Seminar

Service Orientierte Architektur

Geschäftsprozessmodellierung mit BPEL4WS:

Aufbau und Beispiel

Agenda

(1) Überblick
(2) Der Geschäftsprozess
(3) Konzept und geschichtliche Entwicklung
(4) BPEL4WS als Teil der Webservices-Architektur
(5) Beschreibung des BPEL4WS-Service per WSDL
(6) Die Sprache BPEL4WS im Detail
(7) BPEL4WS in der Anwendung
    (1) Vorführung eines Beispiels
(8) Ausblick: Erweiterungen zu BPEL4WS
(9) Fazit

1. Überblick

- Problemstellung:
  - Unternehmen: Enterprise Application Integration (EAI)
  - Komplexe Geschäftsabläufe
  - Hohe Anforderungen
  - Vorhandene Ressourcen optimal nutzen
  - Standards finden
- Lösungsansatz: BPEL4WS:

  Business Process Execution Language for Web Services
  - XML-Sprache
  - Orchestrierung (Kombination / Zusammenfassung) von Webservices
  - Zentrale Verwaltung der Geschäftslogik
  - B2B, Veränderung der Geschäftslogik

2. Der Geschäftsprozess

Geschäftsprozess

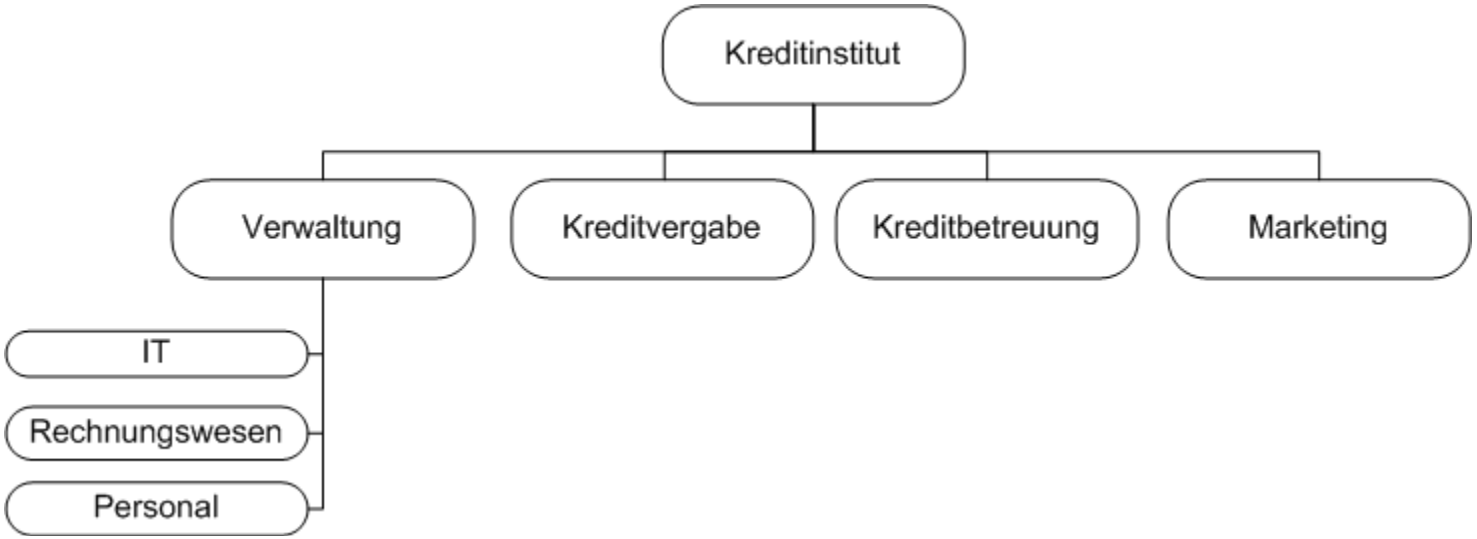
✓eine Abfolge von Vorgängen zur Erreichung eines Resultates im betriebswirtschaftlichen Sinne

✓Kann Unterprozesse enthalten (bzw. Teil eines höheren sein)

✓Kann Vorgänger und Nachfolger besitzen

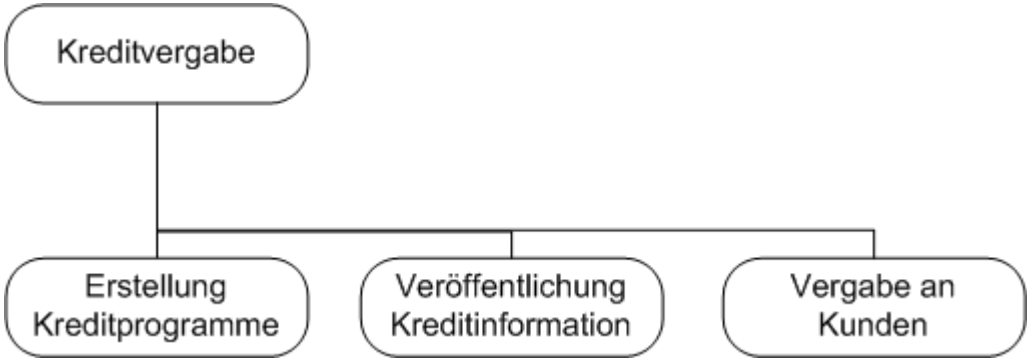
Beispiel

Ein Kreditinstitut:



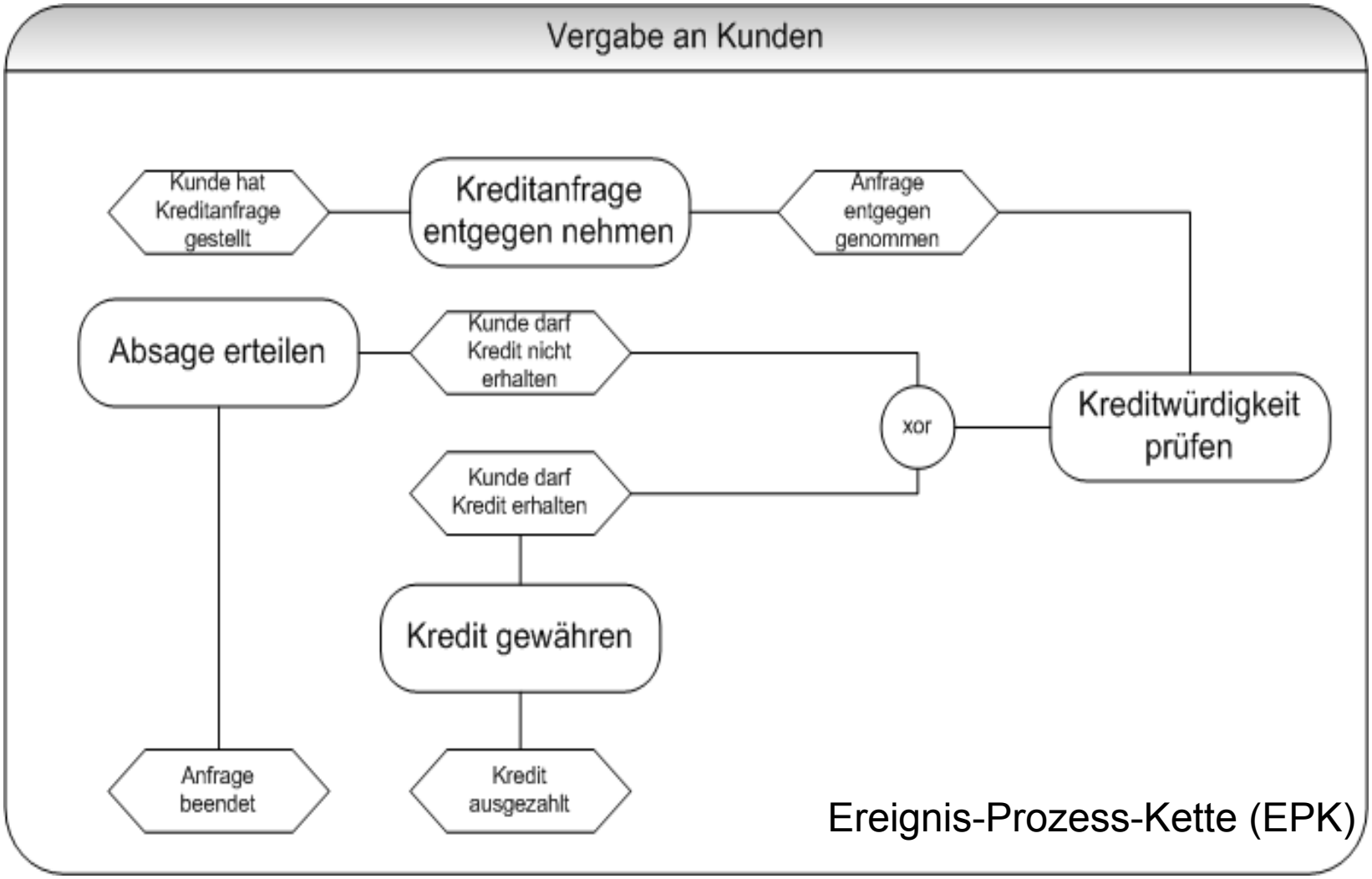
Ausschnitt eines Funktionsbaums

Aufspaltung der Kreditvergabe



Wertschöpfungskette

Kreditvergabe an Kunden im Detail



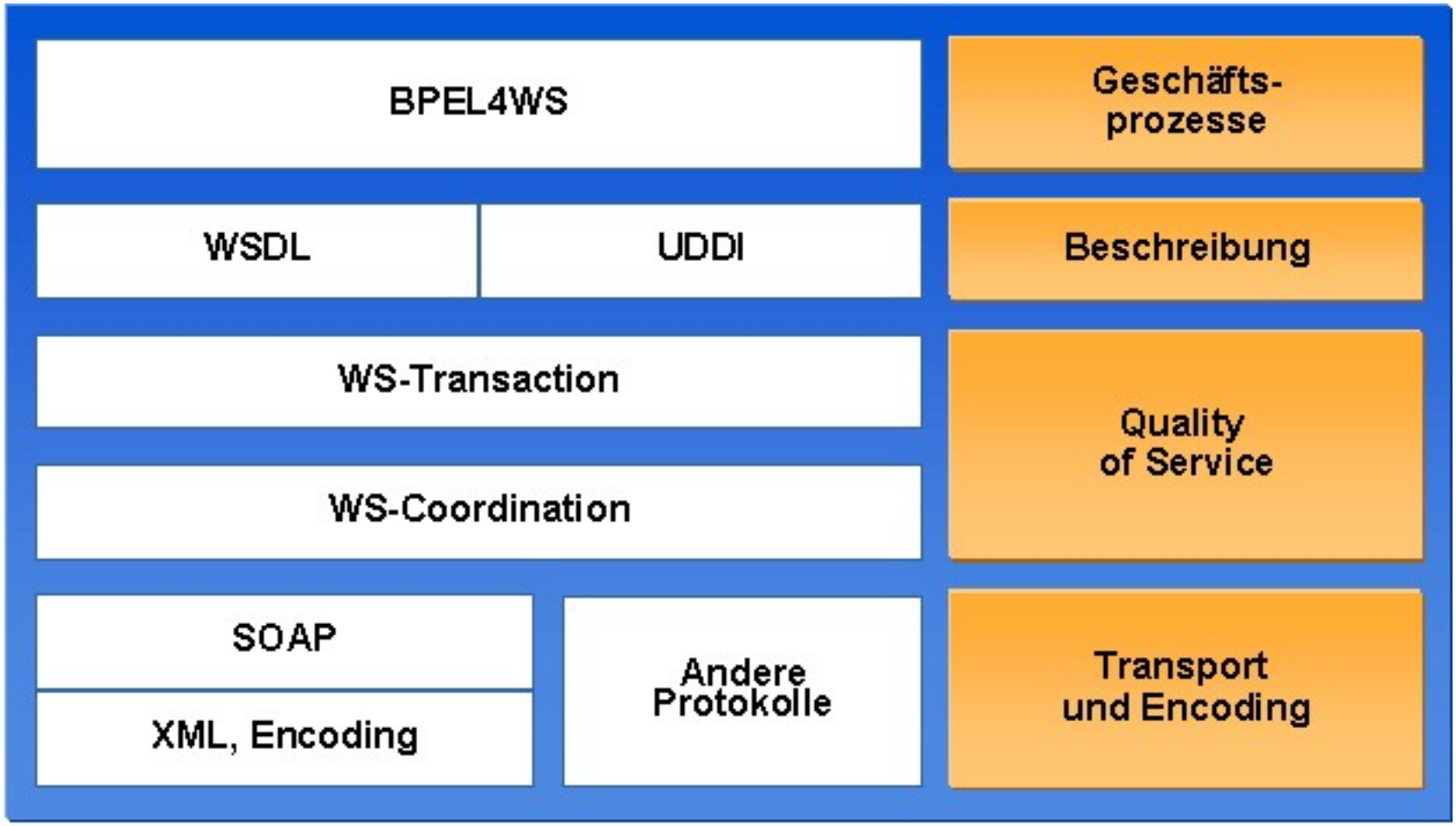
3. Konzept und geschichtliche Entwicklung

- Konzept
  - EAI-Bereich
  - Trennung zwischen Geschäftslogik und konkreter Implementierung
    - Reaktion auf Veränderung der Geschäftslogik
    - Bessere Nutzung der Potentiale
    - B2B (⇨ Outsourcing)
  - Einheitliche Schnittstellen
  - Ganzheitliche Informationssysteme
  - Gemeinsame Standards

- Geschichte
  - Juli 2002
  - IBM (WSFL), Microsoft (XLANG), BEA
  - Siebel Systems, SAP
  - OASIS
  - BPEL4WS -> WS-BPEL -> „BPEL“

4. BPEL4WS als Teil der Webservices-Archtiktur

4. BPEL4WS als Teil der Webservices-Architektur



Vereinfachte Sicht nach Andreas Spall, April 2005

BPEL4WS –

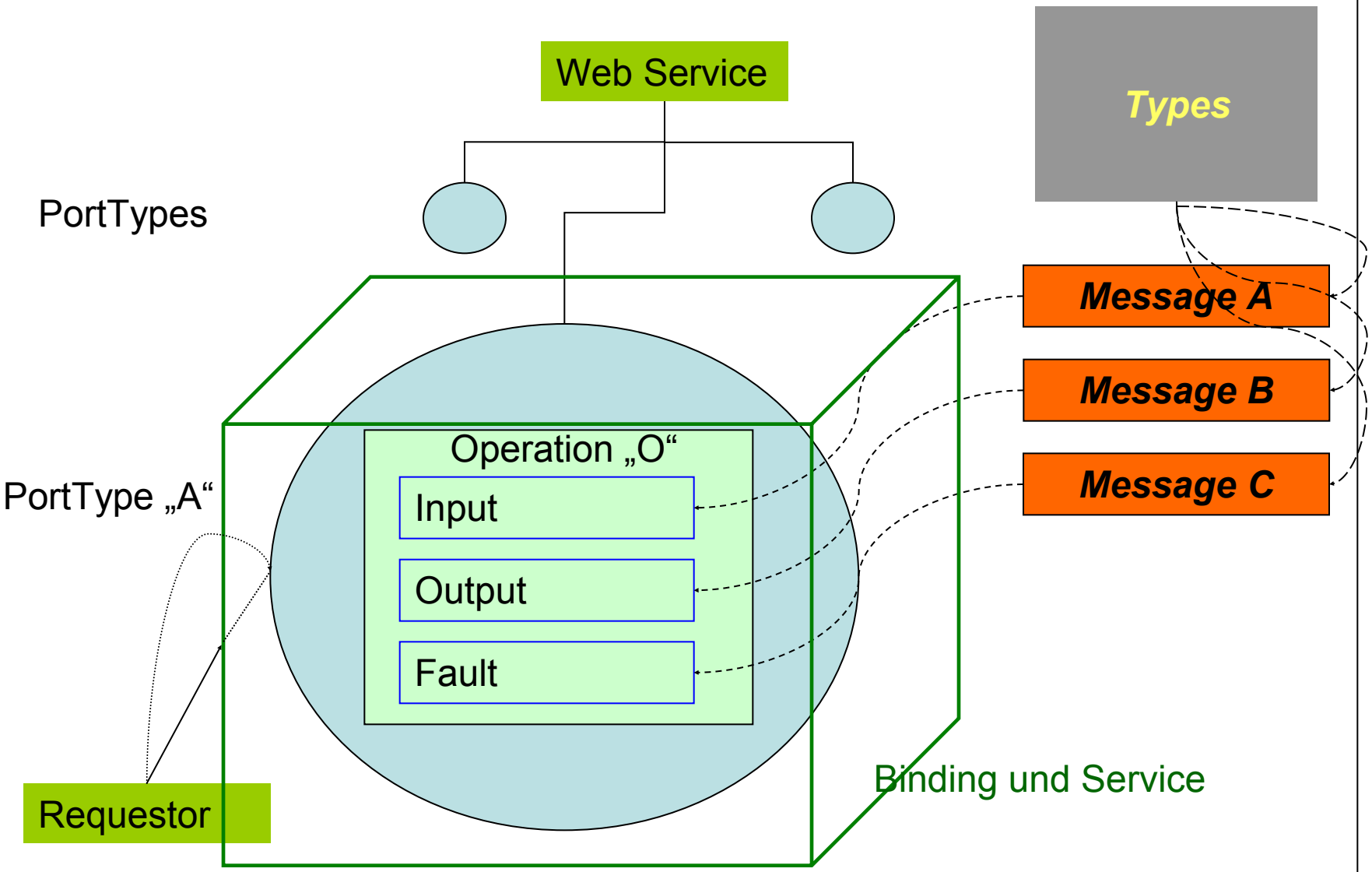
Spezifikation des „orchestration service layer“

- Zusammenspiel von Webservices
- Zusammenschluss zu neuem Web Service
- Gleichzeitig Requester und Service-Anbieter
- Orchestrierung zu einem Geschäftsprozess
- Beschreibung durch WSDL

5. Beschreibung des BPEL4WS-Service per WSDL

- Kurzer Blick auf WSDL
  - Beschreibungsgrundlage eines jeden SOAP-Web Service, also auch für bpel-Prozesse
  - Types
  - Message
  - portType
  - Binding
  - Service

5. Beschreibung des BPEL4WS-Service per WSDL



6. Die Sprache BPEL4WS im Detail

- Technisch: spezielles XML-Schema
- Logisch: imperative Programmiersprache
- Konkrete Ausführung durch Prozess-Manager oder Workflow-Engnines
- Arbeitet auf dem „orchestration service layer“
- <process> als Wurzelelement
  - Z.B. <process name=„myBPELService“>

- Sprachkonzepte:

a) Partnerbeziehungen und Kommunikation

b) Strukturanweisungen

c) Zustände / Sitzungen

d) Ausnahme- und Fehlerbehandlung

e) Zusätzliche Sprachelemente

- Sprachkonzepte und –elemente:

a) Partnerbeziehungen und Kommunikation

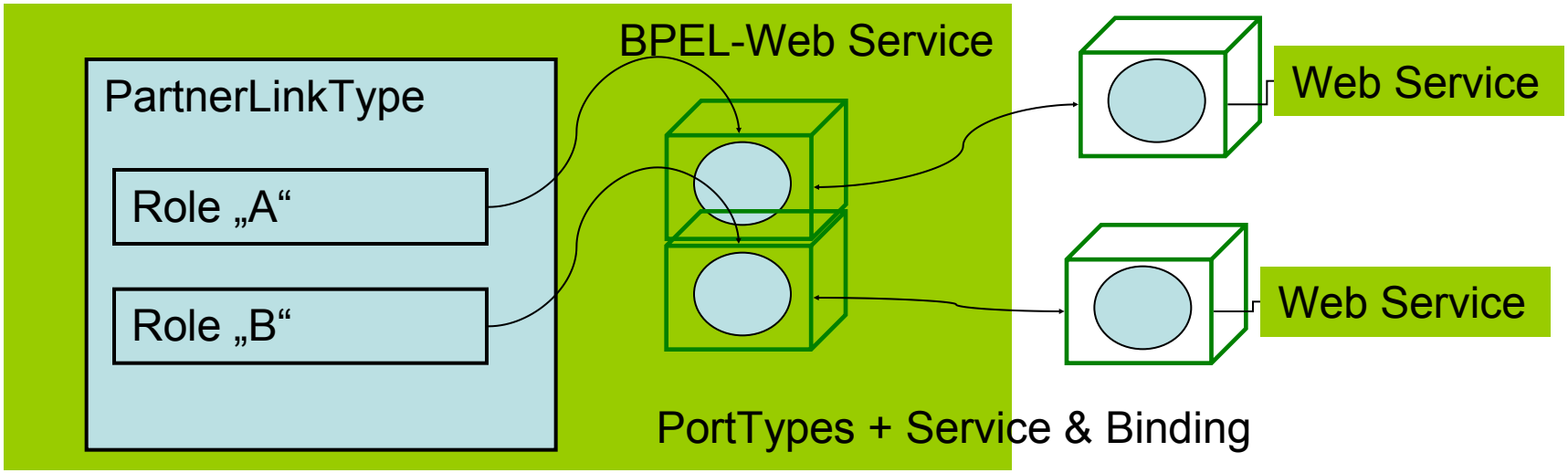
- PartnerLinks, PartnerLinkTypes, Roles
- Invoke
- Receive
- Reply
- Ereignisse
  - EventHandler

Sprachkonzepte und –elemente:

a) Partnerbeziehungen und Kommunikation

PartnerLinkTypes, Roles

- Erweiterung des WSDL-Dokuments des BPEL-Services
- Definition von Kommunikationstypen



Sprachkonzepte und –elemente:

a) Partnerbeziehungen und Kommunikation

PartnerLinks, Roles

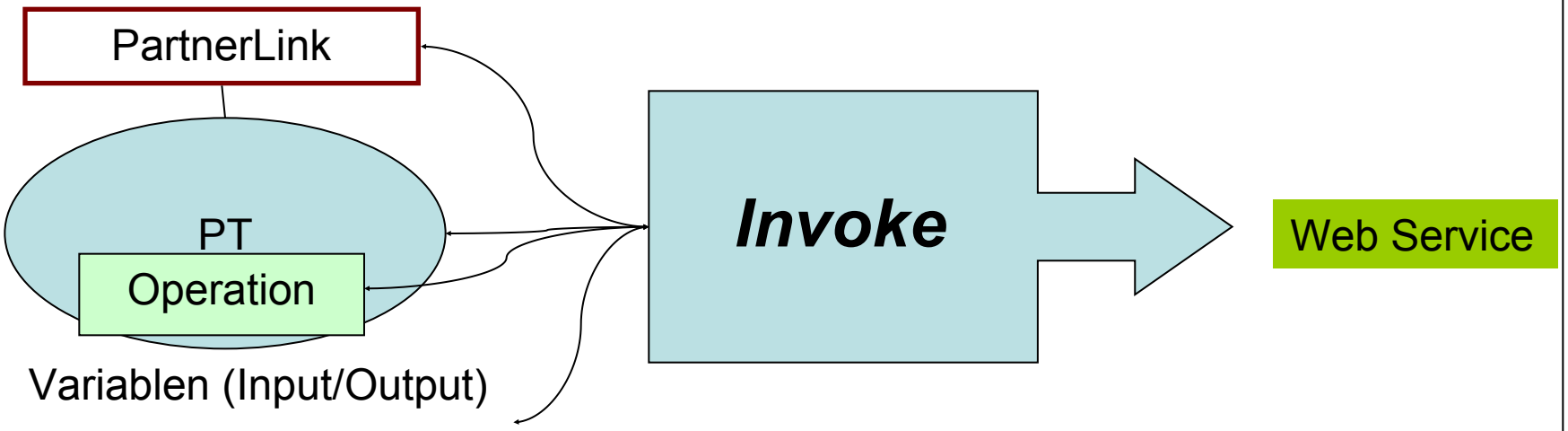
- Innerhalb des BPEL-Dokuments
- Bezugnahme zu PartnerLinkTypes
- myRole, partnerRole

Sprachkonzepte und –elemente:

a) Partnerbeziehungen und Kommunikation

Invoke

- Aufruf eines fremden Web Services



Sprachkonzepte und –elemente:

a) Partnerbeziehungen und Kommunikation

Receive / Reply

- Realisierung eingehender Webdienstaufrufe
- Blockung des weiteren Ablaufs bis zum Empfang
- Für synchrone Kommunikation Reply
  - Ähnlich dem „Return“ klassischer Programmiersprachen
- Variablen zur Aufnahme der Ergebnisse
- Neue Instanz des Prozesses?

Sprachkonzepte und –elemente:

a) Partnerbeziehungen und Kommunikation

Ereignisbehandlung

- EventHandler
- Unabhängig
- Unmittelbar
- Operationsaufrufe oder abgelaufene Zeitlimits
  - Z.B. eine Abbruchmeldung oder Zeitüberschreitung
  - Z.B. on Message

- Sprachkonzepte:

a) Partnerbeziehungen und Kommunikation

b) Strukturanweisungen

c) Zustände / Sitzungen

d) Ausnahme- und Fehlerbehandlung

e) Zusätzliche Sprachelemente

- Sprachkonzepte und –elemente:

  b) Strukturanweisungen

  - Selektion <switch>
  - Iteration <while>
  - Sequenz <sequence>
  - Nebenläufigkeit
    - <flow>
    - <link>
  - Pick
  - Gültigkeitsraum <scope>

Sprachkonzepte und –elemente:

b) Strukturanweisungen

Selektion

- Allgemeinste Form implementiert: <switch>
- Bedingte Aktion („if“)
- Alternative („if-then-else“)
- Auswahl
  („switch-case-otherwise“ bzw. „case-of-else-end“)

Sprachkonzepte und –elemente:

b) Strukturanweisungen

Iteration

- <while>-Schleife
- Klassisch: Wiederholung, so lange eine boolesche Bedingung erfüllt ist.

Sprachkonzepte und –elemente:

b) Strukturanweisungen

Sequenz

- Explizite Angabe einer sequenziellen Abarbeitung möglich.
- Wird besonders interessant innerhalb nebenläufiger Umgebungen
- <sequence>-Element

Sprachkonzepte und –elemente:

b) Strukturanweisungen

Nebenläufigkeit

- Threads innerhalb <flow>-Element
- Gleichzeitiges Abarbeiten mehrerer Aktionen
- Synchronisation über Vor- und Nachbedingungen möglich (<Link>: „source“ / „target“)

```
<flow>
    <links>
        <link name="AtoB">
    </links>
    <!-- invoke operation A -->
    <invoke ..>
        <source linkName="AtoB"/>
    </invoke>
    <sequence>
        … <!-- some activity -->

        <!-- invoke operation B -->
        <invoke ..>
            <target linkName="AtoB"/>
        </invoke>
    </sequence>
</flow>
```

Sprachkonzepte und –elemente:

b) Strukturanweisungen

<Pick>:

- Kombination aus Selektions- und Receive-Semantik
- Nur eine Aktion aus mehreren Alternativen soll ausgeführt werden.
- Vom Programm kann jedoch nicht bestimmt werden, welche zuerst begonnen wird. (nicht deterministisch)
- Blockade aller anderen Möglichkeiten beim Start einer.

Sprachkonzepte und –elemente:

b) Strukturanweisungen

<Pick>

```
<pick>
        <onMessage partnerLink="selling" portType="SellerPT"
        operation="getPrice" variable="itemid">
                <!-- some activities -->
        </onMessage>
        <onMessage partnerLink="selling" portType="SellerPT"
        operation="buy" variable="itemid">
                <!-- some other activities -->
        </onMessage>
</pick>
```

Sprachkonzepte und –elemente:

b) Strukturanweisungen

Gültigkeitsraum

- Scope
- Gültigkeit für z.B.
  - Variablen
  - Fehlerbehandlung, Kompensation
- Darf alles außer „PartnerLinks“ enthalten

- Sprachkonzepte:

a) Partnerbeziehungen und Kommunikation

b) Strukturanweisungen

c) Zustände / Sitzungen

d) Ausnahme- und Fehlerbehandlung

e) Zusätzliche Sprachelemente

- Sprachkonzepte und –elemente:

  c) Zustände / Sitzungen

  - Variablen
  - Korrelation
  - Eigenschaften

Sprachkonzepte und –elemente:

c) Zustände / Sitzungen

Variable

- Daten referenzieren
- Auch Prozessinstanzen
- Wertezuweisung mit Assign

```
<assign>
        <copy>
                <from>70</from>
                <to variable="anzahl">
        </copy>
</assign>
```

Sprachkonzepte und –elemente:

c) Zustände / Sitzungen

Korrelation

- Eigenschaftsmengen zur Identifikation
- Einmalige Initialisierung
- Verwendung bei ein- und ausgehenden Nachrichten
- „Session“-Informationen

Sprachkonzepte und –elemente:

c) Zustände / Sitzungen

Eigenschaften

- Interne Werte von Nachrichten werden normalerweise transparent übertragen
- Mit <property> lässt sich ein Alias definieren auf eine interne Eigenschaften
- Auf dieses Alias ist der externe Zugriff erlaubt
- Interessant z.B. bei Identifikations-Nummern

- Sprachkonzepte:

a) Partnerbeziehungen und Kommunikation

b) Strukturanweisungen

c) Zustände / Sitzungen

d) Ausnahme- und Fehlerbehandlung

e) Zusätzliche Sprachelemente

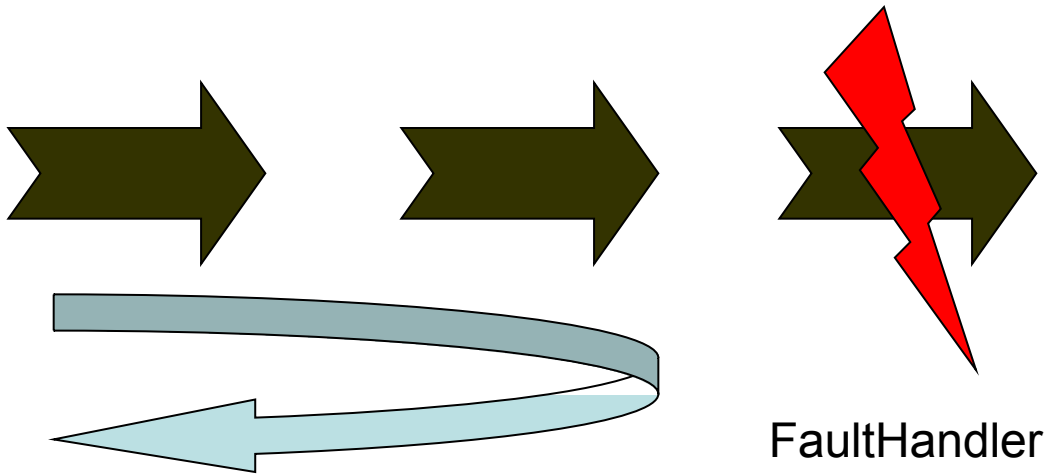
- Sprachkonzepte und –elemente:

d) Ausnahme- und Fehlerbehandlung

- Faults: Teil der WSDL-Sprache
- Globaler Scope > Terminierung
- FaultHandler: <catch>, <catchAll>
- Explizites Hervorrufen
  - <throw>
  - „Faultname“ eines <reply>
- Kompensation
  - Bei langlaufenden Prozessen
  - Rückgängig machen von eigentlich in sich erfolgreichen Aktionen
  - CompensationHandler
  - compensate

- Sprachkonzepte und –elemente:

  d) Ausnahme- und Fehlerbehandlung



FaultHandler kümmert sich um die konkrete Fehlerabwicklung

<Compensate> bzw. CompensationHandler machen Ereignisse rückgängig, die eigentlich in sich erfolgreich waren

u. U. auch mit neuen Webdienst-Aufrufen

- Sprachkonzepte:

a) Partnerbeziehungen und Kommunikation

b) Strukturanweisungen

c) Zustände / Sitzungen

d) Ausnahme- und Fehlerbehandlung

e) Zusätzliche Sprachelemente

- Sprachkonzepte und –elemente:

e) Weitere Sprachelemente

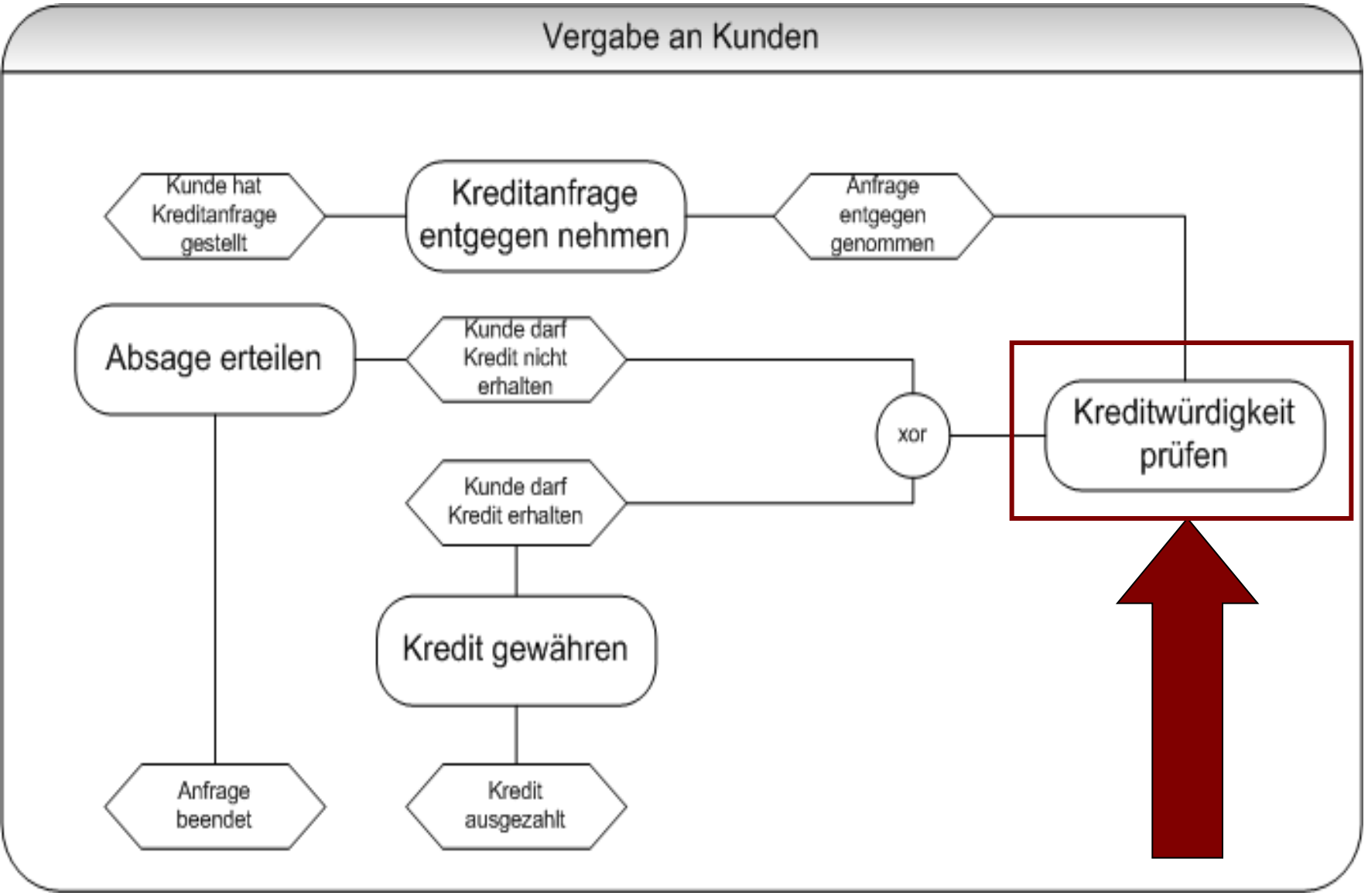
- <terminate>
  - Exit-Semantik
- <wait>
  - Z.B. <wait for="'PT1H2M20S'"/>
- <Empty>
  - Nichts tun
  - Z.B. im Fall eines Fehlers

7. BPEL4WS in der Anwendung

- **Konkrete Ausführung:**
  - BPEL-Workflow-Engines
  - Prozess-Manager
  - Konkrete Umsetzung uninteressant

- **Design:**
  - Design-Tools
  - Code
  - Graphische Elemente

- **Verschiedene Tools:**
  - Oracle BPEL Process Manager
  - Twister
  - INTALIO
  - BPWS4J
  - WebSphere Integration Developer
  - Microsoft Biz Talk Server
  - Active BPEL


Vergabe an Kunden
Kunde hat Kreditanfrage gestellt
Kreditanfrage entgegen nehmen
Anfrage entgegen genommen
Absage erteilen
Kunde darf Kredit nicht erhalten
xor
Kreditwürdigkeit prüfen
Kunde darf Kredit erhalten
Kredit gewähren
Anfrage beendet
Kredit ausgezahlt

- **Kreditwürdigkeit prüfen:**
  - Kunde schickt Daten inklusive Wunschkredit an Prüfungsprozess
  - < 10000: Assessor mit „check“ („prüfen“)
    - Risk: low: „yes“
    - Risk: high: „approve“
  - >= 10000: Approver mit „approve“ („billigen“)
    - Risk: yes / no
  - Rückgabe der Entscheidung
    - Yes / no

8. Ausblick:

BPEL-Erweiterungen

- **BPELJ:**
  - Kombination aus WS-BPEL und Java
  - Kleine Änderungen an Grundsprache
  - Java-Code-Elemente

- **BPEL4People:**
  - Wunsch: Nicht nur Maschine-zu-Maschine
  - „Standardisierung des Human-Interface-Teils“
  - Realiät: viel Benutzerinteraktion

9. Fazit

- **Webservices: Kein neues Konzept**
  - Middleware-Lösungen wie CORBA
- **Möglichkeiten für Geschäftslogik vorhanden**
  - **Weit entwickelte Applikations-Server und Frameworks**
- **Altsysteme nicht wertlos**
  - **Vorzüge nicht verkennen**
- **Neustrukturierung teuer**
  - **IT-Infrastrukturen sind nicht „einfach so“ austauschbar.**

- **Gemeinsamer Standard**
  - Hohe Akzeptanz bei wichtigen Softwareunternehmen
- **Verbessert die Pflege der Geschäftslogik**
  - Zentralisiert, abstrahiert
- **Junge Unternehmen werden es nutzen**
  - Gefahr, abgehängt zu werden technologisch

- **Grad der Granulierung prüfen**
  - Den Grad der Nutzung richtig wählen
  - Nicht Java oder .Net einfach komplett rausschmeißen
- **Mutig Strukturen verbessern**
  - Nicht aus reiner Angst vor Veränderung wichtige Strukturverbesserungen „verschlafen“

Vielen Dank für die Aufmerksamkeit