
Salemi EDV-Beratung

Aggregation von Web Services: Choreography und WS-CDL

Nickolaos Kavantzas, Web Services Architekt
Herausgeber - Oracle Corporation

Alireza Salemi, Web Services Architekt
Übersetzung - Salemi EDV-Beratung

17 September 2004

Gliederung

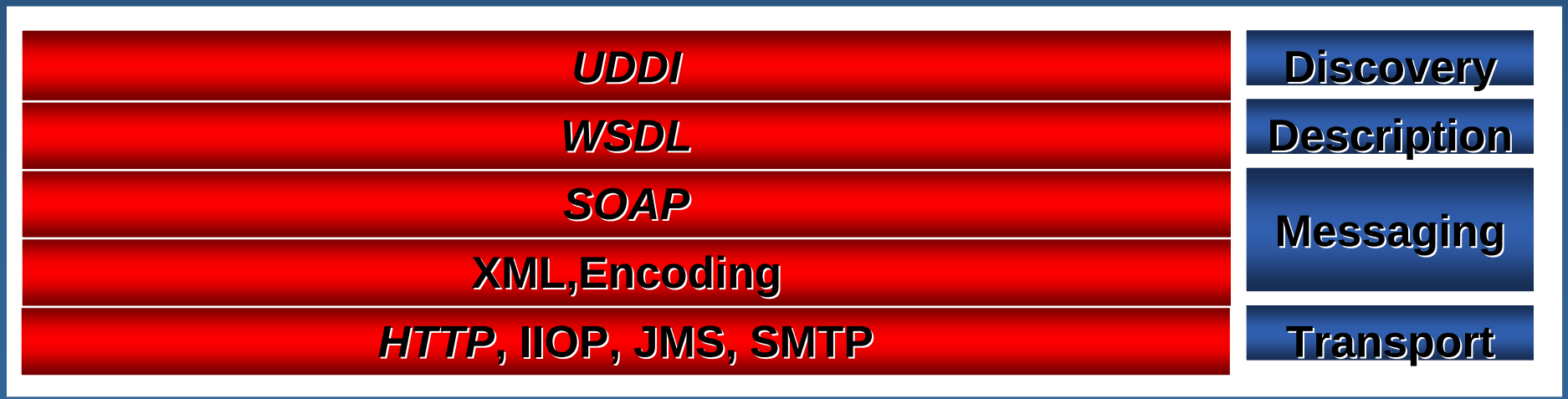
- Web Services Technologiestack
- Aggregation von Web Services
  - Orchestration und BPEL4WS
  - Choreography und WS-CDL
- Kollaborationsprotokolle
  - Business-Protokolle
  - Business-Protokollanforderungen
  - Modellierung von Kollaborationsprotokolle
- WS-CDL in Detail

Web Services

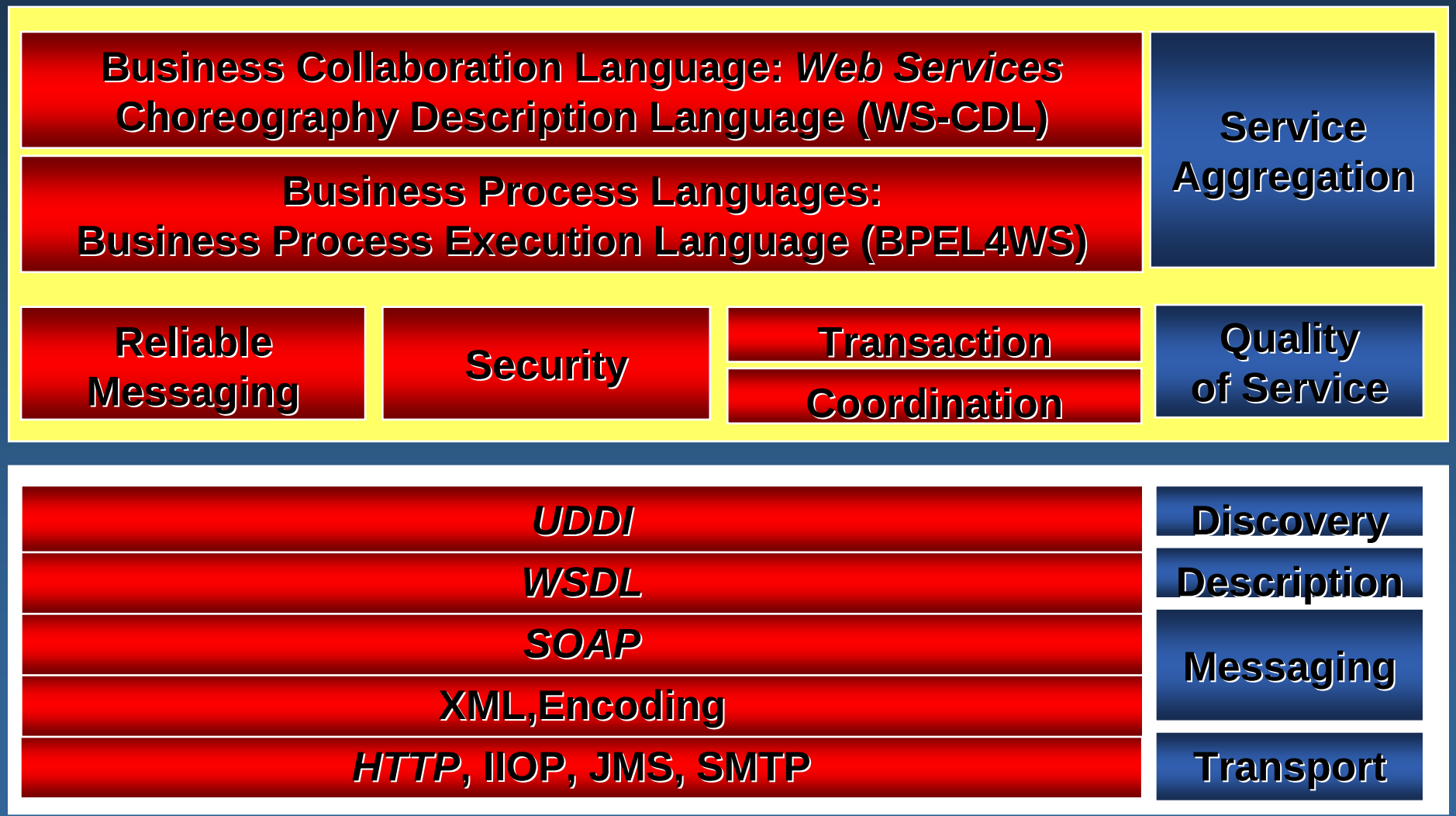
- Organisationen entwickeln neue Lösungen, indem Sie existierende verteilte Anwendungen in ihrem eigenen Domäne integrieren , um die Produktivität zu steigern und operative Kosten zu senken.
- Web Services Plattform agiert als allgegenwärtige Netzwerkfabrik, durch welche neue und existierende Anwendungen durch Informationensaustausch nahtlos kooperieren. Die Kommunikation ist dabei unabhängig von Hardware, Betriebsystemplattform oder Programmiersprache

Aktueller Web Services Technologiestack

- Kern des Kommuniaktionsframework überbrückt heterogene Modelle
  - Verteilt, lose-gekoppelt, zustandslos
  - Austausch von auf Typen überprüfte Information



Neuer Web Services Technologiestack



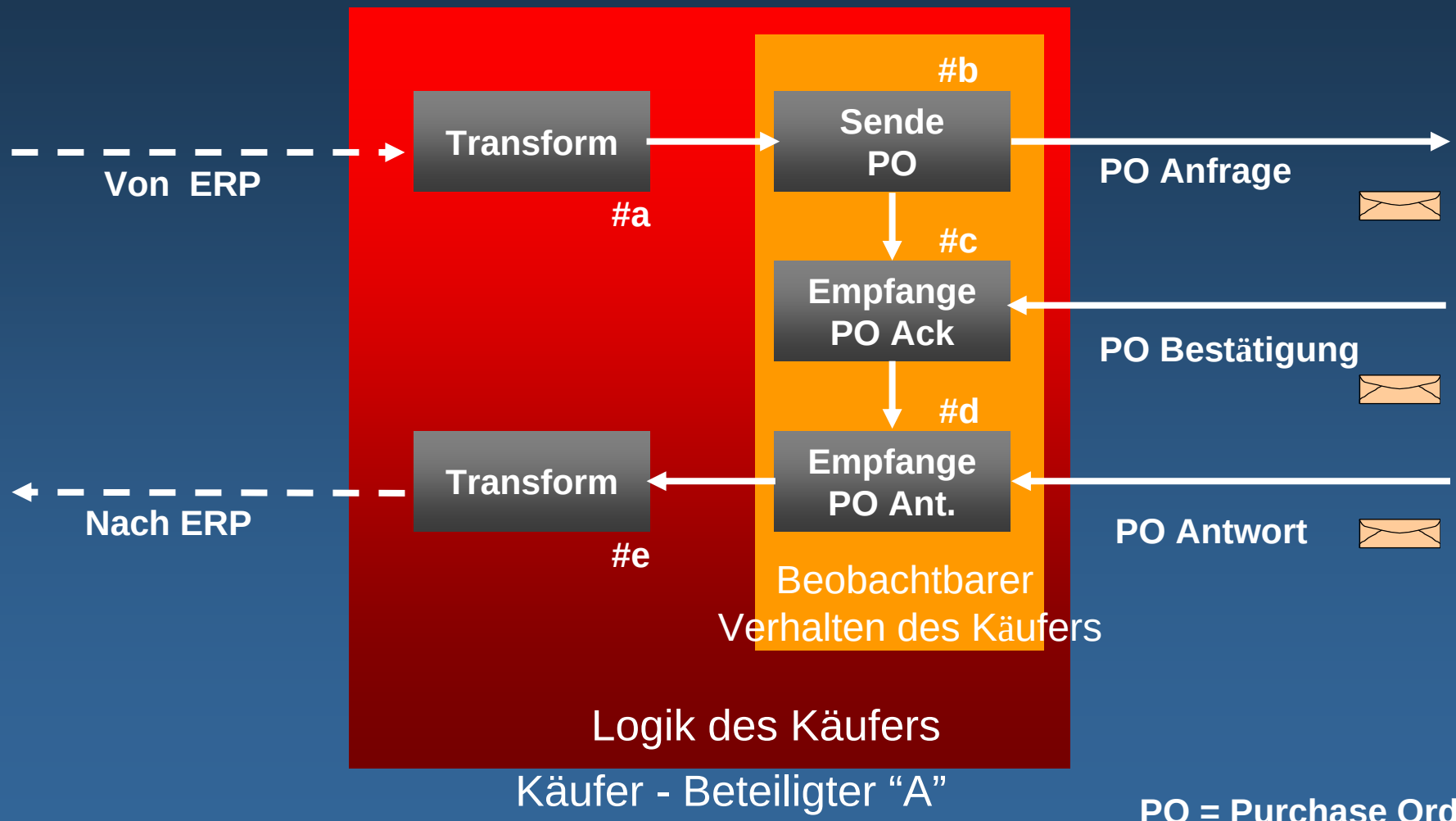
Aggregation von Web Services

- Der inherente Nutzen von Web Services liegt in der sorgfältigen Planung, welche beschreibt: wie die verschiedenartige Anwendungen modelliert sind und wie sie mit einander und Menschen interagieren, wenn sie zusammen aggregiert werden.
- Die operationale Semantiken für Erfüllung dieser Ziele werden durch Folgendes ausgedruckt:
  - Choreography & Orchestration
  - Choreography vervollständigt Orchestration
    - Choreography- und Orchestration-Standards werden für die nächste Generation der Web Services Plattform benötigt

Orchestration und BPEL

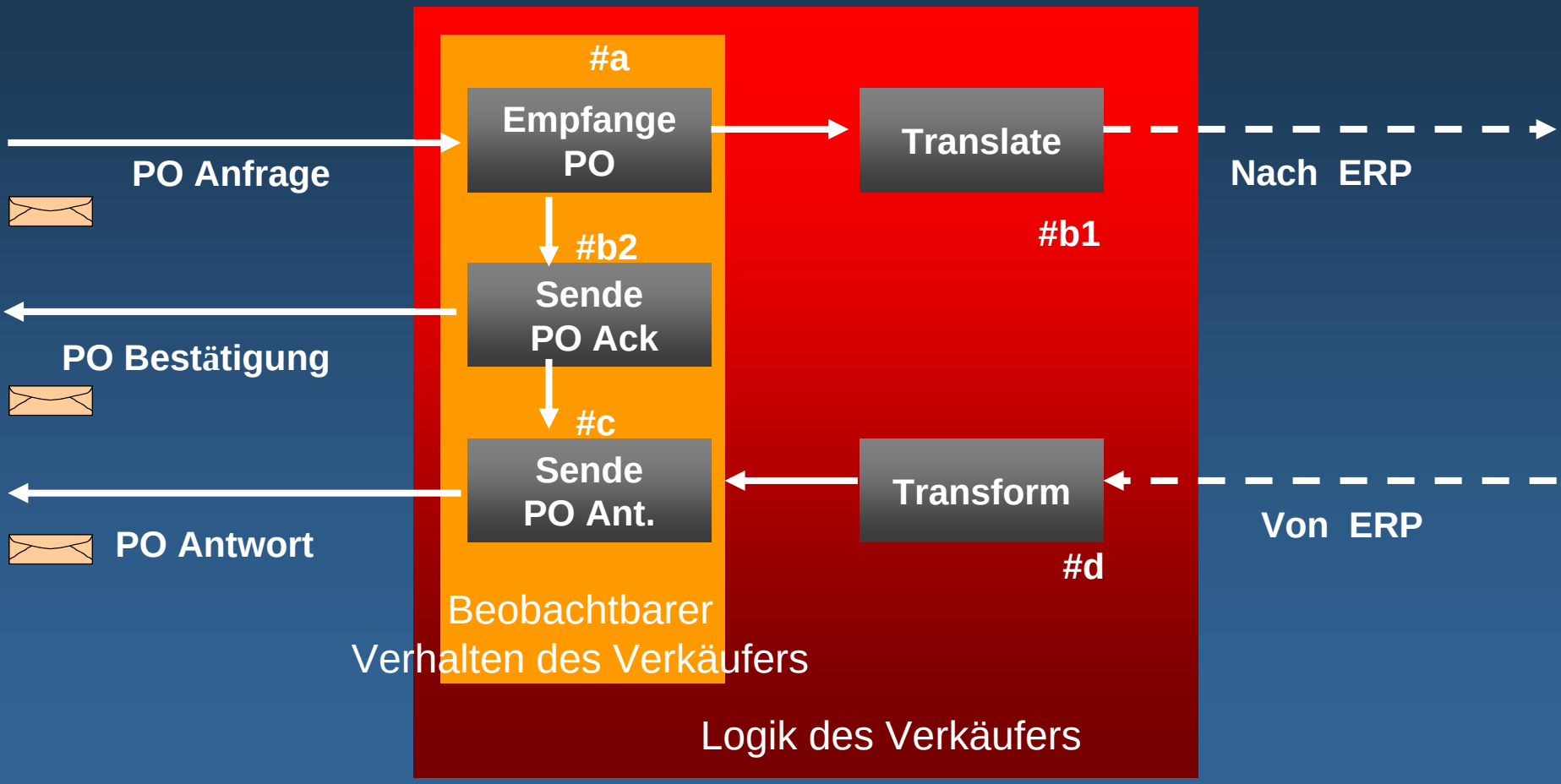
- Orchestration legt den Fokus auf das Verhalten einzelner Teilnehmer
  - Hub/Stern basiertes Modell, wo ein zentrale Steuerung in einem verteilten System existiert, der Lokal den Ablauf von ein Prozess durch die strikte Befolgung seiner Prozessdefinition durchführt.
- BPEL beschreibt die Berechnungslogik für ein Teilnehmer
  - Kontrollfluss (conditionals, sequencing, parallelism, loops)
  - Beobachtbare Verhalten von einem Teilnehmer, aus der Sicht des Teilnehmers selbst
  - Variablen und Konstrukte für deren Manipulation
  - Ereignis-Handlers
  - Timeout-Verwaltung
  - Ausnahmen- und Kompensation-Handlers
- Executable-BPEL/Abstract-BPEL
  - Alle beide besitzen den selben Berechnungs-ausdruckfähigkeit, aber mit verschiedenen syntaktischen Regeln, wobei die Spezifikation der Prozessdefinitions-information durch AbsBPEL abgedeckt ist, als durch ExecBPEL

Orchestration: Logik des Käufers, mit seinem beobachtbaren Verhalten



PO = Purchase Order
d.h. eine Bestellung

Orchestration: Logik des Verkäufers, mit seinem beobachtbaren Verhalten



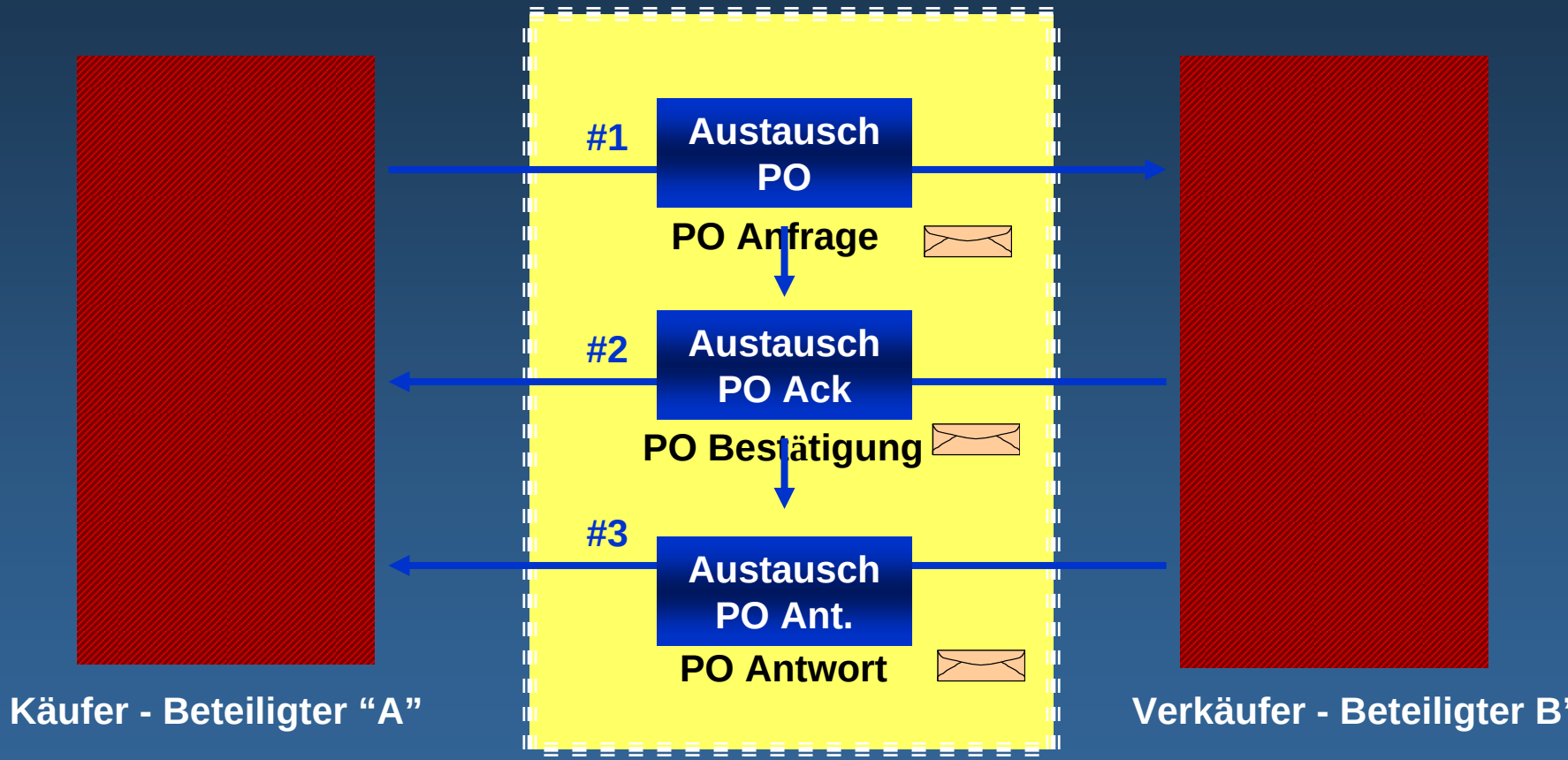
Verkäufer - Beteiligter “B”

PO = Purchase Order
d.h. eine Bestellung

Choreography

- Choreography befasst sich mit globale, „multi-party“, peer-to-peer Kollaborationen, die Interoperabilität zwischen beliebigen Anwendungskomponenten unabhängig von unterstützten Plattformen oder Programmiermodelle sicherstellt, wobei die Komponenten innerhalb oder über die vertrauens-würdige Organisationsbereichen verteilt sein können
  - Die Beteiligten agieren als „peers“, welche sich in langlebende, zustandsbehaftete und koordinierte Sitzungen beeinflussen können
- Choreography ist nicht auf einen zentralisierter Steuerungseinheit (Kontroller) angewiesen.

Choreography: Käufer und Verkäufer kollaborieren



Choreography von Kollaboration des Käufers und Verkäufers

WS-Choreography Description Language (WS-CDL)

- Choreography definiert ein gemeinsames, beobachtbares Verhalten von zwei oder mehreren Beteiligten, welches sich auf eine globale, von Beteiligten nicht wahrzunehmende Sicht konzentriert; wobei der Austausch von Informationen nur dann stattfindet, wenn die miteinander abgemachte, informations-orientierte und reaktive Regeln erfüllt sind.

- WS-CDL ist eine Sprache, die eine Choreography-Beschreibung spezifiziert
  - Zunächst durch Oracle konzipiert und dann von W3C Choreography WG in September 2003 weiterentwickelt
  - Standardisierung erfolgt durch W3C Choreography WG mit der ersten Working Draft, welche in April 2004 veröffentlicht wurde

WS-CDL Nutzung

- Modelliere ein Komplexes Businessprotokoll wie z.B. die Auftragsbearbeitung, das die Interoperabilität zwischen jeglicher Typen von Anwendungs-komponenten ermöglicht, und zwar unabhängig von unterstützten Plattformen oder Programmiermodelle
- Generiere die Berechnungslogik eines einzelnen kollaborierenden Teilnehmers in
  - Z.B. BPEL, Java, C#

WS-CDL Formalismen (1)

- *Globale Modellformalismus* [Nickolaos Kavantzas, work in progress]
  - Basierend auf eine Variante von pi-Kalkül [R. Milner, J. Parrow, D. Walker], "Explicit Solos" Kalkül [P. Gardner, C. Laneve, L. Wischik], ermöglicht die Modellierung eines Systems aus einem globalen Gesichtpunkt

Syntax:

Inf Menge *N* von Namen x, y, u und Literale

x heißt x_1, …x_n (n>=0), loc meint die Einsatzorte

Prozess P, Q, E, F ::=

0	; untätig
| ?g !h P	; globasierte Trigger, abgegliechen
| loc:**x**.$\#_l$ > u > loc':**y**.$\#_r$	; globasierte Interaktion: paired out||in, mit nur Fusion
	; Fortsetzung reduziert auf loc:$\#_l$ || loc':$\#_r$ || loc:loc':**x** # **y**
| (loc:x) P	; Sichtbarkeit
| P || Q	; parallele Komposition
| **loc**:**x** # **y**	; explizit Fusion
| P & Q	; globalisierte Auswahl zwischen Alternativen
| loc >> P	; Projektion eines Prozesses auf einem Lokation
| P @ E @ F	; Choreography von: P normal, E exception, F finalizer

Guard g, h ::=

loc:u | loc:u # v | g + g | g g | h + h | h h

WS-CDL Formalismen (2)

- Linear Typing to support safety/liveness properties in the presence of non-determinism & non-termination [K. Honda, N. Yoshida]

Choreography vs. Orchestration (1)

- WS-CDL ➢ Globale Deklaration sichtbarer Information ebenso die Spezifikation eines Einsatzortes
  - BPEL ➢ Teilnehmer-spezifische Deklaration sichtbarer Information
- WS-CDL ➢ Globale Nachrichtenaustausch (Anfrage & komplemäter passende Zusage) zwischen 2 Teilnehmer über einem gemeinsamen Kanal (w/ gemeinsame Referenz & Korrelation)
  - BPEL ➢ Sende eine Nachricht an einem Teilnehmer, Empfange eine Nachricht von einem Teilnehmer, mit Teilnehmer-spezifische Sicht der messageInformation/partnerLocators/correlations
- WS-CDL ➢ Global Reaktive Regeln für deklarative Festlegung des normalen/abnormalen Ablaufes, basierend auf das flexible, informationsorientierte Modell (information driven model)
  - BPEL ➢ Teilnehmer-spezifische Sichten des Kontrollflusses (seq, flow, switch, controlLinks, exception/compensation handlers), basierend auf dem starren, imperativen Modell

Choreography vs. Orchestration (2)

- WS-CDL ➢ Gemeinsame Übereinkunft über das Ergebnis (Information alignment) der Beteiligten
  - BPEL ➢ Keine Abkommen über das Ergebnis der Teilnehmern
- WS-CDL ➢ Rekursiv zusammengesetztes Modell auf der Choreography Ebene und die Schablonen(Template)-Bildung erlaubt die Wiederverwendung in verschiedenen Business-Kontexte
  - BPEL ➢ Zusammengesetztes Modell auf dem WSDL Level
- WS-CDL ➢ Erzeugte Berechnungslogik kann je nach kollaborierender, individueller Teilnehmer aus BPEL, Java, C#, etc bestehen.
  - BPEL ➢ Berechnungslogik eines Einzelteilnehmers ist BPEL

Collaboration Protokolle

- Business-Protokolle, wie z.B. Auftragsabwicklungs-verwaltung
  - Umfasst die Erzeugung von Bestellungen und Änderung/Stornierung von laufenden Bestellungen
  - Gefunden in
    - Finanzprotokolle, wie FIX, TWIST Protokolle
    - RosettaNet PIPs
- Aber kann sich auch um technischen Protokolle handeln, wie WS-Business Activity Protokoll/WS-AtomicTransaction [Microsoft, IBM]
  - Definiere ein klassische/erweiterte Transaktionsmodell

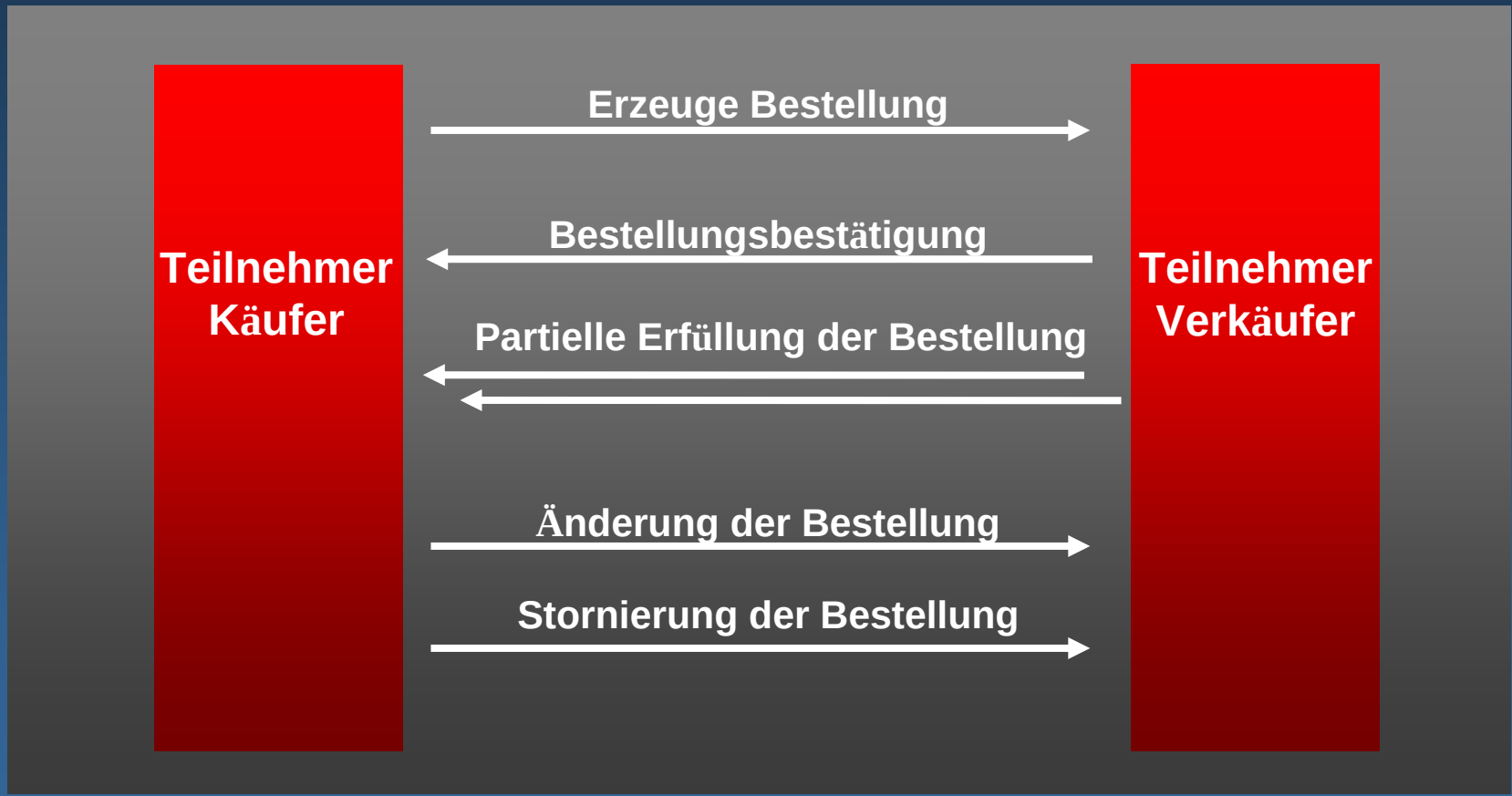
Kollaborationsprotokoll Anforderungen (1)

- Wiederverwendbarkeit in verschiedenen Kontexte (Industrie, Schauplatz, etc.)
  - Erlaubt die inkrementelle Spezifikation mit variierender Detailstufe
- Zusammengesetztes Modell zur Ermöglichung skalierbarer Modellierung
  - Baue zuerst kleine Choreographies
  - Kombiniere sie zu einer Form der größeren Choreography zusammen

Kollaborationsprotokoll Anforderungen (2)

- *Globale Modell* über zwei oder mehrere kollaborierende Teilnehmern, von:
  - Aussetzfreier Nachrichtenaustausch (Anfragen stimmen mit komplementären Bestätigungen überein)
  - Channel-Passing, durch zwei oder mehrere Teilnehmer
  - Zustanderzeugungen/-änderungen, Wettlaufsituationenaufnahme
  - Gemeinsame, reaktive Regeln & Reihenfolgebedingungen (normal, exceptional), festgesetzter und informationsorientierter Berechnungsablauf in eine deklarative und flexible Art und Weise
  - Informationsorientierung garantiert ein gemeinsames Verständnis von ausgetauschte Information und Zustandwechseln
  - Koordinierter Ablauf (normal, exceptional)
- Mit größerer Ausdrucksfähigkeit als das CCS Modell
- Und die Gewährleistung der Sicherheits-/Lebendigkeits-Eigenschaften mit Anwesenheit von Nichtdeterminismus & Non-Termination

Real-life Business Kollaboration Protokoll: Auftragsbearbeitung

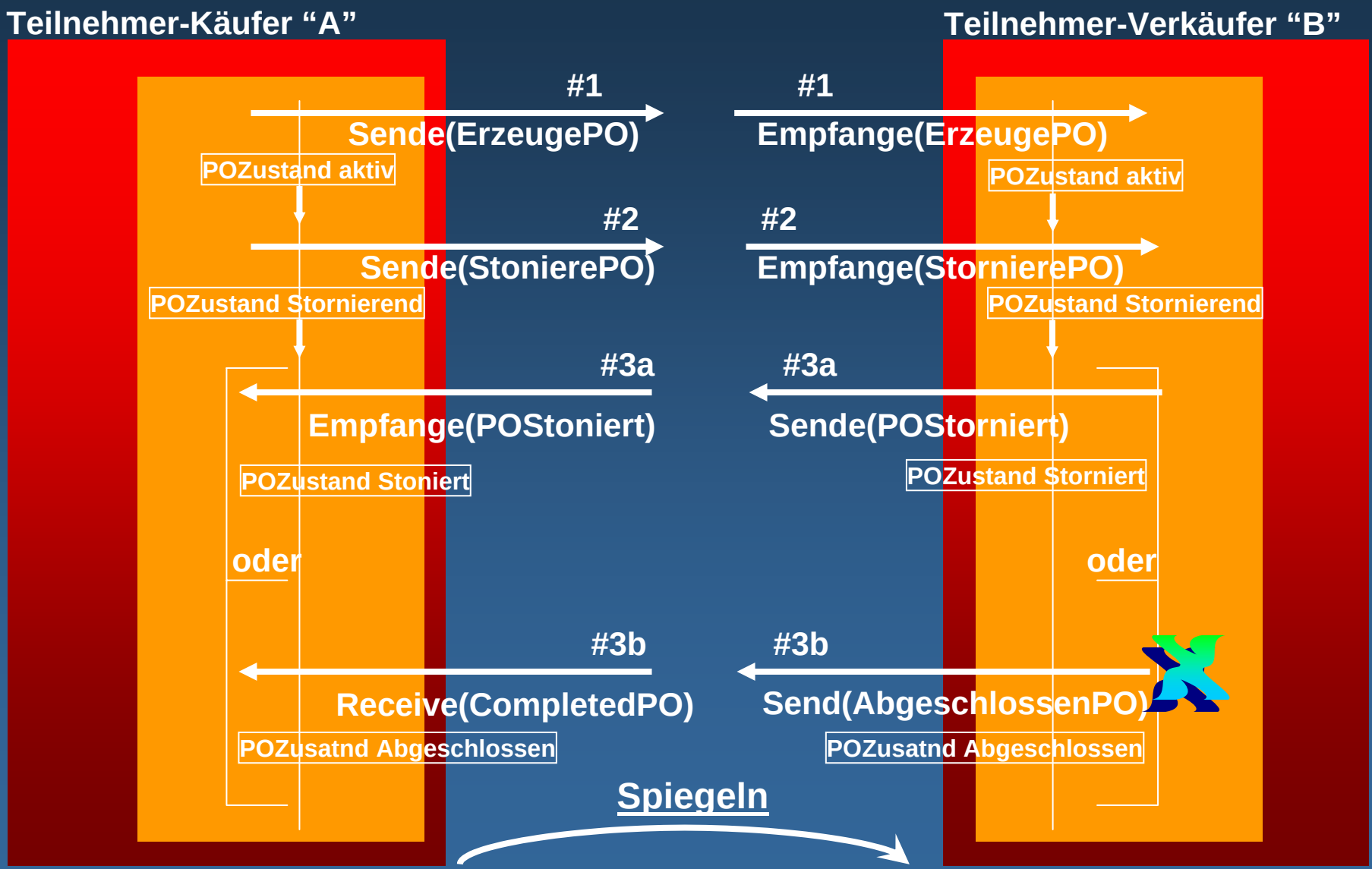


Modellierung eines Kollaborations-Protokolls mit BPEL

Entwurfe individuelle Puzzleteile zuerst und setze sie nachher zusammen

- **Schritt 1:** Entwerfe einen Teil
  - Modelliere beschränktes Verhalten eines Teilnehmers: Käufer
- **Schritt 2:** Entwerfe anderen Teil
  - Modelliere beschränktes Verhalten des anderen Teilnehmers: Verkäufer, Lieferant, etc.
- **Oops!** Die individuell entworfene Teile passen nicht zusammen
  - Das zusammengesetztes Verhalten der individuell modellierten Verhalten kann ein unerwünschtes, globales Systemverhalten darstellen (siehe #3b, Teilnehmer-Verkäufer auf der folgenden Folie)

Modellierung eines Kollaborationsprotokolls mit BPEL: *unerwünschtes* Globalverhalten

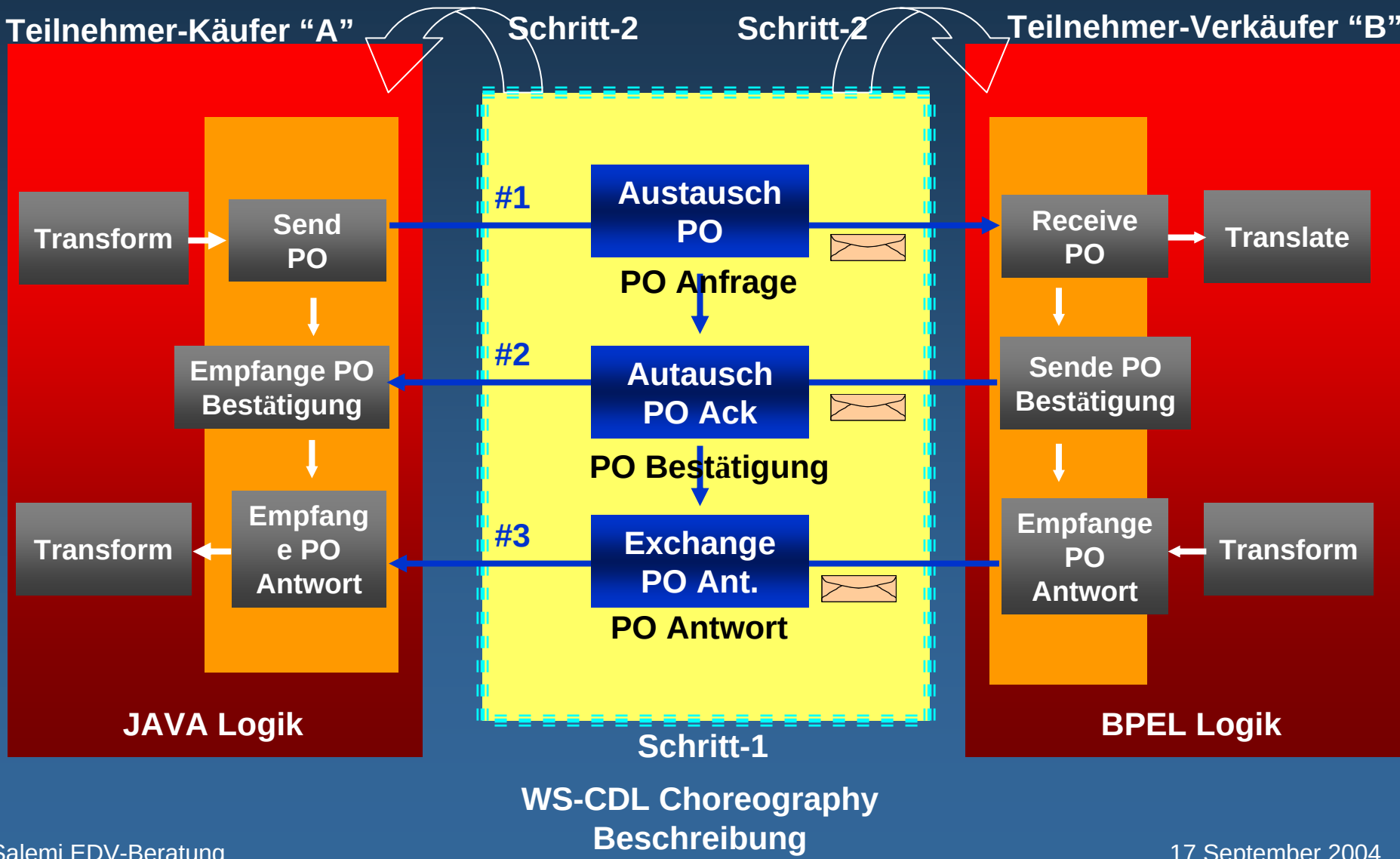


Modellierung eines Kollaborations-protokolle mit WS-CDL

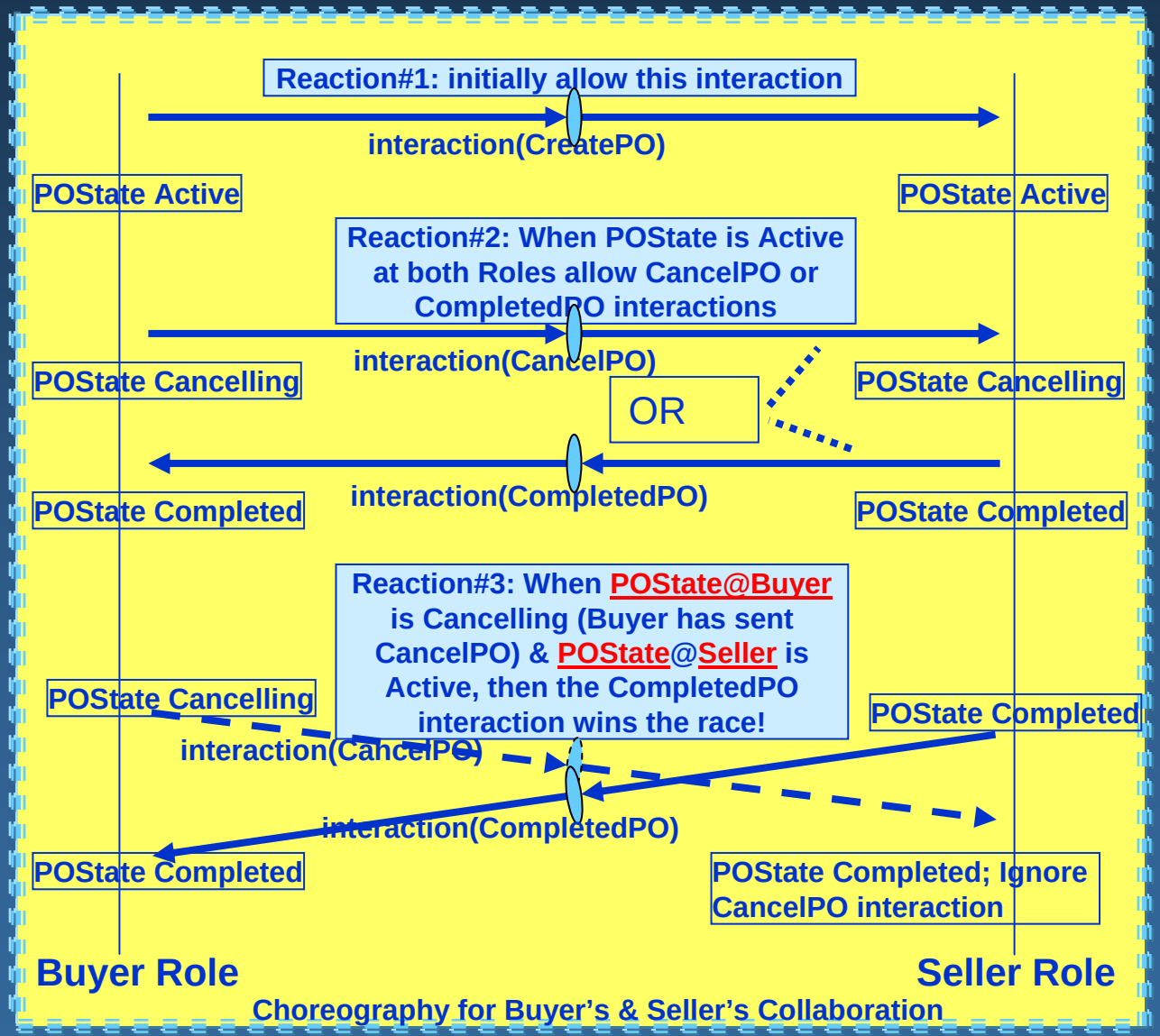
Zuerst ein Gesamtbild und spätere Zerlegung des Puzzles in die Einzelteile

- **Schritt 1:** Ein Gesamtbild
  - Starte bei der Modellierung der Kollaboration zwischen dem Teilnehmer Käufer und dem Teilnehmer Verkäufer
- **Schritt 2:** Dann zerlege es dann in Teile
  - Aus WS-CDL können wir BPEL/Java/C# Logik für jede Teilnehmer generieren, z.B.
    - Java Logik für den Käufer
    - BPEL Logik für den Verkäufer

Modellierung eines Kollaborations-protokolle mit WS-CDL



Modellierung eines Kollaborationsprotokolls mit WS-CDL : *unerwünschtes* Globalverhalten



WS-CDL im Detail

WS-CDL Konzepte

- Typisierung
  - Information Type
  - Token, Token Locator
- Indentifizierung & Koppelung kollaborierende Teilnehmer
  - Role
  - Relationship
  - Participant
- Informationsorientierte Kollaborationen
  - Channel Type
  - Variable Definitions
  - Activities
    - interaction, perform, assign, noaction
  - WorkUnit/Reaction
  - Choreography
  - Package, Import

Typisierung

- Information type
  - Kürzt WSDL-Typen und XSD-Typen & -Elemente ab.
  - Unterstützt andere Typensysteme
- Token type
  - Spezifiziert Name und Typ als eine Abkürzung für ein Informationsteil innerhalb eines Dokumentes
- Token Locater type
  - Spezifiziert Regeln zur Auswahl der Informationsteile innerhalb eines Dokumentes

Typisierung Syntax

```
<informationType  name="ncname"
                          type=”qname”? |
                          element=”qname”? />


<token  name="ncname"  informationType="qname" />

<tokenLocator  tokenName="qname"
      informationType="qname"
      query="XPath-expression"? />
```

Roles, Relationships, Participants

- Role type
  - Aufzählung der beobachtbaren Verhalten, die durch ein oder mehrere Behavior type(s) spezifiziert sind, und eine kollaborierende Teilnehmer darstellen
  - Behavior type spezifiziert die unterstützten Operationen
    - Optional WSDL interface type
- Relationship type
  - Spezifiziert die gegenseitigen Verpflichtungen, bezüglich der Roles/Behavior types, zweier kollaborierende Teilnehmer werden zur Definition benötigt
- Participant type
  - Aufzählung einer Menge von einer oder mehreren Rollen, die kollaborierenden Teilnehmern darstellen

Roles, Relationships, Participants Syntax

```
<role  name="ncname" >
  <behavior  name="ncname"
      interface="qname"? />+
</role>

<relationship  name="ncname">
  <role  type="qname"  behavior="ncname" />
  <role  type="qname"  behavior="ncname" />
</relationship>

<participant  name="ncname">
  <role  type="qname" />+
</participant>
```

Channel (1)

- Realisiert einen *dynamischen* Punkt der Kollaboration, durch welche die kollaborierende Teilnehmer interagieren
  - Wo & wie sind Nachrichten auszutauschen
    - Spezifiziert *Role/Behavior* und die *Referenz* auf ein kollaborierende Teilnehmer
    - Identifiziert eine *Instanz* einer Rolle
  - Identifiziert einen Instanz einer Konversation zwischen zwei oder mehrere kollaborierende Teilnehmern
    - Ein Konversation gruppiert eine Menge von verwandten Nachrichtenaustauschen

Channel (2)

- Ein oder mehrere channel(s) KÖNNEN von einer Rolle zu eine oder mehrere anderen Rolle(n) durchgereicht werden, möglicherweise in einer laufenden Sitzung durch eine oder mehrere Zwischenrolle(n), in dem neue Punkte der Kollaboration dynamisch erzeugt werden
  - Eine „Channel type“ KANN die erlaubten Typen von Channel(s), welche zwischen der Webservice-Teilnehmern auszutauschen sind, durch dieser Kanal festlegen.
  - Eine Channel type KANN seine Verwendung begrenzen, in dem es die Anzahl der Kanalzugriffe spezifiziert

Channel Syntax

```
<channelType  name="ncname"
    usage=“once”|”unlimited”?
    action=”request-respond”|“request”|”respond”? >

  <passing  channel=”qname”
        action=“request-respond”|“request”|”respond”?
        new=”true”|”false”? />*

  <role  type="qname"  behavior="ncname"? />

  <reference>
    <token type="qname"/>+
  </reference>

  <identity>
    <token  type="qname"/>+
  </identity>*
</channelType>
```

Variablen (1)

Aufnahme der Instanzinformation über die Objekte in der Kollaboration

- Sterung des Kollaborationsablaufs zwischen den kollaborierenden Teilnehmern
- Variable Typen:
  - Informationsaustauschvariablen: definiert die Instanzen der auszutauschenden Dokumente zwischen den Rollen in einer Interaktion
  - Zustandsvariablen: definiert die Instanzen der Zustands-information einer Rolle
  - Kanalvariablen: definiert Instanzen einer Kanaltyp
- Und ihre Definitionen:
  - Spezifiziert den Typ des Wertes durch die Elemente informationType & channelType, der eine Variable enthält
  - Spezifiziert die Rolle des kollaborierende Teilnehmer, welche eine Variable beinhaltet

Variablen (2)

- Der Wert einer Variable
  - Ist verfügbar für alle Rollen, in dem er vor dem Start der Kollaboration initialisiert wird. Gemeinsame Variablen enthalten die Information über die gemeinsame Wissen von zwei oder mehreren Rollen
  - Kann für die anderen Rollen verfügbar gemacht werden, in dem er als Ergebnis einer Interaktion weitergereicht wird.
  - Kann durch die Zuweisung anderer Informationsdaten für eine Rolle abrufbar gestaltet werden. Lokal definierte Variablen, die Information enthalten, werden lokal durch eine Rolle erzeugt und geändert. Sie können Informations-austausch-, Zustand- oder Kanalvariablen sein
  - Kann genutzt werden, um die Entscheidungen und Aktionen zu ermitteln , die die innerhalb einer Choreography auszuführen sind.

Variable Syntax

```
<variableDefinitions>
   <variable   name="ncname"
      informationType=”qname”|channelType="qname"
      mutable=”true|false”?
      free=”true|false”?
      silent-action=”true|false”?
      role="qname"? />+
</variableDefinitions>
```

Aktivitäten: Interaction (1)

Ermöglicht kollaborierende Teilnehmern zu kommunizieren und Information abzugleichen

- Nachrichtenaustausch zwischen zwei Rollen innerhalb einer Beziehung
  - Anfrage & Bestätigung einer Operation durch einen gemeinsamen Channel
    - Einweg oder Anfrage-Antwort Operation
    - Informationsfluss
      - request Richtung: fromRole nach toRole
      - response Richtung: toRole nach fromRole

Aktivitäten: Interaction (2)

- Zustandsaufnahmen über eine Rolle
  - Erzeugung neue, Änderung existierende Variablen über eine Rolle
- Informationsausrichtung: Übereinkunft über ihre gemeinsame Verständnis von den
  - Zustandsänderungen der Variablen innerhalb einer Rolle mit den Zustandsänderungen der Variablen innerhalb anderen Rollen
  - Werte der ausgetauschten Nachrichten von einer Rolle nach eine andere Rolle

Interaction Syntax

```
<interaction  name="ncname” channelVariable="qname"
              operation="ncname"
              time-to-complete="xsd:duration"?
              align=“true”|”false”?
              initiateChoreography="true"|"false"? >
  <participate  relationship=”qname”
                fromRole="qname"
                toRole="qname" />
  <exchange  messageContentType="qname"
             action=“request”|”respond” >
    <use  variable=“XPath-expression”/>
    <populate  variable="XPath-expression"/>
  </exchange>*
  <record  name="ncname" role=“qname” action=“request”|”respond” >
    <source  variable=“XPath-expression” />
    <target    variable="XPath-expression" />
  </record>*
</interaction>
```

Activities: Assign

- Zustandsaufnahmen über eine Rolle
  - Erzeugung neue, Änderung existierende Variablen über eine Rolle

```
<assign  role=“qname”>
  <copy  name="ncname">
    <source  variable =“XPath-expression” />
    <target  variable="XPath-expression"  />
  </copy>+
</assign>
```

WorkUnit / Reaction

- Informationsorientiertes Modell, wo ein Reaktionsregel die Menge der Aktivitäten überwacht, und die Festlegung der benötigten Bedingungen für die Information die Ausführung der normalen/abnormalen Ablauf sicherstellen
  - Reaktionüberwachung formulieren die Interesse an die Verfügbarkeit einer oder mehreren Informationen einer Variable
  - Wenn die Variable verfügbar ist / wird und die Überwachungsbedingung nach Wahr wechselt, dann werden die enthaltenen Aktivitäten aktiviert.
  - Überwachungsbedingung kann für die Variablen, die in verschiedene Rollen existieren definiert sein.
- Repeat: kennzeichnet die Wieder-Aktivierung von einem workunit
- isAligned(var1, var2, rel) spezifiziert die Interesse an einem Angleichung der Variablen in zwei Rollen innerhalb einer Beziehung (Relationship)

WorkUnit / Reaction Syntax

```
<workunit  name="ncname"
      guard=“xsd:boolean XPath-expression”?
      repeat=“xsd:boolean XPath-expression”?
      block=“true|false” >

      Activity
</workunit>
```

Aktivierung innerhalb einer Menge von Aktivitäten

- Sequential: spezifiziert die Aktivierung der Aktivitäten in einem sequentiellen Modell
- Parallel: spezifiziert die Aktivierung der Aktivitäten in einem parallelen Modell
- Choice: spezifiziert die Aktivierung einer Menge von zwei oder mehr Aktivitäten

```
<sequence>
    Activity+
</sequence>

<parallel>
    Activity+
</parallel>

<choice>
    Activity+
</choice>
```

Choreography

- Kombiniert Aktionen mit einem gemeinsamen Verhaltenscharakteristik, um eine Kollaborationseinheit der Arbeit zu formen.
  - Aufzählung aller binäre Beziehungen, Interaktionen der Choreograhy
  - Lokalisieren der Sichtbarkeit von Variablen
    - Nutzung von Variablendefinitionen
  - Festlegung alternative Verhaltensmuster
    - Nutzung von workunits/reactions
  - Aktivierung von Wiederherstellung
    - Nutzung von workunits/reactions
    - Rückwärts: Behandlung von Ausnahmebedingungen
    - Vorwärts: Beenden bereits abgeschlossenen Aktivitäten

Choreography Syntax

```
<choreography  name="ncname"
      complete=“xsd:boolean XPath-expression”?
      isolation=”dirty-write”|
      ”dirty-read”|”serializable”?
      root=”true”|”false”? >

   <relationship  type="qname" />+
        variableDefinitions?
        Choreography*

        Activity

   <exception  name="ncname">
            WorkUnit+
   </exception>?

   <finalizer  name="ncname">
          WorkUnit
   </finalizer>
</choreography>
```

Choreography Komposition (1)

- WS-CDL's zusammengesetztes Modell erlaubt skalierbare Modellierung
  - Zuerst baue Kleine Choreographies auf
  - Kombiniere sie zu einer größeren choreography zusammen
  - Dann bilde den Teilnehmer ab, um den Teilnehmersicht zu bekommen
    - Komposition macht die Teilnehmerabbildungen einfacher
    - Ohne das Gesamtbild von einem "business process" zu betrachten, machen die Teilnehmerabbildungen keinen Sinn

Choreography Komposition (2)

- Choreographies können durch die rekursive Zusammensetzung von neuen Choreographies kombiniert werdem, in dem “perform activity” angewandt wird

```
<perform  choreographyName=”qname”>
  <alias  name="ncname”>
    <this  variable=“XPath-expression” role=“qname” />
    <free  variable="XPath-expression" role=“qname” />
  </alias>+
</perform>
```

Package, Import, Templates

Erleichterung der Modularität und Wiederverwendbarkeit

- Ein Choreography „package“ erlaubt die Kombination der Choreography-Typen unter einem gemeinsamen Namespace:
  - informationType, token, tokenLocator
  - role, relationship, participant
  - channelType
  - variableDefinitions
  - choreography
- Ein „import“ erlaubt die Nutzung von der Choreography-Typen, die in andere Choreography-Pakete definiert sind.
- Schablonenbildung erlaubt die schrittweise Spezifizierung der unterschiedlichen Detailgrade, um die Wieder-verwendbarkeit von Choreographies in verschiedenen Kontexte zu ermöglichen (Industrie, Schauplatz, etc.)

Package, Import Syntax

```
<package
    name="ncname"
    author=”xsd:string”?
    version=”xsd:string”
    targetNamespace="uri"
    xmlns=”http://www.w3.org/ws/choreography/2004/02/WSCDL/”>
    importDefinitions*
    informationType*
    token*
    tokenLocator*
    role*
    relationship*
    participant*
    channelType*
    Choreography*
</package>

<importDefinitions>
    <import  namespace="uri"  location=”uri” />+
</importDefinitions>
```

Zusammenfassung

Oracle: Orchestration und Choreography Strategie

- Orchestration
  - Oracle Bekenntnis zu BPEL innerhalb „middle tier runtime“, Management und Entwicklungswerkzeuge
  - Direkte Produktauswirkung: OracleAS Integration und Oracle Jdeveloper
- Choreography
  - W3C WS-CDL Spezifikation Editor und Schlüssel-teilnehmer in W3C Choreography Working Group
  - Arbeiten mit und Einbeziehen früherer Adaptierende, um spezifische Anforderungen auszuarbeiten

Conclusions

- Choreography ergänzt Orchestration
- Choreography befasst sich mit globale, „multi-party“, peer-to-peer Kollaborationen, die Interoperabilität zwischen beliebigen Anwendungskomponenten unabhängig von unterstützter Plattform oder Programmiermodell sicherstellt, wobei die Komponenten innerhalb oder über die vertrauenswürdige Organisationsbereichen verteilt sein können
- WS-CDL basiert auf ein formale Modell
- WS-CDL ist z.Z. unter Normierung in W3C Choreography WG
  - Bitte, laden Sie die letzte „Working Draft“ und Kommentare unter