Seminar
WS 2002/2003

XML und Workflows

Kristof Barklage
barklage@rhrk.uni-kl.de

INHALTLICHE ZUSAMMENFASSUNG:

Innerhalb dieses Seminars werden verschiedene XML-basierte Workflow Definitionssprachen (XPDL, WSFL, BPML) in den Grundlagen vorgestellt. Anschließend erfolgt ein Vergleich dieser Sprachen und ein Ausblick in die Zukunft.

Inhaltsverzeichnis

Abkürzungsverzeichnis

BPMI	Business Process Management Initiative
BPML	Business Process Modelling Language
JDBC	Java Database Connectivity
SOAP	Simple Objekt Access Protocol
UDDI	Universal Description, Discovery and Integration
WAPI	Workflow Application Programming Interface
WfMC	Workflow Managament Coalition
WfMS	Workflow Management System
WSDL	Web Services Definition Language
WSEL	Web Services Endpoint Language
WSFL	Web Services Flow Language
XML	Extended Markup Language
XPDL	XML Process Definition Language

1. Grundlagen bzgl. Workflow

1.1. Definitionen

In der Literatur gibt es unterschiedlich eng bzw. weit gefasste Definitionen für Workflow. Übersetzt man den aus dem angelsächsischen Raum stammenden Begriff „Workflow“ wörtlich, bedeutet er im Deutschen „Fluss der Arbeit“. Abgeleitet aus dieser Übersetzung kommt man zu einem weit gefassten Verständnis von Workflow:

> *„Ein abgrenzbarer, arbeitsteiliger Prozess wird als Workflow bezeichnet“[1].*

Im Rahmen dieser Seminararbeit wird jedoch eine engere, für die Bearbeitung des Themas notwendige Definition für den Begriff Workflow verwendet:

> *„Ein Workflow stellt einen Arbeitsablauf oder Geschäftsprozess dar. Dieser Prozess wird mit Hilfe einer geeigneten Sprache formal beschrieben.“*

Aus dieser Formulierung lässt sich die Definition eines Workflow-Management-Systems ableiten:

> *„Ein Workflow-Management-System (WfMS) steuert und kontrolliert die Ausführung eines Workflows. Außerdem bietet es die Möglichkeit zur User-Interaktion sowie zur Administration des Workflows.“[2]*

Bei WfMS unterscheidet man wiederum zwischen autonomen und eingebetteten Systemen[3]. In einem autonomen System wird der Workflow von einer unabhängigen Software kontrolliert und gesteuert, bei Bedarf werden von diesem System weitere Applikationen aufgerufen. Bei einem eingebetteten System dagegen ist der Workflow in eine Software integriert und nur ein Bestandteil von dieser.
Ein Beispiel für einen Workflows ist Abb. 1-1:

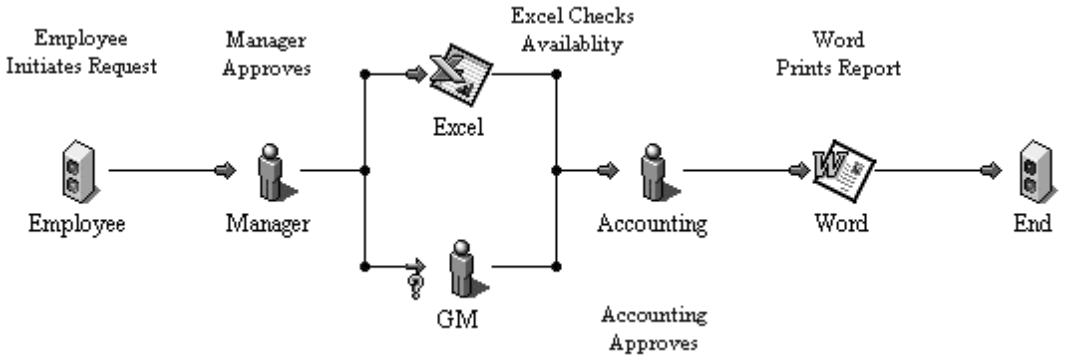

Abbildung 1-1: Grafisches Beispiel für einen Workflow

Die innerhalb des Workflows verwendeten Applikationen sind Microsoft Word und Microsoft Excel, der dargestellte Geschäftsprozess entspricht dem einer Bedarfsanmeldung. Das diesen Workflow steuernde Softwaresystem ist ein WfMS.

1.2. Entstehungsgeschichte

Die ersten Realisationen von WfMS gab es in den späten siebziger Jahren. Zu dieser Zeit wurde die Workflow-Technologie hauptsächlich genutzt, um das Einscannen von Bildern zu automatisieren und dieses anschließend den Nutzern zur Verfügung zu stellen[4].
In der jüngeren Vergangenheit führten viele Unternehmungen so genannte Business Reengineerings durch, um eine Optimierung der Abläufe innerhalb der Unterneh-

mung zu erreichen. Es wurde erkannt, dass der Einsatz von Workflows und dazugehörige Systeme eine Lösung war, die aus diesen Aktivitäten gewonnenen Erkenntnisse bestmöglich zu nutzen. Die Akzeptanz für WfMS hat infolgedessen und auch durch die zunehmende Bedeutung des Internets und darauf basierenden Technologien[5] zugenommen.

1.3. Vorstellung unterschiedlicher Konzepte

Im Rahmen der Weiterentwicklung der Workflow-Technologie haben sich unterschiedliche Ansätze und Konzepte hinsichtlich der Ausgestaltung eines Workflows herauskristallisiert. Die Merkmale der einzelnen Konzepte werden in diesem Teilabschnitt aufgezeigt.

1.3.1. Ad-hoc-Workflow[6]

Dieses Konzept besteht auf keinen festen Prozessablauf. Die durchzuführenden Arbeiten werden situationsabhängig von unterschiedlichen Ressourcen durchgeführt, es gibt unterschiedliche Ausgestaltungsmöglichkeiten bzgl. der Annahme oder auch Ablehnung einer auszuführenden Aufgabe. Häufig wird ein solches Konzept auf der Basis eines Messaging Systems verwirklicht.

1.3.2. Production Workflow[6]

Traditioneller Production Workflow unterstützt strukturierte und zeitkritische transaktionsorientierte Prozesse, die in großer Zahl verarbeitet werden und vorhersehbar sind.

1.3.3. Administrative Workflow[6]

Wie beim Production Workflow existiert hier ein vorhersehbarer Prozessablauf. Dieser Prozess kann jedoch u.U. durch Benutzereingriffe verändert werden, zusätzlich werden die durchzuführenden Aufgaben von den vorhandenen Ressourcen angefordert und die erarbeiteten Ergebnisse im Anschluss selbstständig zurückgemeldet.

1.3.4. E-Forms basierter Workflow[7]

Bei diesem Konzept werden die Prozesse über elektronische Formulare gesteuert. Die Formulare können für einfache Abläufe vorgefertigt oder individuell angepasst werden. Das WfMS steuert, wann welche Elemente zu sehen sind.

1.4. Die Workflow Management Coalition

Die Workflow Management Coalition (WfMC) wurde 1993 als Zusammenschluss von vielen Herstellern von WfMS, großer Anwender von WfMS sowie einiger Forschungsinstitute gegründet[4]. Ihr Ziel ist es, die Verbreitung von WfMS zu fördern, indem sie Standards für die Terminologie des Workflow Managements im allgemeinen und Schnittstellen von WfMS im speziellen entwickelt und etabliert werden. Die WfMC entwickelte ein Rahmenmodell[8] (Reference Model), in dem fünf Schnittstellen (Interfaces) definiert werden:

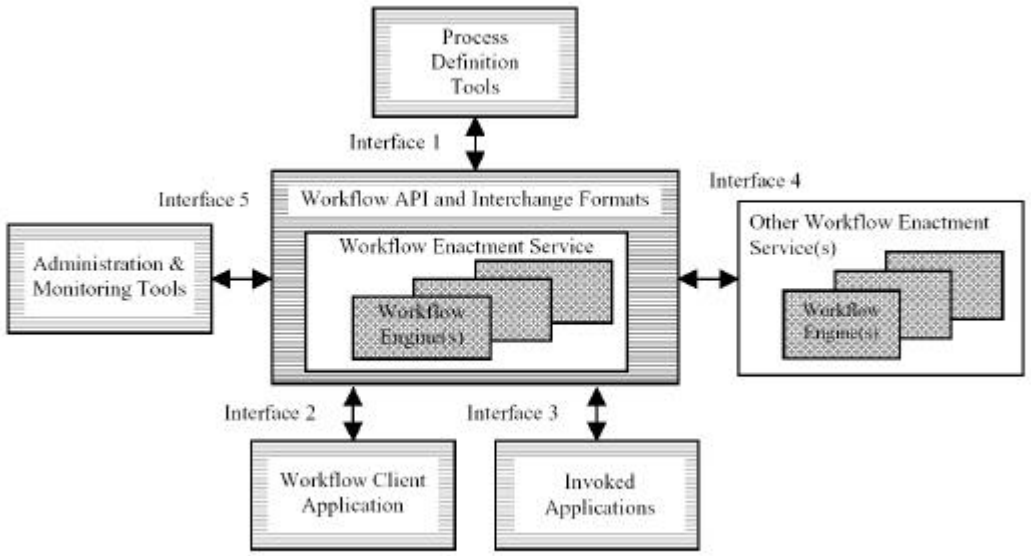

Abbildung 1-2: Reference Model of the WfMC

Eine kurze Beschreibung der unterschiedlichen Interfaces zum weiteren Verständnis folgt[4]:

- Interface 1: Prozessdefinition (Erzeugen, Anzeigen, Andern und Löschen von Workflow-Definitionen.
- Interface 2: Workflow-Client-Applikationen (Workflow Application Programming Interface (WAPI) zur Anbindung von Workflow-Client-Applikationen.)
- Interface 3: Eingebundene Anwendungen (Workflow aus Anwendungen heraus, Aufruf von Anwendungen innerhalb eines Workflows, direkte Integration von Workflow in Anwendungen).
- Interface 4: Interoperabilität verschiedener Workflow-Engines (Selektion, Initiierung und Ausführen von Prozessen auf anderen Workflow-Engines sowie Definition der Austauschformate für Daten).
- Interface 5: Workflow-Administration und Monitoring (Kontrolle der Prozessausführung, Datenbasis für die Analyse einzelner Prozesse)

Anzumerken ist, dass die Interfaces 2 und 3 mittlerweile zusammengefasst worden sind. Der Grund hierfür sind die relativ häufigen Überlappungen innerhalb dieser beiden Schnittstellen.

Dieses Modell wird innerhalb des Seminars als Referenzmodell bei dem Vergleich unterschiedlicher Workflowdefinitionssprachen und auch bei Verweisen auf Interfaces verwendet.

2. XML-basierte Workflow-Definitionssprachen

Innerhalb dieses Kapitels werden drei Sprachen erläutert, diese Auswahl basiert auf einem Dokument von Hollingworth[9]. Zuerst wird auf die XML Process Definition Language von der WfMC eingegangen. Im Anschluss daran wird mit der Web Services Flow Language von der IBM Software Group ein auf die Implementierung von Web Services ausgelegtes Konzept vorgestellt. Am Schluss dieses Kapitels werden die Besonderheiten der Business Process Modelling Language von der Business Process Management Initiative dargestellt, deren Version 1.0 allerdings noch die Merkmale einer Beta-Version hat.

2.1. XML Process Definition Language (XPDL)

Im Oktober 2002 ist von der WfMC die Version 1.0 des „XML Process Definition Language Standards“ veröffentlicht worden. Aufgrund der Aktualität des Standards kann man leider noch nicht auf Erfahrungsberichte aus der Praxis zurückgreifen, was eine Bewertung unmöglich macht. Mit der Hilfe dieser Definitionssprache soll das Interface 1 des Referenzmodells implementiert werden.
Die XPDL betreffenden Ausführungen basieren auf der Spezifikation der WfMC (Version 1.0)[10].

2.1.1. Modell und Zielsetzung

Als Zielsetzung hatte die WfMC ausgegeben, eine Definitionssprache zu schaffen, die es ermöglicht, einen einmal in XPDL definierten Arbeitsablauf in verschiedenen WfMS, die u.U. auf unterschiedlichen Plattformen implementiert sind, nutzen zu können.
XPDL stellt ein Grundgerüst einer Prozessdefinition dar, es werden nicht alle möglichen Anforderungen unterschiedlicher WfMS und Entwicklungstools unterstützt. Mit Hilfe des in Abbildung 2-1 zu sehenden Meta-Modells sollen die minimalen Anforderungen an eine Workflowdefinition abgedeckt werden. Es wurden für die Basisbausteine einer Workflowdefinition Entities und die Relationen zwischen diesen definiert.

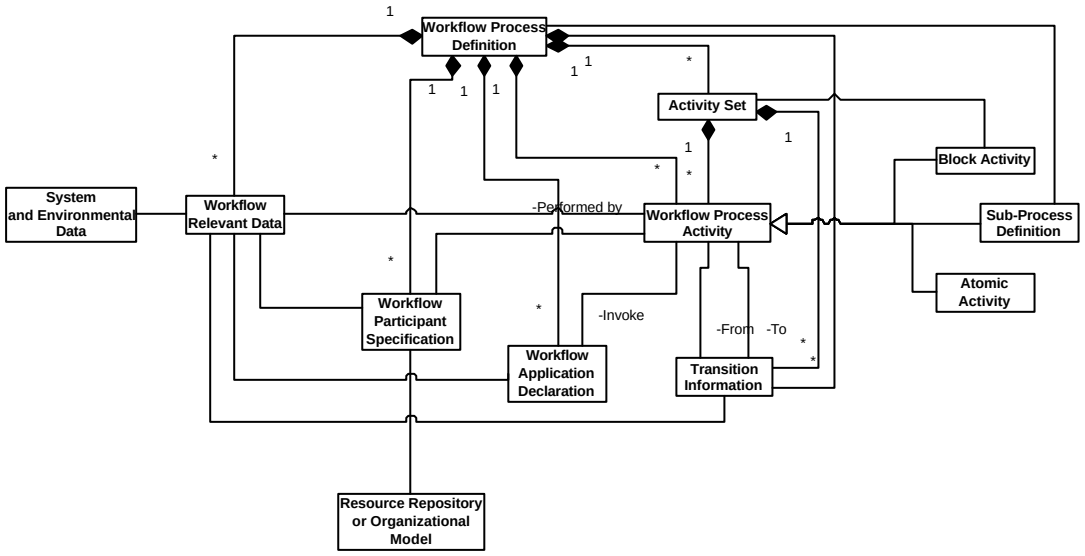

Abbildung 2-1: Meta-Modell von XPDL

2.1.2. Prozessdefinitionen

Als Oberklasse kann man im Meta-Modell von XPDL die Workflow Process Definition verstehen, die mit Hilfe des Entities `WorkflowProcess` implementiert wird. In diesem Entity werden alle Informationen, unabhängig ob Attribut oder Relation, gekapselt. In dem Quellcode werden neben der eindeutigen Identifikationsnummer `ID` und dem Namen `Name` die erlaubten und möglicherweise auch innerhalb der Prozessdefinition verwendeten Entities definiert, diese sind zum größten Teil mit einem selbsterklärenden Namen versehen.
Aus der gegebenen Menge möglicher Entities ist jedoch eine genauere Beschreibung des Entities `ActivitySet` notwendig. Innerhalb eines solchen Elements wird eine abgeschlossene Menge untereinander abhängigen und somit zusammengehörigen `Activity`-Objekten zusammengefasst.
Zu einer Instanz von `WorkflowProcess` gehört ein sogenannter Header, in dem alle für den Prozess relevanten Informationen gespeichert werden:

```
<xsd:element name="ProcessHeader">
       <xsd:complexType>
          <xsd:sequence>
             <xsd:element ref=“xpdl:Created" minOccurs="0"/>
             <xsd:element ref=“xpdl:Description" minOccurs="0"/>
             <xsd:element ref=“xpdl:Priority" minOccurs="0"/>
             <xsd:element ref=“xpdl:Limit" minOccurs="0"/>
             <xsd:element ref=“xpdl:ValidFrom" minOccurs="0"/>
             <xsd:element ref=“xpdl:ValidTo" minOccurs="0"/>
             <xsd:element ref=“xpdl:TimeEstimation" minOccurs="0"/>
          </xsd:sequence>
          <xsd:attribute name="DurationUnit">
             <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                   <xsd:enumeration value="Y"/>
                   <xsd:enumeration value="M"/>
                   <xsd:enumeration value="D"/>
                   <xsd:enumeration value="h"/>
                   <xsd:enumeration value="m"/>
                   <xsd:enumeration value="s"/>
                </xsd:restriction>
             </xsd:simpleType>
          </xsd:attribute>
       </xsd:complexType>
    </xsd:element>

    <xsd:element name="Created" type="xsd:string"/>

    <xsd:element name="Description" type="xsd:string"/>

    <xsd:element name="Limit" type="xsd:string"/>

    <xsd:element name="Priority" type="xsd:string"/>

    <xsd:element name="TimeEstimation">
       <xsd:complexType>
          <xsd:sequence>
             <xsd:element ref=“xpdl:WaitingTime" minOccurs="0"/>
             <xsd:element ref=“xpdl:WorkingTime" minOccurs="0"/>
             <xsd:element ref=“xpdl:Duration" minOccurs="0"/>
          </xsd:sequence>
       </xsd:complexType>
    </xsd:element>

    <xsd:element name="WaitingTime" type="xsd:string"/>
    <xsd:element name="WorkingTime" type="xsd:string"/>
    <xsd:element name="Duration" type="xsd:string"/>
    <xsd:element name="ValidFrom" type="xsd:string"/>
    <xsd:element name="ValidTo" type="xsd:string"/>
```

Es ist zu erkennen, dass innerhalb dieses Entities mit Hilfe der einzelnen Attribute Daten wie z.B. die Dauer, Gültigkeit oder Priorität des Prozessablaufes oder auch eine kurze Beschreibung von diesem definiert werden. Erwähnenswert ist in diesem Zusammenhang noch das Attribut `TimeEstimation`, hier werden für Simulationszwecke alle geschätzten Zeitangaben nochmals zusammengefasst. Zusätzlich zu diesem Header existiert noch ein weiterer, der `Workflow Process Redefinable Header`. Innerhalb dieses Headers werden verwaltungsrelevante Daten wie Autor, Version, Verantwortlicher oder auch Entwicklungsstatus (Mögliche Attribute: „In Überarbeitung“, „Testphase“, „Veröffentlicht“) der Prozessdefinition gespeichert.

2.1.3. Beschreibung einzelner Aktivitäten

Die WfMC unterscheidet zwischen fünf unterschiedlichen Typen von Aktivitäten:

- **Leere Aktivität** (`NONE Activity`): dieser Typ repräsentiert Aktivitäten, die nicht vom Computer kontrolliert werden und die Bestätigung der Durchführung von einem Benutzer gemeldet werden muss.
- **Generische Aktivität** (`Generic Activity`):dieser Typ steht für ein Element in einer sequentiellen Folge von Aktivitäten
- **Subworkflow** (`SUBFLOW Activity`): unter einem Subworkflow versteht man eine eigenständige Prozessdefinition, deren Durchführung zum Fortfahren der aktuellen Prozessdefinition notwendig ist.
- **Ablauf Aktitvität** (`Route Activity`): durch die Verwendung einer solchen Aktivität kann man Restriktionen einführen oder Verzweigungen innerhalb einer Sequenz von Aktivitäten implementieren. Verzweigungen können innerhalb von XPDL nur vom Typ `Join` (zwei oder mehr Aktivitäten sind Voraussetzung für die Fortsetzung der darauffolgenden) oder `Split` (nach der Ausführung einer Aktivität kann die Abarbeitung zwei oder mehrerer Aktivitäten erfolgen) sein.
- **Blockaktivität** (`BLOCK Activity`): mit Hilfe von diesem Typ wird zwei oder mehrere Teilaktivitäten zu einer einzigen zusammengefasst.

Veranschaulicht wird dieses Klassifikationsschema in Abbildung 2-2:

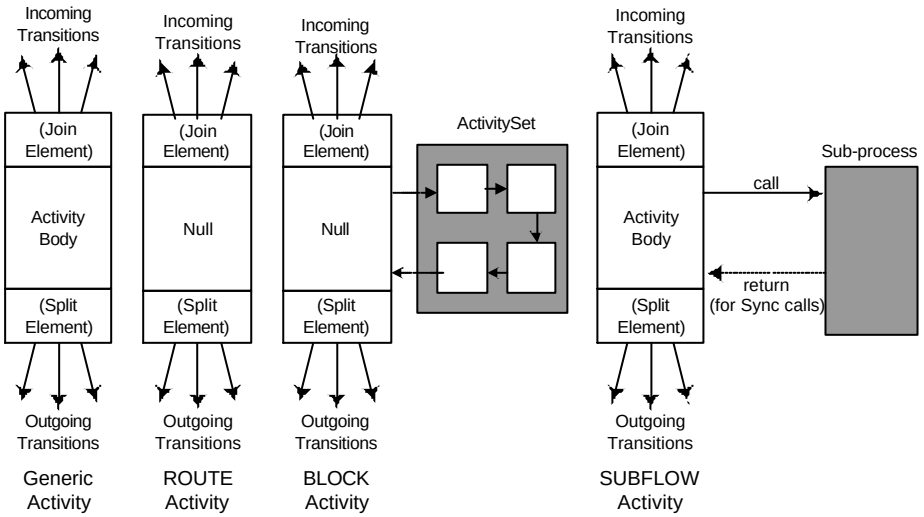

Abbildung 2-2: Klassifikationsschema von Aktivitäten innerhalb einer Prozessdefinition

Eine Aktivität wird mit Hilfe des Entities `Activity` definiert. Hier die dem Entity zugrundeliegende Struktur:

```
<xsd:element name="Activity">
   <xsd:complexType>
      <xsd:sequence>
```

```
                <xsd:element ref=“xpdl:Description" minOccurs="0"/>
                <xsd:element ref=“xpdl:Limit" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element ref=“xpdl:Route"/>
                    <xsd:element ref=“xpdl:Implementation"/>
                    <xsd:element ref="xpdl:BlockActivity"/>
                </xsd:choice>
                <xsd:element ref=“xpdl:Performer" minOccurs="0"/>
                <xsd:element ref=“xpdl:StartMode" minOccurs="0"/>
                <xsd:element ref=“xpdl:FinishMode" minOccurs="0"/>
                <xsd:element ref=“xpdl:Priority" minOccurs="0"/>
                <xsd:element ref="xpdl:Deadline" minOccurs="0"
maxOccurs="unbounded"/>
                <xsd:element ref=“xpdl:SimulationInformation" minOccurs="0"/>
                <xsd:element ref=“xpdl:Icon" minOccurs="0"/>
                <xsd:element ref=“xpdl:Documentation" minOccurs="0"/>
                <xsd:element ref=“xpdl:TransitionRestrictions" minOccurs="0"/>
                <xsd:element ref=“xpdl:ExtendedAttributes" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="Name" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
```

Unmittelbar in diesem Entity werden zum größtenteils Standarddaten wie z.B. textuelle Beschreibungen von Anfangs- und Endverhalten oder auch Informationen über die Art der Ausführung (*manuell* oder *automatisiert*) verwaltet.
Wie im vorigen Teil der Arbeit beschrieben, muss man zwischen dem Typ der zu definierenden Aktivität wählen. Zur Auswahl stehen die erwähnten Typen `Route` und `BlockActivity` sowie das Entity `Implementation`, was nachfolgend näher erklärt wird:

```
<xsd:element name="Implementation">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element ref=“xpdl:No"/>
                <xsd:element ref=“xpdl:Tool" maxOccurs="unbounded"/>
                <xsd:element ref=“xpdl:SubFlow"/>
            </xsd:choice>
        </xsd:complexType>
</xsd:element>
```

Innerhalb dieses Entities wird entschieden, ob eine leere Aktivität („`No`“) implementiert oder ein Subworkflow („`SubFlow`“) aufgerufen wird, oder ob eine Applikation („`Tool`“) zur Durchführung der benötigten Aktivität aufgerufen werden soll. Ein solcher Aufruf wird über das Interfaces 2 oder 3 des Referenzmodells ermöglicht, wobei innerhalb des Entities `Tool` die zur Durchführung der Aktivität benötigten Attribute definiert werden müssen, u.U. mit Hilfe des Entities `Extended Attribute`.

2.1.4. Transitionen

Die WfMC interpretiert einen Workflow als die Knoten und Kanten innerhalb eines Ablaufdiagramms, Objekte vom Typ `ActivitySet` stellen die Knoten, Objekte vom Typ `Transition` die Kanten dar.

```
<xsd:element name="Transition">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref=“xpdl:Condition" minOccurs="0"/>
                <xsd:element ref=“xpdl:Description" minOccurs="0"/>
                <xsd:element ref=“xpdl:ExtendedAttributes" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
```

```
            <xsd:attribute name="From" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="To" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="Name" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
```

Innerhalb einer Instanz von `Transition` muss die vorhergehende und die nachfolgende Aktivität definiert werden, dieses geschieht mit Hilfe der Attribute „`From`“ und „`To`“.
Innerhalb einer Instanz von `Condition` werden die Bedingungen an diese Transition definiert. Neben der Beschreibung der Bedingung(en) gibt es innerhalb dieses Entities vier Aktionsmöglichkeiten, die bei einer Verifizierung bzw. Falsifizierung von ihr(ihnen) eintreten können:

- **CONDITION**: die Transition wird bei Eintreten ihrer Bedingungen ausgeführt.
- **OTHERWISE**: die Transition wird durchgeführt, falls keine der Bedingungen zutreffen.
- **EXCEPTION**: die Transition wird durchgeführt, falls eine Exception auftritt und die gestellten Bedingungen erfüllt sind.
- **DEFAULTEXCEPTION**: die Transition wird durchgeführt, falls eine Exception auftritt und die an diese gestellten Bedingungen nicht erfüllt sind.

2.1.5. Definition teilnehmender Ressourcen

Mit Hilfe eines solchen Objektes kann man festlegen, was für Ressourcen für die Ausführung eines Workflows notwendig sind. Innerhalb der Definition von `Participant` selber gibt es keine Besonderheiten, jedoch wird das Entity `ParticipantType` instanziiert. Innerhalb einer Instanz dieses Entities wird festgelegt, was für einer Art die jeweilige Ressource entspricht. XPDL unterscheidet zwischen sechs unterschiedlichen Ressourcentypen, die je nach Bedarf verwendet werden können:

- `RESOURCE_SET`: eine Menge von Ressourcen.
- `RESOURCE`: explizite Nennung einer Maschine.
- `ROLE`: je nach Schwierigkeitsgrad werden Menschen mit unterschiedlicher Qualifikation zur Durchführung einer Arbeit benötigt. Eine solche Anforderung wird mit Hilfe dieses Typs ausgedrückt.
- `ORGANAIZATIONAL_UNIT`: eine Abteilung innerhalb einer Unternehmung.
- `HUMAN`: ein Mensch beliebiger Qualifikation.
- `SYSTEM`: eine selbstständig arbeitende Maschine, die vom WfMS kontrolliert wird.

2.1.6. Datenhaltung in XPDL

Als relevante Daten werden von der WfMS Variablen angesehen, die unmittelbar für die Durchführung eines Workflows benötigt werden. Diese Daten werden innerhalb der Workflowdefinition explizit berücksichtigt und mit Hilfe des Entities `DataField` verwaltet. Jedes Element von DataField hat eine bestimmten Datentyp. Die unterschiedlichen Datentypen werden durch das Entity `DataTypes` definiert. Eine kurze Erklärung der verfügbaren Standarddatentypen:

- `BasicType:` Integer, Float, String...
- `DeclaredType:` Möglichkeit zur Definition eines neuen Datentyps, der über eine Referenz wie ein standartisierter XPDL Datentyp genutzt werden kann
- `SchemaType:` Möglichkeit zur Definition neuer Datentypen, die explizit der XML Schema Syntax entsprechen müssen
- `ExternalReference:` Referenz auf einen externen Datentyp (Veraltet)

- `RecordType:` Referenz auf eine Aktennummer
- `UnionType:` Vereinigung mehrerer Variablen
- `EnumerationType:` Aufzählung verschiedener Werte
- `ArrayType:` Array eines bestimmten Datentyps
- `ListType:` Liste bestimmter Attribute eines Datentyps

2.1.7. Flexibilität in XPDL

XPDL ist darauf ausgelegt, bei Bedarf größtmögliche Flexibilität und Erweiterbarkeit implementieren zu können. Diese Agilität der Sprache wird dadurch ermöglicht, dass innerhalb des Standards von XPDL auch anwendungsspezifische Daten mit Hilfe des Entities `Extended Attribute` definiert werden können:

```
<xsd:element name="ExtendedAttribute">
      <xsd:complexType mixed="true">
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                  <xsd:any         processContents="lax"         minOccurs="0"
                  maxOccurs="unbounded"/>
            </xsd:choice>
            <xsd:attribute name="Name" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="Value" type="xsd:string"/>
            </xsd:complexType>
</xsd:element>
```

Dieses Entity kann in allen anderen Entities als Attribut verwendet werden, Arrays von diesem Element sind ebenso erlaubt. `Extended Attribute` besteht im Prinzip nur aus dem Identifier `Name` und dem Wert `Value`, es liegt somit eine sehr basische Implementierung vor, die den Grundstein für die Flexibilität von XPDL bildet.

2.2. Web Services Flow Language (WSFL)

Im Mai 2001 hat die IBM Software Group die Version 1.0 des WSFL-Standards veröffentlicht, der auf die Bedürfnisse von Web Services ausgelegt ist. WSFL hat somit einen engeren Fokus als XPDL und ist auf mehrere parallel ablaufende Instanzen ausgelegt.
IBM schreibt explizit, dass WSFL in einem Nutzungsverbund mit der Web Services Description Language (WSDL) zu sehen ist, da WSFL auf die durch WSDL beschriebenen Web Services zugreifen soll und daher Synergieeffekte zwischen den beiden Sprachen zu erreichen sind. Zusätzlich schlägt IBM die Entwicklung einer Web Services Endpoint Language (WSEL) vor, durch die das Interface von Endpunkten in dem Unternehmensverbund eines Web Services definiert werden soll. In einer möglichen Implementierung eines beliebigen Web Services würden beide vorhandenen Beschreibungssprachen dann mit dieser Neuentwicklung interagieren. Auf diese beiden Sprachen wird im Rahmen dieser Arbeit jedoch nicht weiter eingegangen.
Die WSFL betreffenden Ausführungen basieren auf der Spezifikation der IBM Software Group (Version 1.0)[11].

2.2.1. Zugrundeliegendes Meta-Modell

Die Spezifikation von WSFL basiert auf dem Modell einer virtuellen Unternehmung, die im Internet Bücher verkauft. Diese Unternehmung verschickt die Bücher allerdings nicht, sondern beauftragt einen Transporteur, die gewünschte Ware von einem Buchhändler abzuholen und sie dem Kunden zu überbringen. Abbildung 2-3 veranschaulicht eine mögliche Implementierungsart eines solchen Geschäftsprozesses:

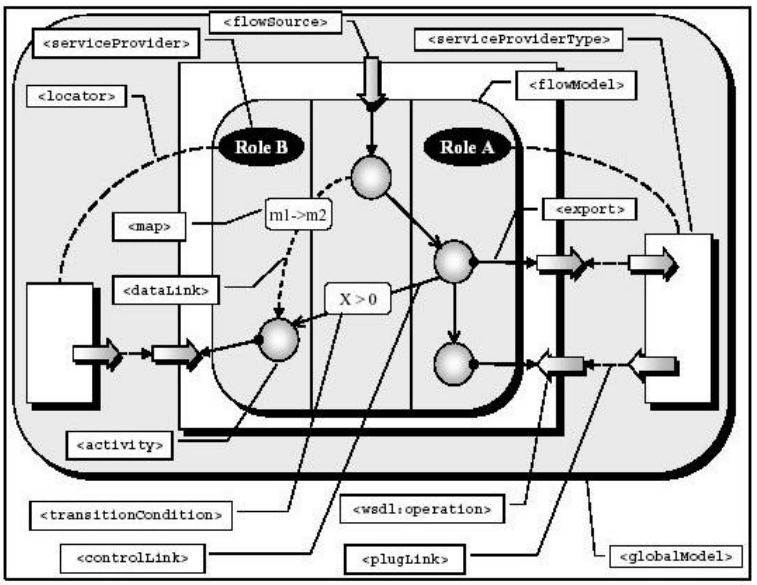

Abbildung 2-3: Beispielworkflow in WSFL

In dieser Grafik repräsentiert *Role A* eine Transportunternehmung und *Role B* eine Buchhändler, alle späteren Code-Beispiele basieren auf dem dargestellten Szenario. Die Formulierung *globaler Web Service* steht im Folgenden für einen solchen Verbund unterschiedlicher Leistungsanbieter von verschiedenen Web Services,

2.2.2. Definition eines Web Service

Die Definition eines Web Services beinhaltet eine Spezifikation der Interaktionen mit anderen Web Services innerhalb des Workflows. Zusätzlich wird das Interface für mögliche Zugriffe auf den implementierten Dienst durch andere Workflowinstanzen spezifiziert. Ein Beispiel:

```
<complexType name="flowModelType">
       <sequence>
              <element name="flowSource"
                     type="wsfl:flowSourceType"
                     minOccurs="0"/>
              <element name="flowSink"
                     type="wsfl:flowSinkType"
                     minOccurs="0"/>
              <element name="serviceProvider"
                     type="wsfl:serviceProviderType"
                     minOccurs=”0” maxOccurs="unbounded"/>
              <group ref="wsfl:activityFlowGroup"/>
       </sequence>
       <attribute name="name" type="NCName" use="required"/>
       <attribute name="serviceProviderType" type="Qname”/>
</complexType>

<group name="activityFlowGroup">
       <sequence>
              <element name="export" type="wsfl:exportType"
                     minOccurs="0" maxOccurs="unbounded"/>
              <element name="activity" type="wsfl:activityType"
                     minOccurs=”0” maxOccurs="unbounded"/>
              <element name="controlLink" type="wsfl:controlLinkType"
                     minOccurs="0" maxOccurs="unbounded"/>
              <element name="dataLink" type="wsfl:dataLinkType"
```

```
                    minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </group>
```

Ein in WSFL wohlgeformter Workflow nutzt u.a. folgende sechs Elemente:

- `flowSource` und `flowSink:` Eingangs- und Ausgangsknoten/-service des Workflowmodells.
- `serviceProvider:` Am Workflow teilnehmende Service Anbieter.
- `activity:` Aufzählung aller durch den Workflow in Anspruch genommenen externen Services.
- `controlLink` uns `dataLink:` Stehen für den Kontroll- und den Datenfluss innerhalb des Workflows.

Innerhalb eines `serviceProvider` Elements wird das externe Interface eines beliebigen Web Services beschrieben. Es werden die unterschiedlichen Ports angegeben, über die man die Dienste des jeweiligen Services in Anspruch nehmen kann. Der Großteil dieser Ports referenziert auf einen importierten Port eines externen Dienstanbieters. Ein solcher Port kann allerdings auch für einen „originären" WSFL Port stehen, auf diese Art und Weise kann auf einen lokal in WSFL definierten Workflow zugegriffen werden (*Subworkflow*).
Um die Funktion eines solchen Ports nutzen zu können, muss er eine eindeutige Adressierung besitzen; diese wird innerhalb einer Portbeschreibung über ein `locator` Element definiert. In WSFL gibt es vier unterschiedliche Bindungsarten zu am Service teilnehmenden Ressourcen:

- *static binding*: Zugriff auf bereits zuvor definiertes service Element das einen bestimmten Web Service repräsentiert
- *local binding*: Verbindung zu einer lokalen Softwarekomponente (z.B. JDBC)
- *UDDI binding*: Verbindung zu einer SOAP API
- *mobility binding*: Zugriff auf Daten, die bereits zuvor innerhalb des Workflows genutzt wurden

2.2.3. Prozessedefinitionen

Zur Beschreibung der unterschiedlichen Elemente im Flussgraphen eines Geschäftsprozesses werden in WSFL vier Elemente verwendet:

- `activities` (Knoten): Operation innerhalb des Workflows.
- `control links` (Kante): Den Kontrollfluss innerhalb des Workflows determinierende Kanten
- `data links` (Kante): Datenfluss zwischen den unterschiedlichen Aktivitäten innerhalb des Workflows
- `plug links` (Kante): Aufruf eines von einem **externen** Dienstanbieter angebotenen Web Services

2.2.4. Definition einer Aktivität

Die für eine einzelne Aktivität innerhalb eines Workflows stehenden `activity` Elemente haben im Vergleich zu XPDL semantisch keine substantiellen Unterschiede, auch hier gibt es die Möglichkeit die in einem Workflow üblichen Standardoperationen zu definieren: bspw. mehrere eingehende Kanten des Kontrollflusses in einer einzigen Operation zusammen zu synchronisieren. Da in WFSL zwischen dem Kontroll- und dem Datenfluss unterschieden wird, muss allerdings noch darauf explizit eingegangen werden. Es besteht die Möglichkeit mögliche Konflikte in den durch den Datenfluss übergebenen Attributen mittels einer Hierarchisierung von diesen zu un-

terbinden, z.B. eine Unterscheidung nach der Herkunft der unterschiedliche Werte in einer Variablen.

2.2.5. Trennung zwischen Kontroll- und Datenfluss

Die Elemente `controlLink` und `DataLink` werden gemeinsam vorgestellt, obwohl sie separat für den Kontroll- bzw. den Datenfluss innerhalb eines Workflows stehen. Jedoch existieren trotz dieser expliziten Trennung in der Regel Interdependenzen zwischen diesen beiden Flussarten, da der Kontrollfluss zumeist in Abhängigkeit zu vorherigen Ergebnissen steht und diese vom Datenfluss repräsentiert werden.
Ein Beispiel für einen `controlLink`:

```
<controlLink source="processPO"
             target="acceptSR"
             transitionCondition=”processPOOutput/x &gt; acceptSRInput.y”/>
```

Es werden immer Quell- und Zielknoten angegeben. Die `transitionCondition` entspricht einem Boolean-Wert, der angibt, ob diese Transition (Kante) durchgeführt werden soll oder nicht. Innerhalb eines `controlLinks` können bei Bedarf Fehlermeldungen erzeugt werden, die ein bestimmtes, vom Anwender vordefiniertes, Verhalten zur Folge haben.
Innerhalb eines `dataLinks` werden ebenfalls der Quell- und der Zielknoten angegeben:

```
<dataLink name="sup-shp1"
          source="processPO"
          target="acceptRequest">
      <map sourceMessage="anINVandSR" targetMessage="anSR"
           sourcePart=”SR” targetPart=”SR”/>
</dataLink>
```

Zusätzlich zu diesen beiden Referenzen, wird auf ein `map` Element verwiesen, durch das die Datenhaltung in WSFL implementiert wurde. Es können innerhalb eines `map` Elements zusätzliche Attribute benutzt werden; in diesem Beispiel sind dies `sourceMessage` und `targetMessage` die den zu WSDL konformen Teil der übergebenen Daten definieren, während `sourcePart` und `targetPart` dazu benutzt werden, auf ein bestimmtes Feld innerhalb dieses Teils zu referenzieren. Dies können bedarfsweise bei der Herkunfts- bzw. Bestimmungsoperation unterschiedliche Teile oder auch Felder sein, daher jeweils die Trennung in *source* und *target*.

2.2.6. Zugriff auf externe Services

`plugLinks` werden in WSFL genutzt, um entfernte Prozeduren aufzurufen. Mit Hilfe von `plugLinks` sind somit Zugriffe auf nicht lokal angebotene Web Services möglich. Diese entfernten Services benötigen ein mit WSFL konformes und öffentliches Interface, in dem u.a. definiert ist, ob Werte an die aufrufende Instanz zurückgegeben werden oder nicht. Neben der aufrufenden und der aufgerufenen Instanz wird innerhalb der Definition eines `plugLinks` eine Referenz auf ein `map` Element angegeben. Zusätzlich wird bei Bedarf die Bindungsart des nachgefragten Services spezifiziert, dies geschieht über ein bereits zuvor vorgestelltes `locator` Element.

2.2.7. Interfaces in WSDL

In WSDL unterscheidet man zwischen *internen* und *externen* Interfaces:
Mit der Hilfe eines *externen* oder auch öffentlichen Interfaces kann man von außen auf den durch WSDL definierten Workflow zugreifen.
Mit der Hilfe eines *internen* oder auch privaten Interfaces werden die Interaktionen zwischen dem Workflow bzw. Geschäftsprozess und den unterschiedlichen eingebundenen Dienstanbietern bezeichnet. Diese Dienstanbieter sind in einer virtuellen Unternehmung meistens nicht lokal, sondern verteilt implementiert, sie nutzen jedoch trotzdem das interne Interface.
Es läßt sich also sagen, dass durch ein *externes* Interface das **Interface 2** und durch ein *internes* Interface eine Mischung zwischen **Interface 3 und 4 des Referenzmodells** implementiert wird. Hierauf wird im Laufe des Seminars aufgrund des Themas nicht weiter eingegangen.

2.3. Business Process Modelling Language (BPML)

Diese Workflowdefinitionssprache ist im November 2002 von der Business Process Management Initiative (BPMI) in der letzten Version veröffentlicht worden und basiert auf XML. Diese Sprache dient zur Beschreibung von Geschäftsprozessen. Es wird das **Interface 1** des Referenzmodells implementiert; für eben dieses Interface sollten unterschiedliche Definitionssprachen untersucht werden.
Die BPML betreffenden Ausführungen basieren auf der Spezifikation der BPMI (Last-Call/Beta-Version)[12].

2.3.1. Aktivitäten in BPML

In BPML gibt es eine Vielzahl unterschiedlicher Aktivitätstypen, im Gegensatz zu den anderen Sprachen kann man eine Aktivität in BPML bei Bedarf in rekursiv definierte Subaktivitäten aufteilen. In den anderen vorgestellten Sprachen können solche Abläufe mit Hilfe von Prozessen und Subprozessen definiert werden. Die Konzeption einer Aktivität und deren Möglichkeiten werden durch Abbildung 2-4 verdeutlicht:

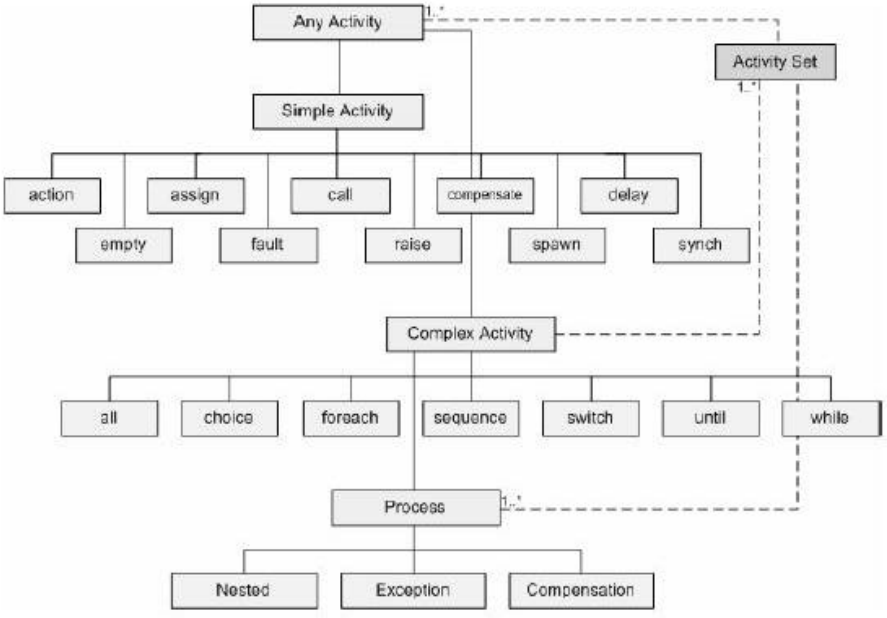

Abbildung 2-4: Klassifikationsschema unterschiedlicher Aktivitäten in BPML

Wie in der Abbildung zu sehen ist, unterscheidet BPML zwischen komplexen und simplen Aktivitäten. Eine simple Aktivität besteht aus einer atomaren Aktion und lässt sich nicht weiter zerteilen. Eine komplexe Aktivität hingegen kann man in mehrere

Teilaktivitäten unterteilen, bedarfsweise kann sie aus simplen oder rekursiv definierten komplexen Aktivitäten bestehen. Eine stichwortartige Beschreibung der unterschiedlichen Prozesstypen:

Simple Type

action	Führt eine Operation durch, die Eingangs- und Ausgangsdaten beinhaltet.
assign	Weist einer Variable einen neuen Wert zu.
call	Instanziiert einen Prozess und wartet auf dessen Beendigung.
compensate	Negativ eines Prozesses (kompensiert diesen).
delay	Repräsentiert Standzeit.
empty	Leere Aktivität.
fault	Erzeugt eine Fehler innerhalb der Umgebung.
raise	Gibt ein Signal.
spawn	Instanziiert einen Prozess und wartet nicht auf dessen Beendigung.
synch	Synchronisiert ein Signal.

Complex Type

all	Durchführung aller Aktivitäten parallel.
choice	Durchführung der in einer Teilmenge definierten Aktivitäten, determiniert durch ein vorheriges Ereignis.
foreach	Einmalige Durchführung aller für ein Element einer Liste definierten Aktivitäten.
sequence	Sequentielle Durchführung der Aktivitäten.
switch	Durchführung der in einer Teilmenge definierten Aktivitäten, determiniert durch den Wahrheitswert einer Umgebungsvariable.
until	Einmalige oder mehrmalige Durchführung der Aktivitäten, determiniert durch den Wahrheitswert einer Umgebungsvariable.
while	Keine oder mehrmalige Durchführung der Aktivitäten, determiniert durch den Wahrheitswert einer Umgebungsvariable.

2.3.2. Definition eines Prozesses

Unter einem Prozess wird eine komplexe Aktivität verstanden, die im Gegensatz zu einer solchen normalerweise eine eigene Laufzeitumgebung besitzt (auf den Sonderfall wird später eingegangen) und eine Menge von Aktivitäten bündelt.
Ein Prozess kann bei Bedarf Subprozesse besitzen und in BPML auf drei unterschiedliche Arten instanziiert werden. Zum einen kann er ins Leben gerufen werden, wenn das entsprechende Signal von einer beliebigen Aktivität gegeben wird, zum anderen aber auch, wenn ein prozessbezüglicher Input vorliegt. Als dritte Möglichkeit gibt es die direkte Instanziierung durch eine Aktivität oder einen vorliegenden Zeitplan. Alle drei Möglichkeiten können in BPML genutzt werden.
Falls ein Prozess in einer bereits vorhandenen Laufzeitumgebung ablaufen soll, so nennt man ihn einen *Nested Prozess.* Eine Berücksichtigung dieser Möglichkeit muss gewährleistet sein, da, wie schon erwähnt, Subprozesse definiert werden können.

Innerhalb einer Prozessdefinition werden neben den Standardvariablen (Name, Beschreibung des Prozesses, Eingangsparameter...) auch bspw. verschiedene Identitäten oder auch ein Kompensationsprozess definiert. Anhand einer Identität kann man einen Prozess bei mehreren Instanzen identifizieren, wohingegen der Kompensationsprozess dafür verwendet wird, die Wirkung des Prozesses nachträglich zu neutralisieren. Eine Beispielsyntax für eine Prozessdefinition:

```
<process
        name = NCName
        identity = list of QName
        persistent = boolean : false>
                Content: (documentation?, (event | parameters?),
                context?, {any activity}+, compensation?)
</process>

<event
        activity = list of NCName
        exclusive = boolean : false/>
```

Innerhalb des optionalen `event` Teils können Ereignisse definiert werden, die eine Abarbeitung der Prozessdefinition zur Folge haben.

2.3.3. Datenhaltung in BPML

Die Spezifikation geht nur auf direkt für den Workflow relevante Daten ein, wie mit externen Daten zu verfahren ist, wird nicht erwähnt. Dieses ist u.U. auf die Tatsache zurückzuführen, dass BPML nur in einer Beta-Version vorliegt.
Die Datenhaltung wird mit Hilfe von Laufzeitumgebungen, sogenannten `context` Elementen implementiert. Es besteht die Möglichkeit öffentliche wie lokale Daten zu definieren, eine Unterteilung der lokalen Daten in verschiedene, voneinander separate Datenbereiche wird auch unterstützt. Innerhalb dieser Datenbereiche ist eine hierarchische Baumstruktur implementiert, jedes Kindelement mit einem eigenen Datenbereich (bspw. eine Subaktivität) hat Zugriff auf die Daten des jeweiligen Vaterelements.
Zusammenfassend ist zu sagen, dass verschiedene Aktivitäten bzw. Prozesse mit Hilfe der Laufzeitumgebung miteinander kommunizieren und Daten austauschen können, durch den öffentlichen Bereich der Laufzeitumgebung wird vermutlich der Zugriff für externe Applikationen ermöglicht.

2.3.4. Transaktionen

In BPML gibt es drei unterschiedliche Transaktionsformen, die je nach Bedarf unter Befolgung der Restriktionen eingesetzt werden können:

- zwischen atomaren Aktivitäten: Eine Transaktion zwischen zwei oder mehreren atomaren Aktivitäten, als Sonderfall gibt es hier noch die Transaktion zwischen komplexen Aktivitäten. Diese Aktivitäten benötigen aber eine gemeinsame und private Laufzeitumgebung (in diesem Fall können sie wie eine atomare Aktivität behandelt werden).
- zwischen abgeschlossenen Prozessdefinitionen: Hier legt BPML ein besonderes Augenmerk auf die Datenkonsistenz, die hierfür verantwortliche Indikatorvariable wird erst bei gesicherter Konsistenz auf „wahr“ gesetzt und ein Fortschreiten im Rahmen des Workflows ermöglicht.

- zwischen zwei Transaktionen: Diese Möglichkeit ist implementiert worden, damit zwei parallel ablaufende Prozesse oder Aktivitäten während der Bearbeitung ihrer Aufgabe Informationen miteinander austauschen können.

3. Vergleichende Betrachtung, Fazit und Ausblick

Innerhalb diese Abschnitts sollen anhand der zuvor dargestellten Ergebnisse vorhandene Divergenzen sowie Konvergenzen zwischen den vorgestellten XML-basierten Workflowdefinitionssprachen beschrieben werden. Im Anschluss daran wird ein Fazit des Autors bzgl. der Realisationsmöglichkeiten in der Geschäftswelt erfolgen und die momentane Situation dargestellt. Außerdem enthält dieser Teilabschnitt einen kleinen Ausblick in die Zukunft.

3.1. Unterschiede zwischen XPDL, WSFL und BPML

Zu Beginn muss festgestellt werden, dass alle Sprachen eine unterschiedliche Zielgruppe besitzen. Während mit XPDL fast alle möglichen Arten eines Workflows implementiert werden können, ist dagegen WSFL nur für die Nutzung innerhalb von Web Services vorgesehen und dementsprechend spezialisiert. BPML hingegen ist auf die Definition und Kontrolle von Geschäftsprozessen ausgelegt, hat also erneut eine unterschiedliche Zielsetzung. Auf diesen Tatsachen basieren ein Großteil der Unterschiede.

Konzeptionell kann man XPDL und WSFL miteinander vergleichen, hier ist das Verständnis eines Workflows mehr oder weniger identisch: Prozesse sind in beiden Sprachen dafür da, mehrere atomare Aktivitäten zu kapseln; Transitionen gibt es jeweils zwischen Aktivitäten und Prozessen. Die Unterschiede zwischen diesen beiden Sprachen liegen eher im Detail.
Bei dem Entwurf von XPDL hat man aufgrund der durch die Zielsetzung benötigten Vielseitigkeit den Schwerpunkt auf eine höchstmögliche Flexibilität gelegt. Es ist möglich, verschiedene Ressourcetypen zu beschreiben und in den Workflow einzubinden, unabhängig davon, ob Mensch oder Maschine. Man kann in Verbindung mit dem Element `ExtendedAttribute` Ressourcen beliebigen Typs nach einmaliger Definition in einer Geschäftsprozessdefinition nutzen. Zusätzlich können mit diesem Element auch beliebige Informationstypen innerhalb des Workflows verwendet werden. Bei Bedarf ist es sogar möglich, neue Datentypen zusätzlich zu den Standarddatentypen zu definieren, die in einer XPDL Workflowdefinition anschließend wie originäre Datentypen behandelt werden können.
In WSFL dagegen sollen insbesondere mit WSDL beschriebene, automatisierte Ressourcen an einem Workflow teilnehmen können. Falls eine automatisierte Ressource nicht durch WSDL beschrieben werden kann, so kann sie nur mit Hilfe einer speziellen, immer neu zu entwerfenden und zu WSDL kompatiblen Schnittstelle in den Workflow miteinbezogen werden. Die Besonderheit von WSDL ist die Trennung von Daten- und Kontrollfluss, die mit Hilfe von `controlLinks` und `dataLinks` ermöglicht wird. Die Datenhaltung ist in WSFL mit Hilfe von `map` Elementen implementiert worden, auch hier ist eine weitreichende Flexibilität gegeben, externe Daten können jedoch nicht direkt in den Workflow eingebunden, sie müssen importiert werden.

Eine Sonderstellung nimmt Arbeit BPML ein. Hier liegt bzgl. der Definitionsmöglichkeiten eines Workflows ein großer Unterschied vor: Aktivitäten müssen in BPML nicht unbedingt atomar sein, sondern sie können auch komplex definiert sein, d.h., mehrere simple (atomare) oder, rekursiv definierte, komplexe Aktivitäten ineinander kapseln. Ein solcher Vorgang ist mit den anderen beiden Definitionssprachen nur durch Subprozesse auf Prozessebene zu realisieren, allerdings nicht auf der Ebene der einzelnen Aktivitäten.

Es existieren außerdem Unterschiede in der Datenhaltung, im Gegensatz zu den anderen Standards wird diese in BPML über eine Laufzeitumgebung realisiert, innerhalb der man noch weiter Laufzeitumgebungen definieren kann. Der Menge an Laufzeitumgebungen liegt eine hierarchische Baumstruktur zugrunde, wobei die Kindknoten immer Zugriff auf die Daten des Vaterknotens haben. Für den Geschäftsprozess nicht relevante Daten werden in der vorliegenden Version nicht explizit berücksichtigt.

3.2. Fazit und Ausblick

Im Rahmen des Fazits zeigt der Autor die Vor- und die Nachteile der vorgestellten Definitionssprachen auf. Im Anschluss daran folgt ein die Workflow-Technologie betreffender Zukunftsausblick unter Berücksichtigung der momentanen Situation.

3.2.1. Fazit

Alle drei Standards haben ihre Vor- und Nachteile, es ist eine situationsbedingte Entscheidung, welche Sprache zur Anwendung kommen sollte. Aus diesem Grund ist keine Bewertung der unterschiedlichen Workflow-Technologien enthalten. In der Realität wird ein Workflow zumeist sowieso mit Hilfe von mehreren WfMS unter Verwendung von unterschiedlichen Workflowdefinitionssprachen realisiert[13].

XPDL ist eine flexible Sprache, man kann sie in einer heterogenen Umgebung und auch in unterschiedlichen Anwendungsbereichen nutzen. Bei Bedarf ist es möglich, mehrere Abteilungen einer Unternehmung mit einer einzigen Workflowdefinition zu verbinden und somit eine zentrale Steuerungs- und Kontrollinstanz besitzen. Ein weiterer Vorteil ist die Tatsache, dass die WfMC exakt das Interface 1 des Referenzmodells implementiert hat. Das Referenzmodell ist international anerkannt und wird auch von anderen Herausgebern von Standards unterstützt, wenn nicht originär, dann durch virtuelle Interfaces. Auf diese Art und Weise hat die WfMC erreicht, dass XPDL zu fast allen existierenden Standards kompatibel ist.
Allerdings kann man bei XPDL auch sagen, dass sie als Ergebnis dem kleinsten gemeinsamen Nenner der Bedürfnisse aller Mitglieder der WfMC entspricht. Diese Tatsache bedeutet, dass bei einer möglichen Realisation in großem Maße Anpassungsarbeiten anfallen, um Besonderheiten des jeweiligen Workflows zu definieren. Diese Arbeit könnte bei einem auf den Bedarf des Kunden spezialisierten Standard geringer sein, falls dieser existiert.

Bei WSFL ist die Trennung von Kontroll- und Datenfluss u.U. ein Vorteil, insbesondere im Hinblick auf den vorgesehenen Anwendungsbereich. Bei Web Services ist es häufig der Fall, dass Daten- und Kontrollfluss divergieren. In solchen Situationen muss in WSDL keine generische Aktivität erzeugt werden, um die Daten zu transportieren.
Ein Nachteil ist jedoch die Ausrichtung auf von der IBM Software Group (z.B. WSDL) kontrollierte Standards. Eine Kompabilität mit anderen Standards bzw. anders beschreibeschriebenen Ressourcen ist nur unter großem Aufwand erreichbar, auch kann die menschliche Arbeitskraft nur schwer in einen Workflow miteinbezogen werden.

BPML hat im Gegensatz zu den anderen Standards nicht so starke Restriktionen was die Definition von Aktivitäten angeht. Diese Regelung könnte allerdings bei einer tatsächlichen Implementierung Missverständnisse hervorrufen, da rekursive Definitionen ab einer gewissen Tiefe schnell unübersichtlich werden.

Dieses Manko kann man jedoch auch als Vorteil ansehen. BPML stellt eine große, in sich geschlossene, Menge von Aktivitätstypen zur Verfügung. Mit diesem Grundgerüst ist es möglich, ohne allzu große Anpassungsmaßnahmen einen Geschäftprozess zu implementieren. Dieser Prozess muss dazu jedoch so kompatibel wie nur irgendwie möglich mit den Möglichkeiten von BPML sein.

3.2.2. Ausblick

Zum momentanen Zeitpunkt ist die Akzeptanz in der freien Wirtschaft für die vorgestellten Standards sehr gering. Der Workflow-Markt wird noch zum größten Teil von dem nicht auf einem der erwähnten Standards basierenden „*WebSphere MQ family*"-Lösungen von IBM dominiert[14]. Es gibt allerdings nach Meinung der Presse noch immer keinen weltweiten Standard[15].
Das ebenfalls von der IBM stammende WSFL wird noch nicht als marktfähig eingestuft und innerhalb der Softwarelösungen auch nicht verwendet. Es ist allerdings wahrscheinlich, dass WSFL langfristig innerhalb der *WebSphere MQ family* genutzt wird, um das Angebot von dieser Produktserie zu komplettieren. Diese Vermutung beruht auf der Tatsache, dass der für WSFL verantwortliche Entwickler Leiter der Workflow-Sparte bei der IBM ist.
Da die letzte Veröffentlichung von BPML im Prinzip noch eine Beta-Version war, gibt es hier noch keine Realisationen in der freien Wirtschaft; solche sind auch noch nicht angekündigt. Für das von der WfMC stammende XPDL kündigen kleinere Unternehmungen an[16], es in naher Zukunft implementieren zu wollen. Leider existieren noch keine Erfahrungsberichte.
Was die Zukunft angeht, bleibt es abzuwarten, in welche Richtung sich die Workflow-Technologie entwickeln wird. Da sich die Verwendung von XML[17] in anderen Branchen als sinnvoll erwiesen hat, könnte es allerdings sein, dass sich einer der vorgestellten Standards durchsetzt. Hilfreich könnte es hier sein, wenn sich einer der Global Players, z.B. SAP, für die Verwendung eines der vorgestellten Standards entscheiden würde.

Literaturangaben

[1] o.V., DM Dokumenten Management GmbH, 2001 (Zugriff am 13.2.2003)
http://www.dokumenten-management.de/dminfo/forscht/dip1/di121.htm
[2] o.V., Savant Technologies, 2002 (Zugriff am 13.2.2003)
http://www.savanttechnologies.com/savant/pdf%5CWfMS.pdf
[3] zur Muehlen, Universität Münster, 2001 (Zugriff am 11.12.2002)
http://attila.stevens-tech.edu/sigpam/workflow_tutorial/Design_of_Workflow_Applications.pdf
[4] Jablonski, Busseler; Workflow Management – Modeling Concepts, Architecture and Implementation; London, Bonn, International Thompson Computer Press, 1996
[5] o.V., Workflow Management Coalition, 1998 (Zugriff am 13.2.2003)
http://www.wfmc.org/standards/docs/Workflow_Internet_catalysts_for_change.pdf
[6] Plesums, Workflow Management Coalition, 2002 (Zugriff am 13.2.2003)
http://www.wfmc.org/information/introduction_to_workflow02.pdf
[7] o.V., Coextant Systems International AG, n.b. (Zugriff am 13.2.2003)
http://www.coextant.de/projectconsult/project.nsf/a86cc53e97d06313c1256a5a004d9512/41450b0ac2c1f2d841256b65005e2f42!OpenDocument
[8] o.V., Workflow Management Coalition, Stand 2003 (Zugriff am 13.2.2003)
http://www.wfmc.org/standards/model2.htm
[9] Hollingworth, Workflow Management Coalition, 2002 (Zugriff am 13.2.2003)
http://www.aboutworkflow.com/WorkflowHandbook2002/XML-based_architecture.pdf
[10] o.V., Workflow Management Coalition, 2002 (Zugriff am 13.2.2003)
http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf
[11] Leymann, IBM Software Group, 2001 (Zugriff am 13.2.2003)
http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
[12] o.V., Business Process Management Initiative, 2002 (Zugriff am 13.2.2003)
http://www.bpmi.org/specifications.esp
[13] o.V., Ultimus Corp., 2001 (Zugriff am 13.2.2003)
http://www.ultimus.com/ultwhite/wp_10_myths.pdf
[14] o.V., IBM, 2003 (Zugriff am 13.2.2003)
http://www-3.ibm.com/software/ts/mqseries/
[15] Kobielus, Network World Fusion, 2002 (Zugriff am 13.2.2003)
http://www.nwfusion.com/columnists/2002/0701kobielus.html
[16] , Bazancon, Workflow Management Coalition, 2003 (Zugriff am 13.2.2003)
http://www.wfmc.org/pr/Advantys_adopts_XPDL.pdf
[17] World Wide Web Consortium, XML (Zugriff am 13.2.2003)
http://www.w3c.org/XML/

Abbildungsverzeichnis