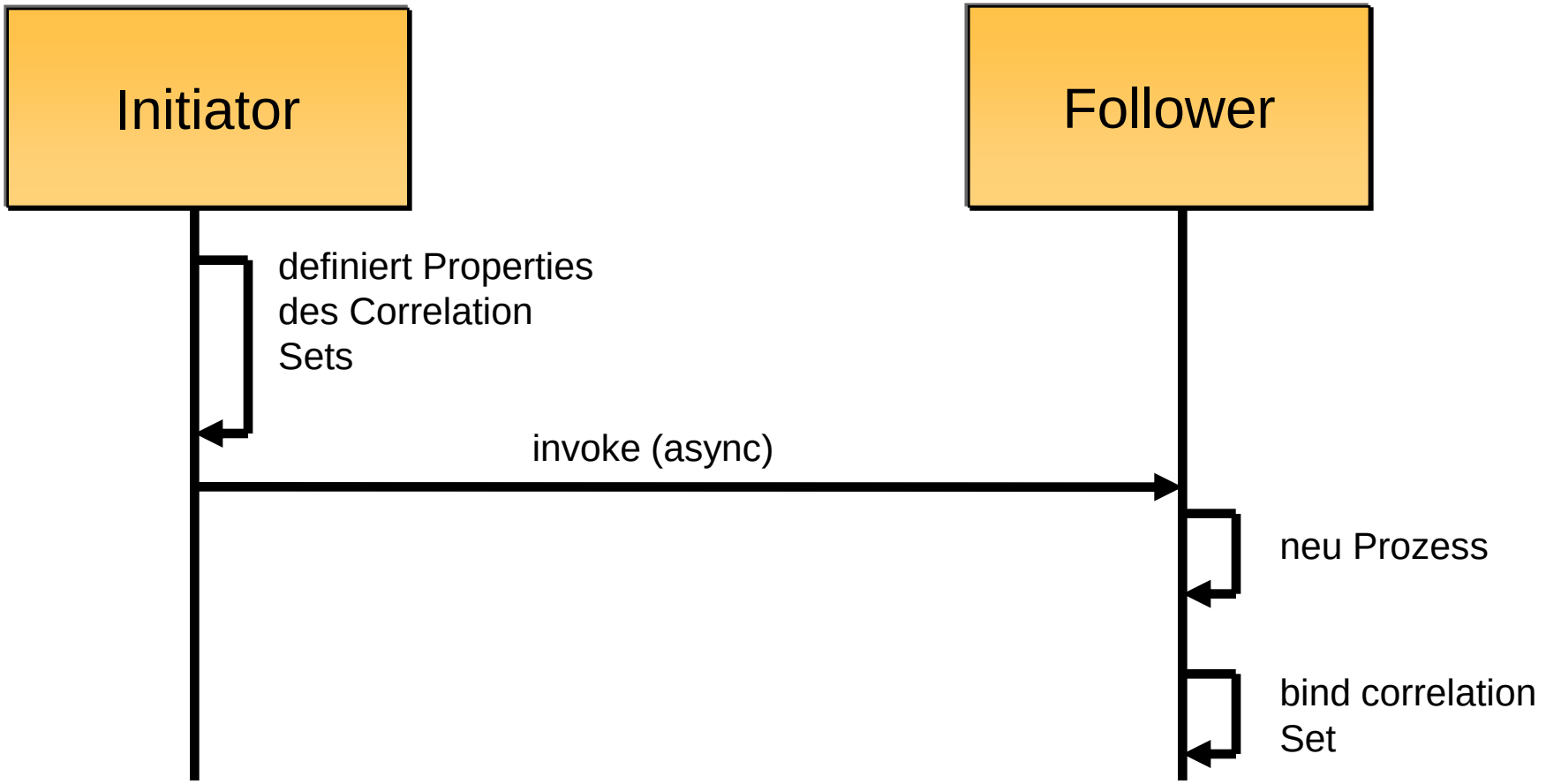




Correlation

- Beispiel: Bestell- und Rechnungsnummern
- Eine Nachricht kann Bestandteil mehrerer Konversationen sein
- Correlation Data
  - Im SOAP Header
  - In der Payload

Correlation



Correlation Set

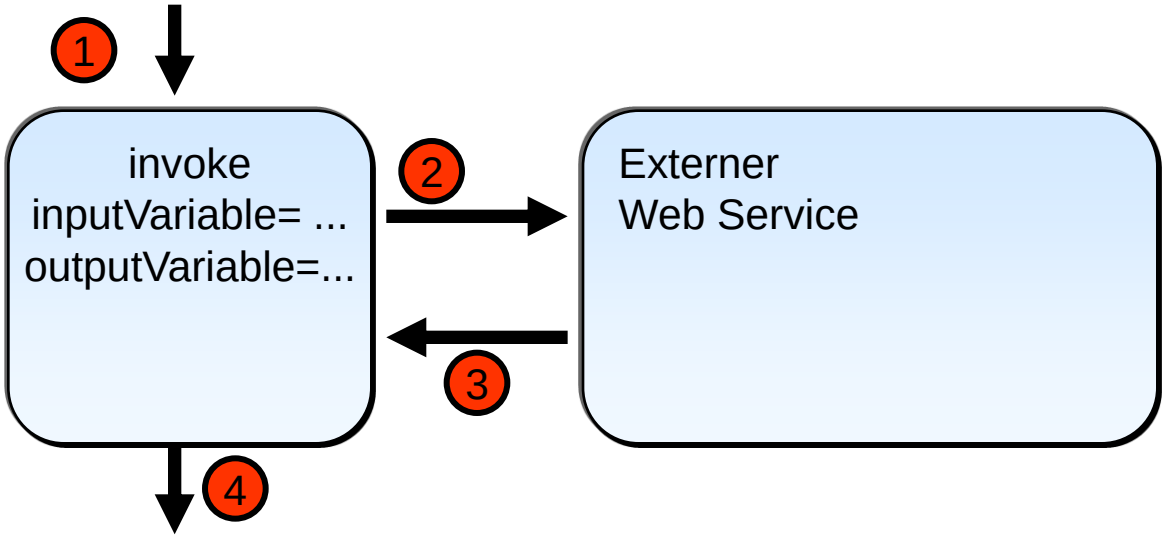
- Menge von Properties, die von Nachrichten einer korrelierten Gruppe geteilt werden
- Sind mit Scope assoziiert
  - lokal oder global
- Binding erfolgt beim Senden oder Empfangen von Nachrichten
- Haben einen Namen
- Id für eine Prozess Instanz

Übung: Correlation

Message Exchange Patterns

Synchroner Aufruf

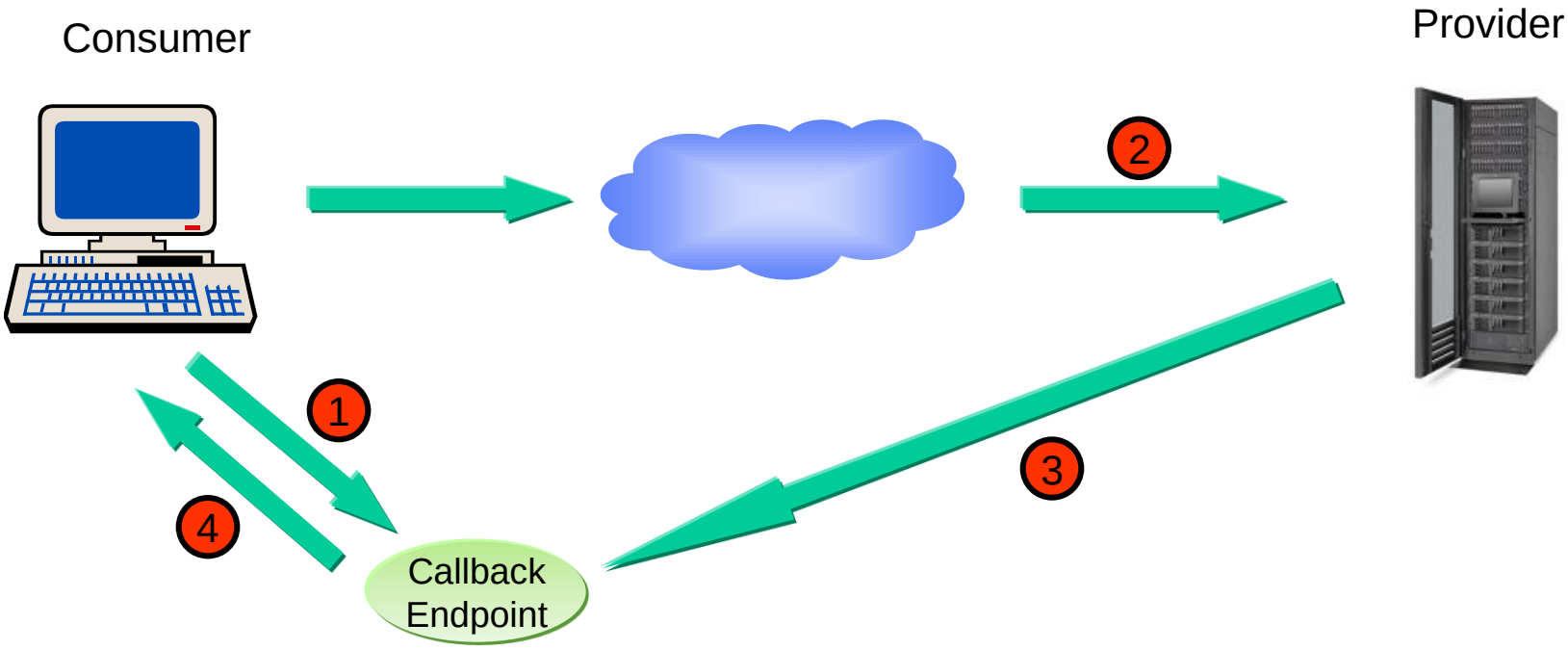
- Request/Response
- Invoke-Aktivität mit input- und output-Variablen
- Blockiert
- Eine Rolle im PartnerLinkType



Asynchrone Prozesse

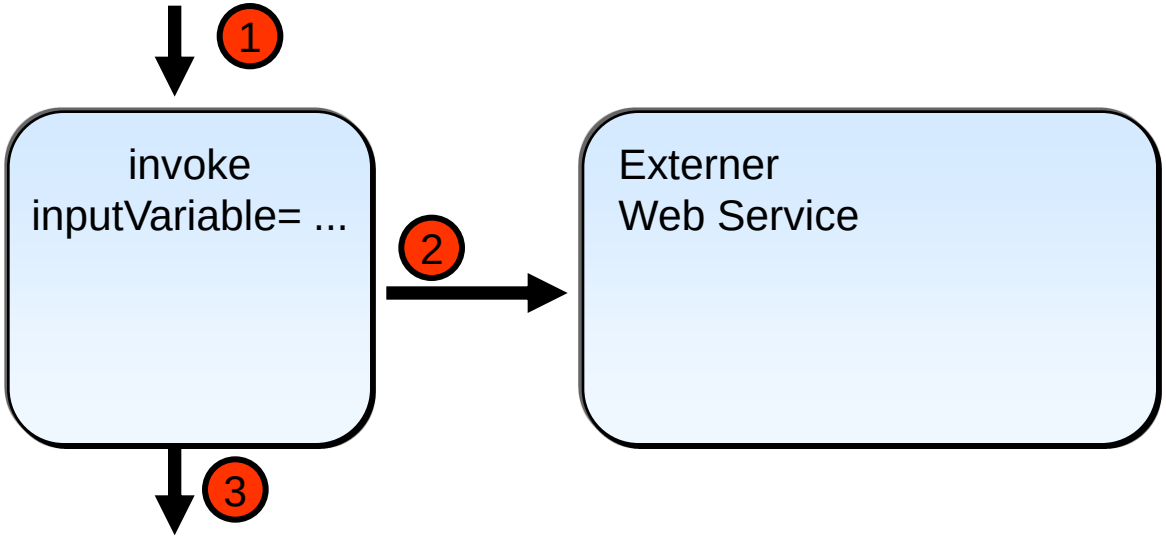
- Oft Langläufer
- Sender wartet nicht auf Antwort
- Empfänger kann über Callback Sender informieren (optional)

Callback



Asynchroner Aufruf (One-Way)

- Invoke Aktivität mit input – aber ohne output-Variablen
- Externer Web Service kann keinen Fault zurücksenden



Asynchrone Prozesse

- 1-2 Rollen in PartnerLinkType (bei späterem Callback)
- Zweite Rolle für Callback

Event Handling

Event Handler

- Für das Senden von Nachrichten an einen bereit laufenden Prozess (onMessage)
- Für das Anstoßen von Ereignissen nach Ablauf einer Zeitspanne (onAlarm)
- Wird bei normaler Prozessausführung aufgeführt

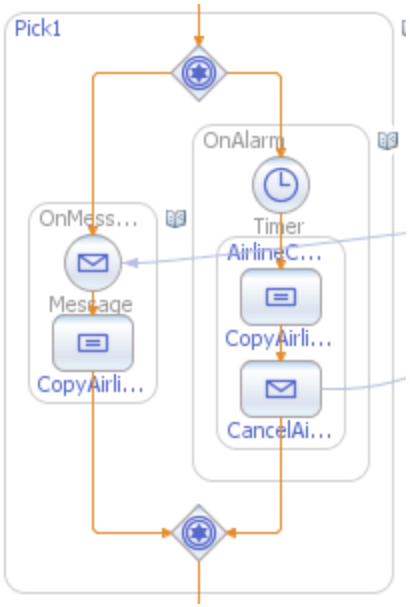
Event Handler Aktivierung

- Nach Empfang der initialen Message
- Start des Scopes in welcehm der Event Handler definiert ist

Übung:
Event Handling

pick - Aktivität

- Wartet auf Nachricht, Timeout oder Alarm
- Das erste Ereignis gewinnt
- Nur eine der Aktivitäten in einer pick-Aktivität wird ausgeführt
- Min. 1 onMessage Element muss angegeben werden



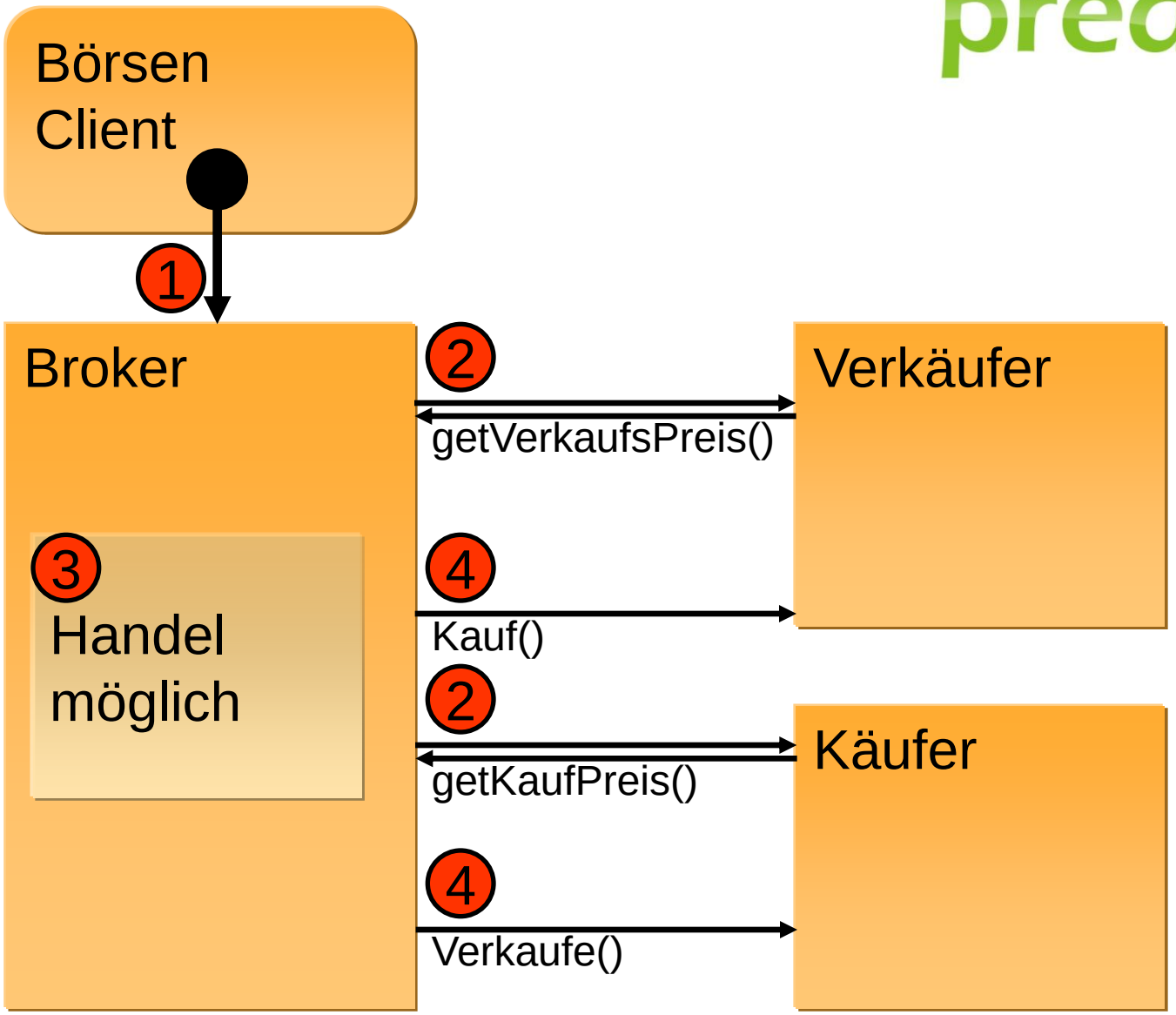
Pick-Aktivität

```
<pick name="Pick1">
  <onMessage partnerLink="Airline" ... >
    <correlations>
      <correlation set="ItineraryCorrelator" initiate="no">      </correlation>
    </correlations>
    <assign name="CopyAirlineReservation">
      <copy>
        <from variable="AirlineReservedIn" part="itinerary"/>
        <to variable="ItineraryOut" part="itinerary"/>
      </copy>
    </assign>
  </onMessage>
  <onAlarm>
    <for>'P0Y0M0DT0H0M20S'</for>
    <sequence name="AirlineCancelSequence">
      <assign name="CopyAirlineCancellation">
        <copy>
          <from variable="CancelAirlineOut"/>
          <to variable="CancelAirlineOut"/>
        </copy>
      </assign>
      <invoke name="CancelAirline" ... />
    </sequence>
  </onAlarm>
</pick>
```

Übung: Pick

Synchronisation

Nebenläufige Ausführung



Link

- Analog zu Signal bei Betriebsystemen
- Hat einen Namen
- Links dürfen keine zyklischen Graphen bilden

```
<flow>
  <links>
    <link name="hausaufgaben-fertig"/>
    <link name="rasen-gemaeht"/>
  </links>
  <receive name="kinogehen" ...>
    <targets>
         <joinCondition>
            $hausaufgaben-fertig and $rasen-gemaeht
         </joinCondition>
         <target linkName="hausaufgaben-fertig"/>
         <target linkName="rasen-gemaeht"/>
      </targets>
   </receive>
  <invoke name="rasenmaehen" ...>
    <source linkName="rasen-gemaeht"/>
  </invoke>
  <invoke name="hausaufgaben" ...>
    <source linkName="hausaufgaben-fertig"/>
  </invoke>
</flow>
```

Abhängigkeit

```
<invoke...    name="A">
    <target linkName="prepdone"/>
</invoke>

<sequence name="B">
   <source linkName="prepdone"/>
</sequence>
```

- A hat Synchroinisations-Abhängigkeit zu B

Link Status

- negative, positive
- Verhindert Deadlock
- Wenn entschieden werden kann, dass eine Aktivität nicht ausgeführt wird, werden die zugehörigen Links Stati negativ
  - z.B. Zweig einer Switch Anweisung

Übergangs - Bedingungen

- Falls wahr, wird der zugehörige Link Status positiv

join - Condition

- Nachdem der Status aller „incoming“ Links bestimmt ist, wird die join-Condition ausgewertet
- Wenn join-Condition falsch ist, wird ein joinFailure geworfen
- Default:
  - Wahr, wenn ein incoming link positiv ist

Dead-Path-Elimination DPE

- Kann eine Aktivität aufgrund eines Join-Failure nicht ausgeführt werden, so wird der Link-Status für jeden outgoing Link auf negative gesetzt.
- Negativer Link Status wird über das Netzwerk propagiert, bis eine joinCondition wahr wird

WS-BPEL 2.0

Neues in BPEL 2.0 (11. April 2007)

- Neue Aktivitäten
  - if-then-else,repeat Until, validate, forEach, extensionActivity
- Initialisierung von Variablen
- XSLT Transformation
- XPath Zugriff auf Daten von Variablen
- Klärung abstrakter Prozesse
- Länge der Spezifikation
  - 1.1: 136 Seiten
  - 2.0: 282 Seiten

Erweiterung durch Skriptsprachen

Programmierung „in the Small“

- Java
  - BPELJ und JSR-207

Business Activity Monitoring

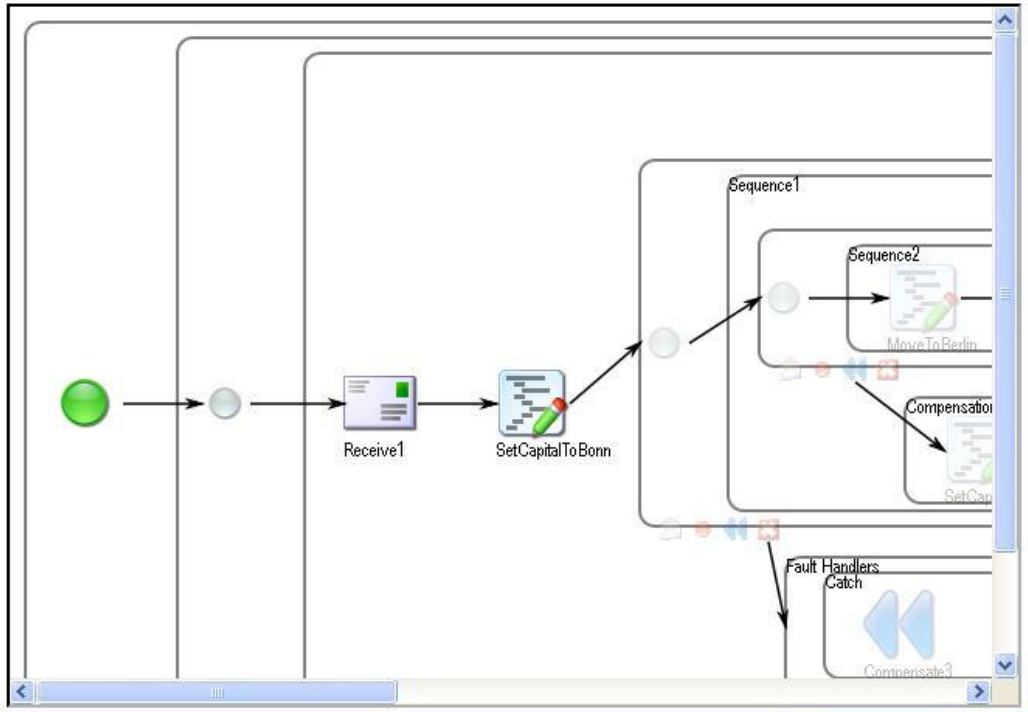
BAM für sun-bpel-engine

Business Process Monitor

Application: BPEL-BeispieleApp-BP

Model: CompensateBeispiel

Start Time	End Time	Status
2009-01-19 14:41:20.64	2009-01-19 14:41:27.859	COMPLETED
2009-01-19 14:41:40.281	2009-01-19 14:41:40.406	COMPLETED
2009-01-19 14:45:12.953	2009-01-19 14:45:13.906	COMPLETED
2009-01-19 15:14:42.437		RUNNING

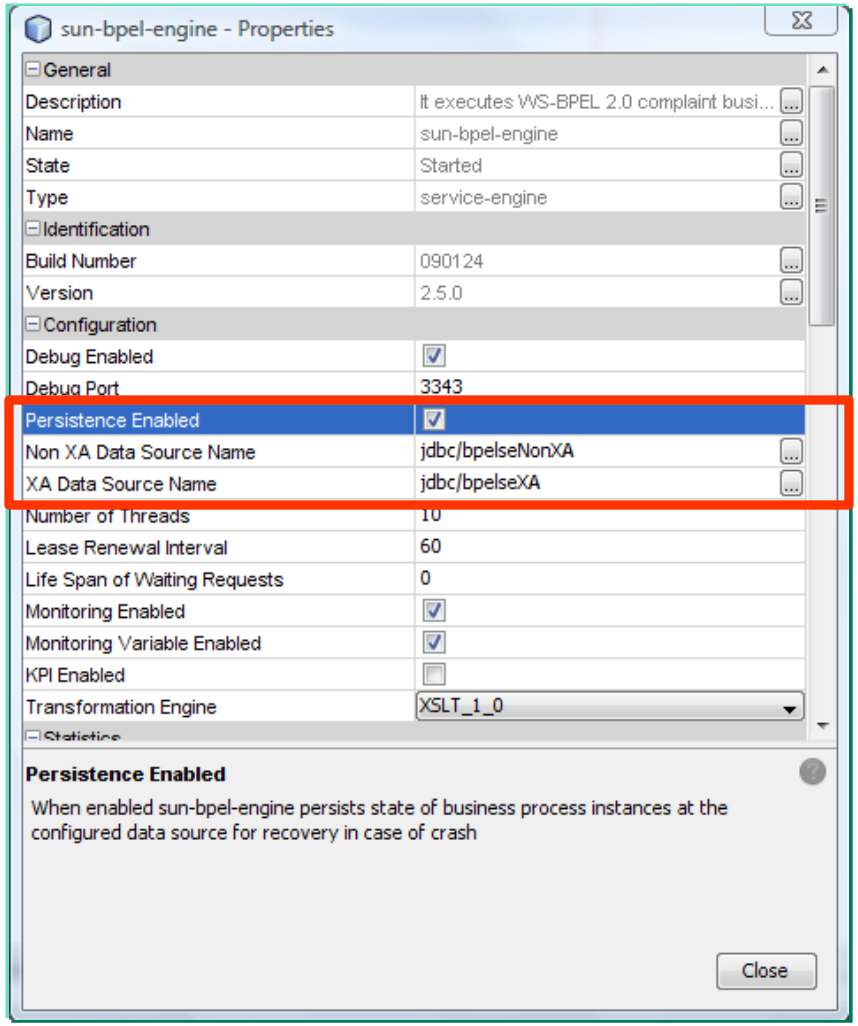


Variable	Value
CompensateTestWSDLOperationIn	
capital	Bonn

Übung:
BPEL Monitoring

Sonstiges

Process Persistence



Document Linking

- import
  - Deklariert Abhängigkeit zu externem Schema oder WSDL
  - URIs für WSDL uznd Schema sind für importType vordefiniert

```
<import namespace="<<URI>>"
        location="<<URL>>"
        importType="<<URI>>" />
```

BPEL XPath Extension Functions

```
bpws:getVariableProperty('varName','propName')
bpws:getLinkStatus('linkName')
```

Nur in ausführbaren Prozessen

```
bpws:getVariableData('varName', 'partName', 'locationPath')
```

Erweiterbarkeit

- Jedes Element kann mit NS qualifizierten Attributen erweitert werden
- Erweiterungen sind mandatory oder optional
  - mandatory: Implementierungen, die die Erweiterung nicht kennen, müssen den Prozess ablehnen
  - optional: Erweiterung kann ignoriert werden
- BPEL 2.0: extensionActivity

WS-BPEL und WS-I Basic Profile

- Implementationen müssen so konfiguriert werden können, dass Prozesse in WS-I Basic Profile 1.1 konformen Interaktionen teilnehmen können
- Implementationen können es erlauben, die Basic Profile 1.1 Konfiguration zu deaktivieren

Fazit

Kräfte die für BPEL sprechen

- Kommunikation zwischen Web Services
- Erstellen von zusammengesetzten Prozessen
- Asynchrone Kommunikation
- Langläufer
- Timing spielt eine Rolle
- Nebenläufigkeit
- Unzuverlässigkeit von partnern, Netzwerken, ...

Glossar

Bullshit Bingo I

forkAktivity	Aktivität mit mehreren „outgoing“ Links, die nach der Ausführung eine Verzweigung (Fork) verursacht
incoming Link	Link, der als „Target“ für eine Aktivität angegeben wurde
outgoingLink	Link, der als „Source“ für eine Aktivität angegeben wurde
initiator	Startet eine Konversation durch das Initialisieren eines Korrelation Sets
follower	Knoten, der eine Nachricht mit Daten zur Korrelation erhält
inbound message activity IMP	receive, onMessage oder onEvent

Bullshit Bingo II

Enclosing Element	Element, welches als Textnode eine Querry oder Expression enthält
FCT	Fault, Compensation, Termination