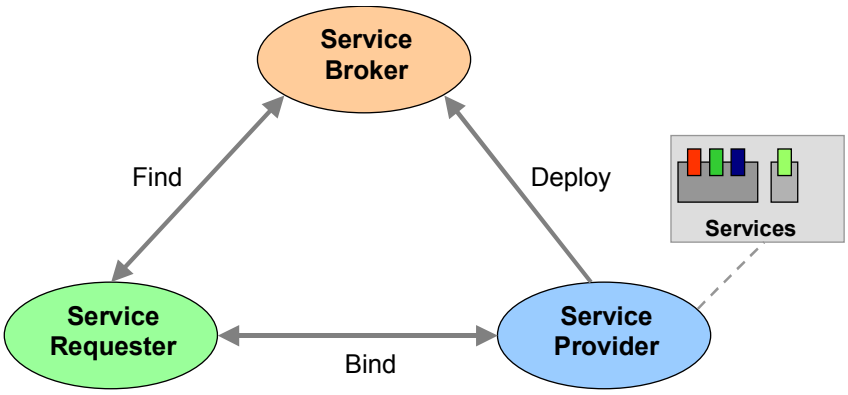
Web Services and Service Flows

- Service Oriented Architecture (SOA):
- Web Services
  - sollen die einfache Nutzung bzw. Integration von Anwendungsdiensten über das Web ermöglichen
  - umfassen ggf. mehrere Operationen, für deren Ausführung gewisse Restriktionen (z.B. Aufrufreihenfolgen) gelten können



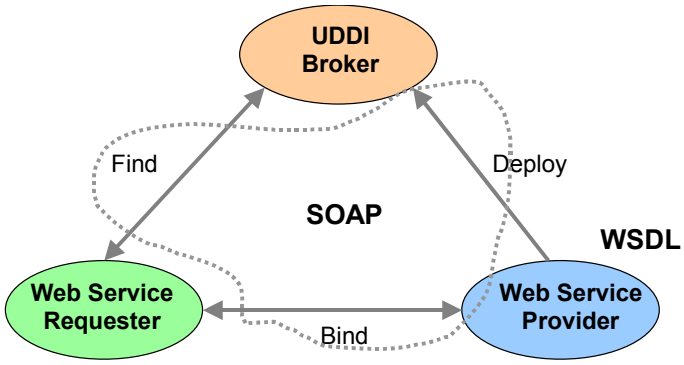
  - sollen in beliebigen Programmiersprachen implementiert und unter verschiedenen Systemumgebungen ablaufen können
  - sollen anspruchsvolle E-Business-Anwendungen durch (prozessorientierte) Komposition der Operationen eines oder mehrerer Web Services ermöglichen

Komponenten

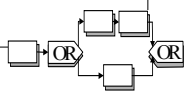
- Entwicklungs- und Laufzeitumgebung für die Implementierung bzw. Ausführung von Web Services (Service Bus) (siehe Kap. 3)
- Sprache(n) zur Beschreibung von Web Services
  - Service Description + (abstrakte) Beschreibung des Service-Typs (sog. Porttype), z.B. angebotene Operationen sowie deren Input-, Output- und Fehlernachrichten
  - Transport Binding + Bindung an konkrete Aufrufprotokolle (z.B. HTTP, SMTP), Nachrichtenformate (z.B. MIME, SOAP) und Endpunkte (z.B. URL, E-Mail)
  - öffentlich zugängliches Verzeichnis für die Registrierung bzw. den Zugriff auf Web-Services
  - Prozessdefinitionssprache zur Verknüpfung der Operationen eines oder mehrerer Web Services (für Public Flow und Private Flow)
- In diesem Umfeld zahlreiche, teilweise konkurrierende Standards und Standardvorschläge

SOAP, UDDI und WSDL

- Zusammenspiel von SOAP, UDDI und WSDL
- SOAP (Simple Object Access Protocol):
  - XSD-/XML-basiertes Protokoll für den Austausch von Nachrichten zwischen Service Requester und Service Provider
  - Service Requester / Provider und UDDI-Broker
  - verwendet als Aufrufprotokoll HTTP

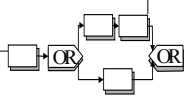


- UDDI (Universial Descritpion, Discovery and Integration)
  - Dienst für das Publizieren und Auffinden von Web-Services in öffentlich zugänglichen Verzeichnissen (Public UDDI)
  - Entwicklerschnittstellen: Publisher-API + Inquiry-API
  - UDDI-Business-Registry (White/Yellow/Green Pages)
  - von den Firmen Ariba, IBM und Microsoft forciert
  - auch als "private UDDI" nutzbar (mit z. T. erweiterbaren Funktionen)



WSDL (Web Service Description Language)

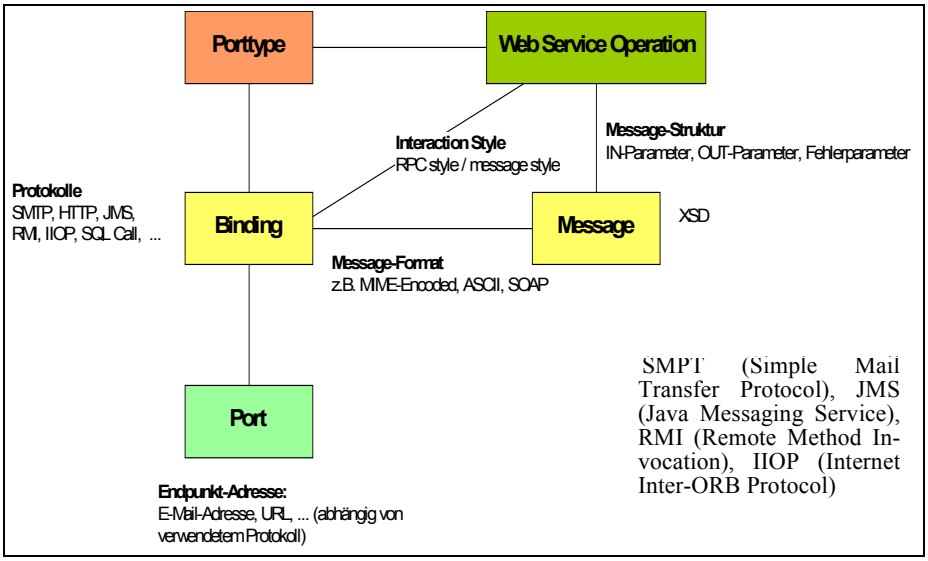
- Standardisierte XML-basierte Beschreibungssprache für Web Services
- Vergleichbar mit IDL in Corba
- Protokoll- und Programmiersprachenunabhängig
- Derzeit in Version 1.2 (März 2003; http://www.w3.org/TR/wsdl12/)
- Präsentiert ein Interface zu einem Dienst
- von Ariba, IBM und Microsoft bei der W3C eingereicht



WSDL (2)

Elemente

- *Message*: Abstrakte Definition der auszutauschenden Daten (typischerweise in Form von XML Elementen)
- *Operation*: Abstrakte Aktionen, die ein Service unterstützt (in anderen Worten: Menge von Input und Output Messages)
- *Port Type*: Menge von Operationen
- *Binding*: Beschreibt Abbildung eines Port Types auf konkretes Datenübertragungsprotokoll wie SOAP, HTTP etc.
- *Port*: Einzelner individueller „Endpunkt“ mit konkretem Binding, identifiziert durch eine Netzwerkadresse
- *Service*: Sammlung zusammengehöriger Ports



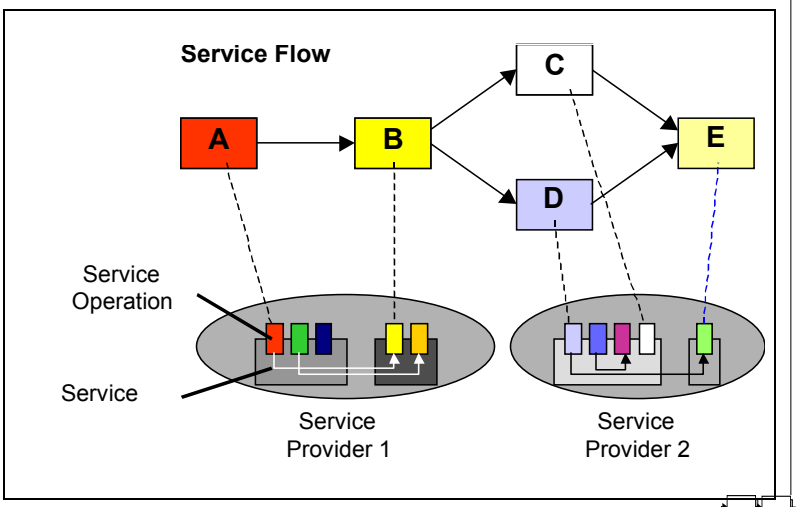
WSDL-Definition - Beispiel

```
<message name="CelsiusToFahrenheitRequest">
  <part name="Celsius" type="xsd:double" />
</message>
<message name="CelsiusToFahrenheitResponse">
  <part name="Return" type="xsd:double" />
</message>
<portType name="Temperature.ctemperaturePortType">
  <operation name="CelsiusToFahrenheit">
    <input message="tns:CelsiusToFahrenheitRequest" />
    <output message="tns:CelsiusToFahrenheitResponse" />
  </operation>
</portType>
```

Sprachen für E-Service (Web-Service) Workflows

- Trend: Realisierung von E-Anwendungen durch prozessorientierte Komposition von Web Services
  - Einbindung von Web Services eines oder mehrerer Provider
  - Wahl des konkreten Providers entweder statisch zur Entwicklungszeit oder dynamisch zur Laufzeit
  - Prozessorientierte Anwendungen können ihrerseits wieder als Web Service gekapselt werden

- Prozessorientierte Verknüpfung der Operationen eines oder mehrerer Web-Services erfordert Berücksichtigung von
  - Constraints einzelner Web Services (z.B. Einhaltung bestimmter Aufrufreihenfolgen für Operationen eines Service)
  - Constraints des Service Providers (z.B. Service-übergreifende Aufrufreihenfolgen für Operationen)



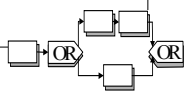
Sprachtypen

- Derzeit mehrere Vorschläge bzw. Implementierungen für *Service Flow Languages*
  - Web Service Flow Language (WSFL); IBM WebSphere
  - XLANG, Microsoft Biztalk
  - Business Process Execution Language for Web Services (BPEL4WS), Microsoft, IBM und Bea Systems
  - Business Process Modeling Language (BPML)

- Allerdings noch einige Fragen offen, etwa in Bezug auf
  - Transaktionen
  - Sicherheit
  - Lizensierungspolitik
  - ...

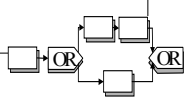
BPEL4WS

- XML-basierte Sprache zur Spezifikation des Verhaltens von Geschäftsprozessen im E-Service Kontext
- Basiert auf WSDL, XMLSchema und XPath
- Spezifikation vorgeschlagen von IBM, Microsoft, BEA
- BPEL4WS = IBM WSFL + Microsoft XLANG
- Derzeitige Version 1.1 (März 2003)
- Block-strukturierte Programmiersprache (Definitionen und Deklarationen auf oberste Ebene beschränkt)

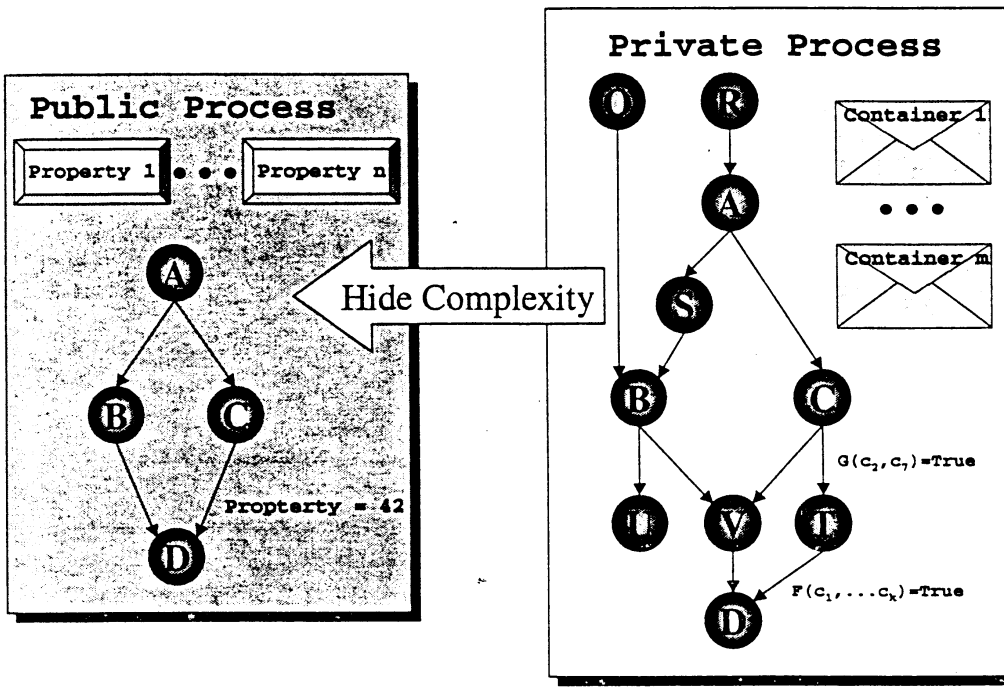


BPEL4WS: Basiselemente

- Elemente für Nachrichtenaustausch (*receive/reply*)
- *Activities* als grundlegende Bestandteile einer Prozessdefinition
- *Structured Activities* spezifizieren Reihenfolge der Aktivitäten
- Basissteuerung durch
  - sequence
  - switch
  - while
- Parallelität und Synchronisierung (*flow*)
- Non-deterministische Wahl (gesteuert durch externe Ereignisse) (*pick*)
- Instanzspezifische Prozessdaten werden in *containers* gespeichert
- Mechanismen für Fehlerbehandlung (*catching and handling faults*)

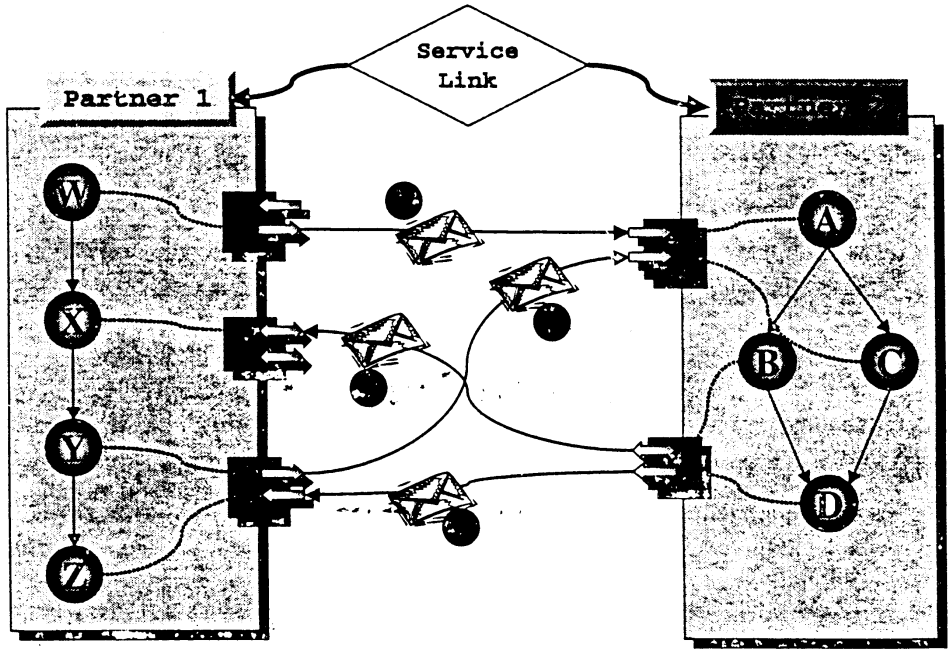


BPEL4WS: Abstrakte Prozesse



BPEL4WS: Prozesskommunikation

Business Protocols



BPEL4WS: "Top-Level"-Prozessattribute

```
<process name="ncname" targetNamespace="uri"
         queryLanguage="anyURI"?
         expressionLanguage="anyURI"?
         suppressJoinFailure="yes|no"?
         enableInstanceCompensation="yes|no"?
         abstractProcess="yes|no"?
         xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
```

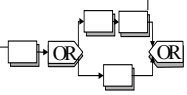
Legende:
A? - 0..1 Mal
A+ - 1..n Mal
A* - 0..n Mal

- queryLanguage. This attribute specifies the XML query language used for selection of nodes in assignment, property definition, and other uses. The default for this attribute is XPath 1.0, represented by the URI of the XPath 1.0 specification: http://www.w3.org/TR/1999/REC-xpath-19991116.

Alternative: XQuery

- expressionLanguage. This attribute specifies the expression language used in the process. The default for this attribute is XPath 1.0, represented by the URI of the XPath 1.0 specification: http://www.w3.org/TR/1999/REC-xpath-19991116.
- suppressJoinFailure. This attribute determines whether the joinFailure fault will be suppressed for all activities in the process. The default for this attribute is "no".
- enableInstanceCompensation. This attribute determines whether the process instance as a whole can be compensated by platform-specific means. The default for this attribute is "no".
- abstractProcess. This attribute specifies whether the process being defined is abstract (rather than executable). The default for this attribute is "no".

JoinFailure: Die Bedingung beim Zusammenführen von Aktivitäten ist nicht erfüllt



BPEL4WS: Nachrichtenaustausch

The <receive> construct allows the business process to do a blocking wait for a matching message to arrive.

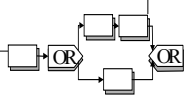
```
<receive partner="ncname" portType="qname" operation="ncname"
         variable="ncname" createInstance="yes|no"?
         standard-attributes>
  standard-elements
  <correlations>?
    <correlation set="ncname" initiate="yes|no"?>+
  </correlations>
</receive>
```

Warten dass Partner "ncname" die Operation "ncname" mit entsprechender Output-Message aufruft

The <reply> construct allows the business process to send a message in reply to a message that was received through a <receive>. The combination of a <receive> and a <reply> forms a request-response operation on the WSDL portType for the process.

```
<reply partner="ncname" portType="qname" operation="ncname"
       variable="ncname" faultName="qname"?
       standard-attributes>
  standard-elements
  <correlations>?
     <correlation set="ncname" initiate="yes|no"?>+
  </correlations>
</reply>
```

Schicken einer Reply-Nachricht vom Variablentyp "ncname" an Partner



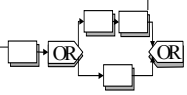
BPEL4WS: Korrelationen

- Konstrukt zur Identifikation des Vorgangs, Kunden etc.

Each correlation set in BPEL4WS is a named group of properties that, taken together, serve to define a way of identifying an application-level conversation within a business protocol instance. A given message can carry multiple correlation sets. After a correlation set is initiated, the values of the properties for a correlation set must be identical for all the messages in all the operations that carry the correlation set and occur within the corresponding scope until its completion. The semantics of a process in which this consistency constraint is violated is undefined. Similarly undefined is the semantics of a process in which an activity with the `initiate` attribute set to `no` attempts to use a correlation set that has not been previously initiated.

```
<correlationSets>?
  <correlationSet name="ncname" properties="qname-list"/>+
</correlationSets>
```

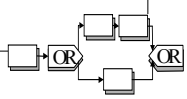
```
<!-- define correlation properties -->
<bpws:property name="customerID" type="xsd:string"/>
<bpws:property name="orderNumber" type="xsd:int"/>
<bpws:property name="vendorID" type="xsd:string"/>
<bpws:property name="invoiceNumber" type="xsd:int"/>
```



BPEL4WS: Aktivitätsaufrufe

The <invoke> construct allows the business process to invoke a one-way or request-response operation on a portType offered by a partner.

```
<invoke partner="ncname" portType="qname" operation="ncname"
        inputVariable="ncname" outputVariable="ncname"?
        standard-attributes>
  standard-elements
  <correlations>?
     <correlation set="ncname" initiate="yes|no"?
                  pattern="in|out|out-in"/>+
  </correlations>
  <catch faultName="qname" faultVariable="ncname"?>*
    activity
  </catch>
  <catchAll>?
    activity
  </catchAll>
  <compensationHandler>?
    activity
```



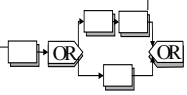
BPEL4WS: Kontrollflusselemente

The <sequence> construct allows you to define a collection of activities to be performed sequentially in lexical order.

```
<sequence standard-attributes>
  standard-elements
  activity+
</sequence>
```

The <switch> construct allows you to select exactly one branch of activity from a set of choices.

```
<switch standard-attributes>
  standard-elements
  <case condition="bool-expr">+
    activity
  </case>
  <otherwise>?
    activity
  </otherwise>
</switch>
```



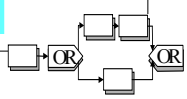
BPEL4WS: Kontrollflusselemente (2)

The <while> construct allows you to indicate that an activity is to be repeated until a certain success criteria has been met.

```
<while condition="bool-expr" standard-attributes>
   standard-elements
   activity
</while>
```

The <pick> construct allows you to block and wait for a suitable message to arrive or for a time-out alarm to go off. When one of these triggers occurs, the associated activity is performed and the pick completes.

```
<pick createInstance="yes|no"? standard-attributes>
  standard-elements
  <onMessage partner="ncname" portType="qname"
             operation="ncname" variable="ncname">+
    <correlations>?
       <correlation set="ncname" initiate="yes|no"?>+
    </correlations>
    activity
  </onMessage>
  <onAlarm (for="duration-expr" | until="deadline-expr")>*
    activity
  </onAlarm>
</pick>
```

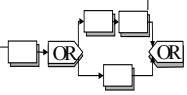


BPEL4WS: Kontrollflusselemente (3)

The <flow> construct allows you to specify one or more activities to be performed concurrently. Links can be used within concurrent activities to define arbitrary control structures.

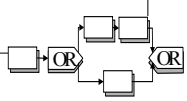
```
<flow standard-attributes>
  standard-elements
  <links>?
    <link name="ncname">+
  </links>

  activity+
</flow>
```

- Aktivitäten in einem Flow werden nebenläufig ausgeführt
- Zusätzliche Ausführungs-Constraints durch
  - Sequence-Konstrukt (strikte Reihenfolge)
  - Links (Synchronisation)

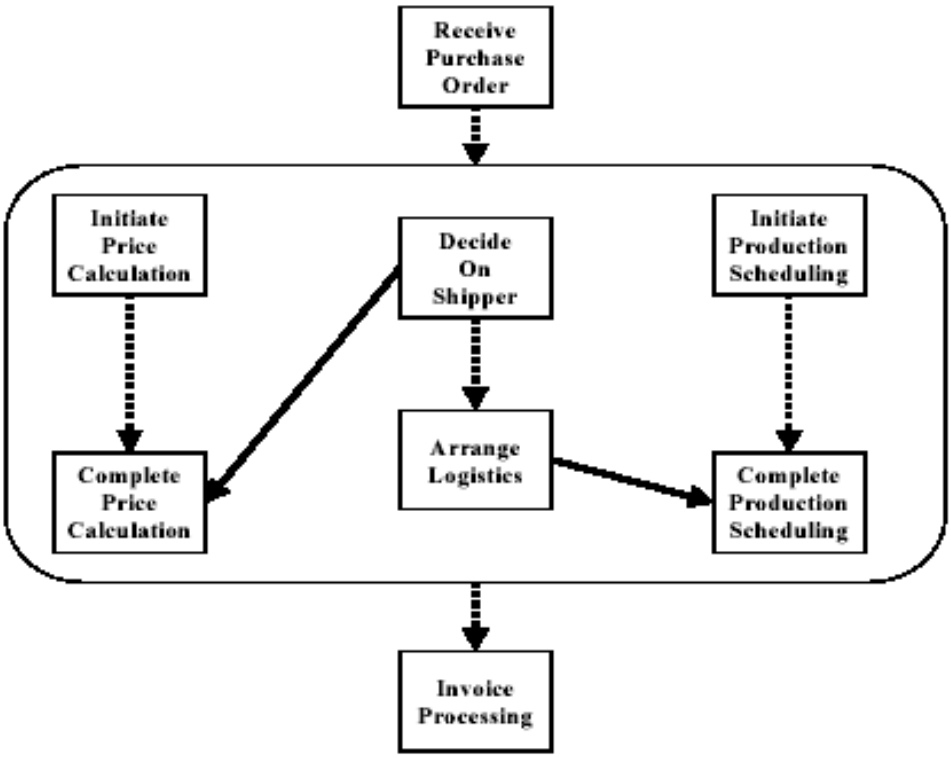


BPEL4WS: Links

- Jede Aktivität, die Ziel eines Links ist, hat ein implizites oder explizites joinCondition-Attribut (auch wenn nur ein eingehender Link vorhanden ist)
- Ohne Link-Spezifikation:
  - Aktivität startet, wenn ihre Vorgängeraktivität innerhalb der Sequenz beendet ist, oder wenn switch-, while-, oder pick-Bedingung erfüllt sind
  - Falls kein(e) Sequenz/Auswahl/Schleife/Pick definiert: Aktivität startet direkt nach Eintritt in <flow ...
- Bei eingehenden Links:
  - Die Bedingungen an allen eingehenden Kanten müssen erfüllt sein



BPEL4WS: Beispiel



BPEL4WS: Assoziierte WSDL-Definition - Beispiel

```
<definitions targetNamespace="http://manufacturing.org/wsdl/purchase"
      xmlns:sns="http://manufacturing.org/xsd/purchase"
…
<message name="POMessage">
   <part name="customerInfo" type="sns:customerInfo"/>
   <part name="purchaseOrder" type="sns:purchaseOrder"/>
</message>
…
<message name="scheduleMessage">
   <part name="schedule" type="sns:scheduleInfo"/>
</message>

<portType name="purchaseOrderPT">
   <operation name="sendPurchaseOrder">
      <input message="pos:POMessage"/>
      <output message="pos:InvMessage"/>
      <fault name="cannotCompleteOrder"
             message="pos:orderFaultType"/>
   </operation>
</portType>
…
<slnk:serviceLinkType name="purchaseLT">
   <slnk:role name="purchaseService">
       <slnk:portType name="pos:purchaseOrderPT"/>
   </slnk:role>
</slnk:serviceLinkType>
…
</definitions>
```

Messages

The WSDL portType offered by the service to its customer

Roles

[Source: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/]

BPEL4WS: Prozess-Definition - Beispiel (1)

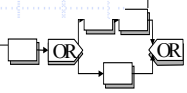
```
<process name="purchaseOrderProcess"
         targetNamespace="http://acme.com/ws-bp/purchase"
...
   <partners>
      <partner name="customer"
               serviceLinkType="lns:purchaseLT"
               myRole="purchaseService"/>
   ...
   </partners>

   <containers>
      <container name="PO" messageType="lns:POMessage"/>
      <container name="Invoice"
                 messageType="lns:InvMessage"/>
   ...
   </containers>

   <faultHandlers>
      <catch faultName="lns:cannotCompleteOrder"
             faultContainer="POFault">
         <reply   partner="customer"
                  portType="lns:purchaseOrderPT"
                  operation="sendPurchaseOrder"
                  container="POFault"
                  faultName="cannotCompleteOrder"/>
      </catch>
   </faultHandlers>
...
```

This section defines the different parties that interact with the business process in the course of processing the order.

This section defines the data containers used by the process, providing their definitions in terms of WSDL message types.

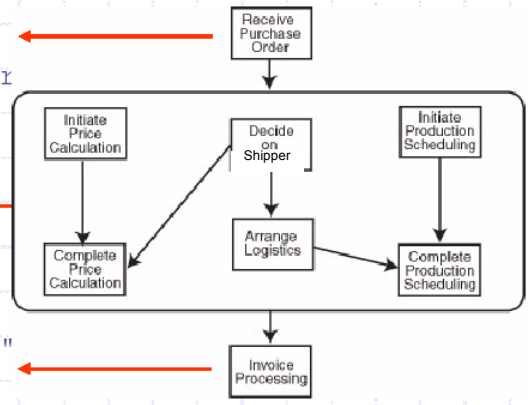
This section contains fault handlers defining the activities that must be executed in response to faults.

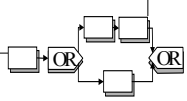
[Source: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/]



BPEL4WS: Prozess-Definition - Beispiel (2)

```
...
    <sequence>

      <receive partner="customer"

   portType="lns:purchaseOrderPT"
               operation="sendPurchaseOrder
               container="PO">
      </receive>

      <flow>
      ...
      </flow>

      <reply partner="customer"
             portType="lns:purchaseOrderPT"
             operation="sendPurchaseOrder"
             container="Invoice"/>
    </sequence>

</process>
```





BPEL4WS: Prozess-Definition - Beispiel (3)

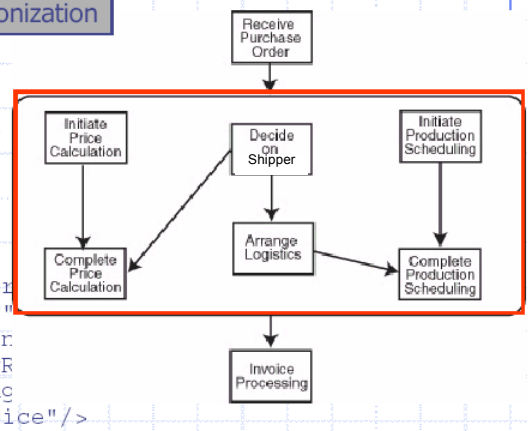
The flow construct provides concurrency and synchronization

Activities are executed sequentially

Activity Call

Activity call

```
<flow>
    <links>
        <link name="ship-to-invoice"/>
        <link name="ship-to-scheduling"/>
    </links>
    <sequence>
    ...
        <invoke  partner="shippingProvider
                 portType="lns:shippingPT"
                 operation="requestShippir
                 inputContainer="shippingR
                 outputContainer="shipping
            <source linkName="ship-to-invoice"/>
        </invoke>
        <receive partner="shippingProvider"
                 portType="lns:shippingCallbackPT"
                 operation="sendSchedule"
                 container="shippingSchedule">
            <source linkName="ship-to-scheduling"/>
        </receive>
    </sequence>
...
</flow>
```



[Source: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/]

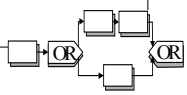
E-Service Flow Sprachen: Zusammenfassung

- Spezifikationssprachen für die Beschreibung des Kontroll- und Datenflusses bzgl. E-Services
- Vertreter: BPEL4WS
- Aufsplitterung des Kontrollflusses durch Skriptbasierte Notation
- Weiterführende Literatur

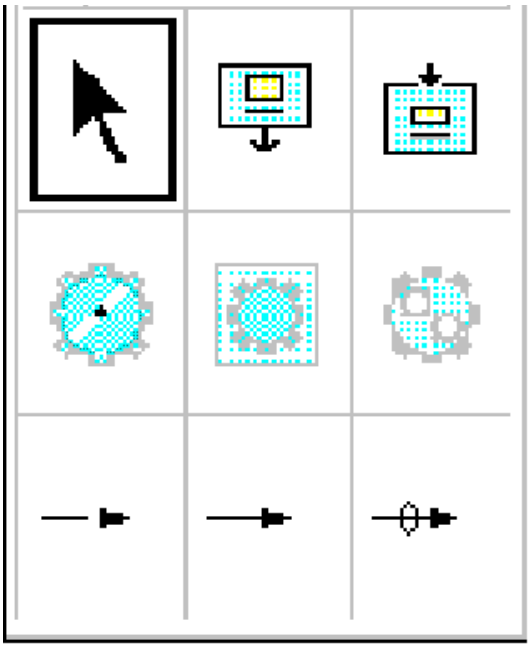
Literaturauswahl
F. Leymann, D. Roller, M.-T. Schmidt: *Web services and business process management*. IBM Systems Journal, 41(2):198-211, 2002.
S. Aissi, P. Malu, K, Srinivasan: *E-Business Process Modeling*: The Next Big Step. IEEE Computer, Mai 2002, S. 55 - 62.
http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
T. Frotscher: *Komposition von Web-Services mit WSFL*. JavaSpectrum, Heft 1, 2002, S. 28-33.
http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm (aktuell implementiert im Microsoft Bizztalk Server)
F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, S. Werawarana: *Business Process Execution Language for Web Services* (V 1.0), Initial Public Draft Release, 2002 (siehe http://www.ibm.com/developerworks/library/ws-bpel)
http://www.bpmi.org/bpml.esp

Flowmark Definition Language (FDL)

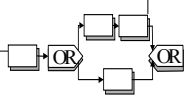
- Workflow-Definitionssprache des IBM Workflow-Management-Systems WebSphere MQ Workflow
- Programm-Aktivitäten, Prozeß-Aktivitäten, Blöcke
- Kontrollfluß
  - Kontrollfluß-Kanten mit Bedingungen, Default-Konnektoren
  - Schleifen nur über EXIT-Bedingungen (Repeat-Until Semantik)
- Datenmodellierung
  - Input- und Output-Container als logische Datenbehälter
  - geschachtelte Strukturen
- (Architektur von WebSphere MQ Workflow später)



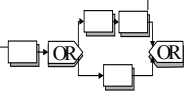
Flowmark Definition Language: Graphische Symbole



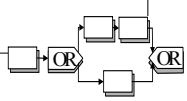
	Source	**Sink**
Programm-Aktivität	**Block**	**Prozeß-Aktivität**
Datenfluß	**Kontrollfluß**	**Default-Konnektor**



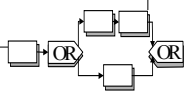
Fallbeispiel: FDL-Definition für *Klärung der Vorbedingungen*



Fallbeispiel: FDL-Definition für *Bearbeitung SHK-Antrag*

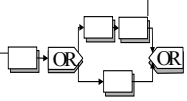


Fallbeispiel: FDL-Definition für *Bearbeitung Hauptvertrag*

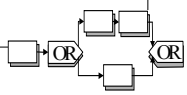


Fallbeispiel: FDL-Skript (Auszug)

```
/************************************************************************
* Generated by FlowMark Import/Export at 04/29/99, 09:23:35.
************************************************************************/
/************************************************************************
* DATA-STRUCTURES
************************************************************************/
STRUCTURE ‚Default Data Structure‘
END ‚Default Data Structure‘

STRUCTURE ‚Nachricht‘
  ‚Bezug_Auf_Nachricht‘:    STRING;
  ‚Gesendet_Am‘: STRING;
  ‚Inhalt‘:     STRING;
  ‚Antwort‘:    STRING;
  ‚Status‘:     STRING;
END ‚Nachricht‘
```



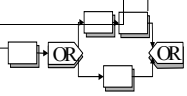
```
/*****************************************************************
* Description of Process Bearbeitung_Hauptvertrag
*****************************************************************/
PROCESS ‚Bearbeitung_Hauptvertrag‘ ( ‚Default Data Structure‘, ‚Default Data Structure‘ )
  LAYOUT   GIVEN
    WINDOW      ‚2063 12 152 497 320 4119 0 0 0 0 100 0 -247 4 1 4 ‚
  SOURCE   XPOS=-851   YPOS=450
  SINK     XPOS=1001   YPOS=-851

  PROCESS_ACTIVITY ‚Process_Prüfung_Unterlagen‘      ( ‚Nachricht‘, ‚Nachricht‘ )
    EXIT AUTOMATIC       WHEN ‚Status=‘‘Unterlagen sind korrekt‘‘‘
    LAYOUT      XPOS=-851   YPOS=199
  END ‚Process_Prüfung_Unterlagen‘
```



Workflow-Definitionssprachen: Zusammenfassung

- Sprachen für die Beschreibung des Kontroll- und Datenflusses bzgl. Aktivitäten
- Workflow-relevante Sprachklassen u.a.
  - Petri-Netze, State- und Activity-Charts, Proprietäre Produktsprachen (z.B. IBM FDL), BPEL4WS
- Bewertungskriterien u.a.
  - Notation (Skript-basiert und/oder graphisch)
  - Mächtigkeit bzgl. Kontrollfluß-Elementen (u.a Kantenbedingungen und temporale Aspekte)
  - Mächtigkeit bzgl. Datenfluß-Elementen
  - Strukturierung (hierarchische Workflows, Modularisierung), Analysierbarkeit (Verifikation)
- Weiterführende Literatur zu Workflow-relevanten Spezifikationssprachen

Sprachklasse	Literaturauswahl
Petri-Netze	Adam et al.: *Modeling and Analysis of Workflows Using Petri Nets*. Journal of Intelligent Information Systems 10(2): 131-158, March 1998.
	van der Aalst. *Verification of Workflow Nets*. In: P. Azema and G. Balbo (eds.). Application and Theory of Petri Nets 1997, volume 1248 of LNCS, 407-426. Springer-Verlag, Berlin, 1997.
State/Activity-Charts	Wodtke, D.: Modellbildung und Architektur von verteilten WFMS. Dissertation. Universität Saarbrücken, 1996.
Logikbasierte Sprachen	Bonner et al.: *An Overview of Transaction Logic*. Theoretical Computer Science (TCS), 133:205-265, 1994.
	Davulcu et al. *Logic-based modeling and analysis of workflows*. ACM Symposium on Principles of Database Systems (PODS), Seattle, WA, June 1998.



Workflow-Definitionssprachen: Zusammenfassung (2)

		Petri-Netze	State-Activity Charts	FDL	BPEL4WS
Notation		Graphisch	Graphisch	Graphisch + Skript	Skript
Kontrollfluss-mächtigkeit	Sequenz	Ja	Ja	Ja	Ja
	Nebenläufigkeit				
	Konditional				
	Schleife				
	Synchronisation				
Datenflussmächtigkeit		Rel./OO/OR Datenfluss nicht hierarchisch	RECORDS Datenfluss nicht hierarchisch	RECORDS	RECORDS
Trennung Kontrollfluss/Datenfluss		nur FUNSOFT	Strikt	Ja	Ja
Explizite Zustands-beschreibungen	Datenzustand	Ja	Nein	Nein	Nein
	Prozess-fortschritt	Nein	Ja	Nein	Nein
Ereigniskonzept		Nein	Ja	Nein	Ja
Organigramm-Elemente		nur FUNSOFT	Nein	Ja	Ja
Temporale Aspekte		Ja	Nein	Nein	Ja
Hierarchien, Modularisierung		Ja	Ja	Ja	Ja
Analysierbarkeit (Verifikation)		Ja	Ja	Nein	Nein