Vergleich von Prozessbeschreibungssprachen: BPEL vs. XPDL vs. jPDL

Martin Heinzerling

29. April 2009

Inhaltsverzeichnis

1 Einleitung **2**

2 BPEL **2**
2.1 Allgemein . . . 2
2.2 Bestandteile eines BPEL-Prozesses . . . 2
2.2.1 Primitive BPEL-Aktivitäten . . . 3
2.2.2 Strukturierte BPEL-Aktivitäten . . . 4
2.3 Gültigkeitsbereiche (Scopes) . . . 4
2.3.1 Ereignisbehandlung . . . 4
2.3.2 Fehlerbehandlung . . . 4
2.3.3 Kompensation . . . 4
2.4 Abstrakte BPEL-Prozesse . . . 4
2.5 Zusammenfassung . . . 5

3 BPEL4People **5**
3.1 Allgemein . . . 5
3.2 Generic Human Roles . . . 5
3.3 PeopleActivity . . . 5
3.4 Zusammenfassung . . . 7

4 BPMN **10**

5 XPDL **11**
5.1 Allgemein . . . 11
5.2 Metamodell . . . 11
5.3 Bestandteile eines XPDL-Prozesses . . . 12
5.3.1 ConformanceClass . . . 12
5.3.2 Pool . . . 12
5.3.3 Participants . . . 12
5.3.4 Activities . . . 12
5.3.5 Application . . . 12
5.3.6 Transitions . . . 12
5.4 Human Task . . . 13
5.5 Zusammenfassung . . . 13

6 jPDL **15**
6.1 Allgemein . . . 15
6.2 Bestandteile eines jPDL-Prozesses . . . 15
6.2.1 Swimlanes . . . 15
6.2.2 Knotentypen . . . 15
6.2.3 Transitionen . . . 15
6.2.4 Actions/Events . . . 15
6.2.5 Tasks . . . 16
6.3 Weitere Features . . . 16
6.4 Zusammenfassung . . . 16

7 Process Engines **16**

8 Zusammenfassung **18**

1 Einleitung

Die Arbeit entstand im Rahmen des Hauptseminars „Technische Informationssysteme“ der Technischen Universität Dresden. Der recht weit gefasste Titel stellt hier keinen Anspruch auf einen vollständigen Vergleich der drei Standards, sondern soll viel eher einen Überblick über den Einsatzbereich und die Möglichkeiten der Prozessbeschreibungssprachen/Prozessausführungssprachen bieten. Ein besonderes Augenmerk wurde dabei auf die Integration von „Human Task“ in den Prozess bzw. Workflow gerichtet.

2 BPEL

2.1 Allgemein

Die Business Process Execution Language ist eine auf XML-basierte Prozessausführungssprache. Seit April 2007 liegt sie in der Version 2.0 vor und trägt die Bezeichnung WS-BPEL. Damit gehört sie zu den WS-*-Spezifikationen.[1] WS steht in diesem Kontext für Webservices, also Komponenten die ggf. dezentral verteilt und gekapselt einzelne Aufgaben erfüllen. BPEL soll durch Orchestrierung (Abstimmung innerhalb eines Prozesses) und Choreographie (Abstimmung zwischen Prozessen) diese einzelnen Webservices in einen Geschäftsprozess einbetten und so die Wiederverwendbarkeit erhöhen. Dazu setzt BPEL auf der Interaktion mit WSDL-Beschreibung[2] der Dienste auf. BPEL selbst ist eine blockstrukturierte Programmiersprache, die auch Rekursion zulässt, somit aber zunächst Beschränkungen an zyklische Prozessgraphen stellt.

2.2 Bestandteile eines BPEL-Prozesses

Ein BPEL-Prozess lässt sich nach (WSBPEL, 2007, S. 18) schwerpunktmäßig in die in Listing 1 dargestellten Bestandteile gliedern.

Variablen Dieser Abschnitt enthält Variablen, die innerhalb eines Prozesses benötigt werden. Sämtliche Variablen sind auf den Gültigkeitsbereich (Scope) begrenzt, in dem sie deklariert wurden. (WSBPEL, 2007, S. 45ff)

[1] Kurzer Überblick unter: http://de.wikipedia.org/wiki/WS-*
[2] Details in: WSDL (2007)

```
<process name=... targetNamespace=... xmlns=... abstractProcess="no">
  <!--Definition und Rollen der Teilnehmer -->
  <partnerLinks/>
  <!--Erweiterungen, z.B. BPEL4People -->
  <extensions/>
  <!--Weitere Schemata//-->
  <import/>
  <!--Daten/Zustand des Prozesses -->
  <variables/>
  <!--Eigenschaften, die Konversationen definieren -->
  <correlationSets/>
  <!--Exception-Handling -->
  <faultHandlers/>
  <!--Fehler-Recovery - Aktionen rückgängig machen -->
  <compensationHandlers/>
  <!--Ereignisse mit eigenem Prozess -->
  <eventHandlers/>
  <!--Business Process Flow: Mindestens eine Activity -->
  <!--(Activities) -->
</process>
```

Listing 1: Grundgerüst BPEL-Dokument

Correlation Sets Correlation Sets werden in Prozessen, die mehrere <receive>- oder <pick>-Aktivität besitzen verwendet. Hier erfolgt eine Abbildung von eintreffenden Nachrichten auf laufende Prozessinstanzen. Die Instanzen werden dabei durch Eigenschaften (Properties) definiert. Z.B besteht eine Bestellung aus einer Kunden- und Bestellnummer.
<correlationSet name="PurchaseOrder"properties="cor:customerID␣cor:orderNumber"/>
(WSBPEL, 2007, S. 74ff)

Partner Links Partner Links sind Assoziationen zwischen einem Prozess und anderen Partnerprozessen und bilden somit die Kommunikation zu den aufzurufenden Webservices ab. Weiterhin werden den einzelnen Partnern dabei auch Rollen zugeordnet.(WSBPEL, 2007, S. 36ff)

Handler Die Fehler- u. Kompensationshandler sind einem Gültigkeitsbereich zugeordnet und behandeln Ereignisse, die diesen Scope betreffen.

Aktivitäten Kernelemente eines Prozesses sind seine Aktivitäten bzw. Aktionen, auf die im Folgenden näher eingegangen wird.

2.2.1 Primitive BPEL-Aktivitäten

Aktivitäten werden in der BPEL-Spezifikation in primitive und strukturierte Aktivitäten unterteilt. Erstere beinhalten neben dem Antworten <reply> auf bzw. dem Empfangen <receive> von synchronen Anfragen und den Aufruf von Webservices <invoke>, auch die Anweisungen <assign>, <exit>, <wait> und <empty>, deren Funktion unmittelbar aus der Bezeichnung folgt. Aktionen der Fehlerbehandlung <throw> bzw. <rethrow> (Throw in übergeordneten Scope), sowie der Wrapper <extensionActivity> für zukünftige bzw. eingebunden Erweiterungen fallen ebenfalls in diesen Bereich. (WSBPEL, 2007, S. 24ff, 84ff)

2.2.2 Strukturierte BPEL-Aktivitäten

Neben den primitiven Aktivitäten existieren auch strukturierte Aktivitäten. Diese definieren eine sequentielle (<sequence>) oder parallele (<flow>) Art der Abarbeitung. Mit der Aktivität <pick> kann auf eine bzw. mehrere Nachrichten oder Timeouts gewartet werden und der Prozesse in eine bestimmten Zweig weiter abgearbeitet werden. Hinzu kommen noch Aktivitäten die man aus den alltäglichen Programmiersprachen kennt (<if>, <while>, <repeatuntil> und <foreach>).(WSBPEL, 2007, S. 24ff, 98ff)

2.3 Gültigkeitsbereiche (Scopes)

Wie bereit angedeutet, werden in BPEL Prozesse in Gültigkeitsbereichen strukturiert und als transaktionale Einheit zusammengefasst. Innerhalb des Scopes können <faultHandlers>, <eventHandlers>, <compensationHandler>, <terminationHandler>, <correlationSets>, <messageExchanges>, <variables> und <partnerLinks> neu und lokal definiert werden. Auf einzelne Handler wird im Folgenden eingegangen.(WSBPEL, 2007, S. 115ff)

2.3.1 Ereignisbehandlung

Einem Gültigkeitsbereich zugeordneten Eventhandler können nachrichtenbasierte oder zeitliche Ereignisse untergeordnet werden. Innerhalb der <onEvent>- oder <onAlarm>-Blöcke können dann wieder die zuvor aufgezeigten Aktivitäten aufgerufen werden. Zeitliche Ereignisse können durch die Unterelemente <for> auf eine Zeitspanne oder <until> auf einen Zeitpunkt terminiert werden.(WSBPEL, 2007, S. 137ff)

2.3.2 Fehlerbehandlung

Fault Handler sind ebenfalls an einen Gültigkeitbereich gebunden und werden ggf. von diesem oder manuell durch <invoke> ausgelöst. Innerhalb des Handlers können beliebige <catch>-Anweisungen gefolgt von einer optionalen <catchAll>-Anweisung stehen. CatchAll ist hier gleichzusetzen mit einen default-Block in den gängigen Programmiersprachen. Sämtlich Catch/CatchAll-Blöcke können nun wieder Aktivitäten enthalten und springen nach Ausführung dieser an das Ende des Scopes. Kann eine Aktivität nicht ausgeführt werden, wird der Fehler an die nächsthöhere Ebene weitergereicht und führt ggf. zum Abbruch des kompletten Prozesses.(WSBPEL, 2007, S. 127ff)

2.3.3 Kompensation

Ein sehr interessantes Feature einer BPEL-Prozess-Spezifikation sind die Kompensationshandler. Diese bieten die Möglichkeit der Rückabwicklung von Aktivitäten, wenn innerhalb einer Transaktion ein Fehler auftritt. Innerhalb der <faultHandlers>-, <compensationHandler>- oder <terminationHandler>-Blöcke kann entsprechend eine <compensate>-Anweisung ausgeführt werden. Mit <compensateScope> kann man darüber hinaus auch noch den Scope bestimmen, der zurückgesetzt werden soll.(WSBPEL, 2007, S. 118ff)

2.4 Abstrakte BPEL-Prozesse

BPEL bietet auch die Möglichkeit der Spezifikation abstrakter und damit nicht ausführbarer Prozesse. Diese können zu Dokumentationszwecken eingesetzt werden oder als öffentlich zugängliche Version (unternehmens-)interner Prozessmodelle publiziert werden. Um ein ausführbarer Prozess zu sein, muss ein Prozess mindestens eine <receive>-oder <pick>-Aktivität enthalten.

2.5 Zusammenfassung

BPEL ist derzeit noch ganz klar der Standard in diesem Sektor, was grundsätzlich immer ein Vorteil ist. Transaktionen, Kompensation, Fehler-Handling und auch abstrakte Prozesse werden in diesem Umfang von keiner anderen Sprache unterstützt. Die entscheidende Nachteile liegen in der fehlenden nativen Unterstützung von Human Task, dem fehlendem Metamodell bzw. Roundtrip zwischen BPMN und BPEL und natürlich der Komplexität.

3 BPEL4People

3.1 Allgemein

Da BPEL zunächst nur auf Verknüpfungen zwischen Webservices angelegt ist, wurde 2007 eine Erweiterung für die Aufnahme von Human Task in Prozesse veröffentlicht. Vor der WS-BPEL Extension for People (BPEL4People) wurden menschliche Interaktionen gewöhnlich durch einen Webservice maskiert und über eine WSDL zugänglich gemacht.

BPEL4People greift die Spezifikation WS-HumanTask[3] auf und integriert diese über den Extensionhook in die BPEL-Prozesse. Dies ist sowohl in abstrakten als auch in ausführbaren Prozessen zulässig. Ein einfacher Beispielprozess ist in Listing 2 dargestellt. (BPEL4People, 2007, S. 8).

3.2 Generic Human Roles

WS-HT bringt eine sogenannte <genericHumanRole> mit, die innerhalb von BPEL4People genauer definiert wird und auf eine der folgenden Rollen abgebildet werden kann. Diese Rollen geben den zugeordneten Personen spezifische Rechte für den Zugriff auf den Prozess.(BPEL4People, 2007, S. 10f)

Process initiator Ein Initiator ist die Person, die einen Prozess auslöst. Das kann automatisiert erfolgen, andernfalls muss aber eine Person als Initiator zugewiesen werden.

Process stakeholders Ein Stakeholder ist eine Personen, die Einfluss auf eine Prozessinstanz hat. Dazu zählen z.B. Überwachen, das Weiterleiten von Aufgaben usw. Es muss eine Stakeholder geben, sonst wird der Initiator zum Stakeholder.

Business administrators Ein Administrator überwacht mehre Prozessinstanzen und kann administrativ auf diese zugreifen, z.B. die Überwachung von Deadlines. Gibt es keinen Businessadministrator über nimmt der Stakeholder diese Rolle.

Listing 3 zeigt eine mögliche Verwendung.(BPEL4People, 2007, S. 47)

3.3 PeopleActivity

Listing 4 zeigt den generellen Aufbau einer PeopleActivity. Daraus resultierend werden innerhalb der Aktivität vier Konstellationen(Abb. 1) für den Aufruf eines Human Task zur Verfügung gestellt. (BPEL4People, 2007, S. 16ff)

htd:task Ein HTD-Task erstellt dabei eine Inline definiere Aufgabe innerhalb der Aktivität. D.h. es erfolgt keine Verweis auf einen Task, sondern dieser wird vollständig innerhalb des Dokuments spezifiziert. (1)

[3]Details in WSHT (2007)

```
<bpel:process ...
  xmlns:b4p=".../BPEL4People"
  xmlns:htd=".../WS-HT">
  ...
  <!--Namensräume importieren//-->
  <bpel:extensions>
    <bpel:extension namespace="...BPEL4People" mustUnderstand="yes"/>
    <bpel:extension namespace="...WS-HT" mustUnderstand="yes"/>
  </bpel:extensions>
  <bpel:import importType="...WS-HT" .../>
  ...
  <!--WS-HumanTask Namespace Elements//-->
  <b4p:humanInteractions>
    <!--Personengruppen//-->
    <htd:logicalPeopleGroups/>
      <htd:logicalPeopleGroup name="..">...</htd:logicalPeopleGroup>
    </htd:logicalPeopleGroups>
    <!--Inline Task//-->
    <htd:tasks>
      <htd:task name="..">...</htd:task>
    </htd:tasks>
    <htd:notifications>
      <htd:notification name="..">...</htd:notification>
    </htd:notifications>
  </b4p:humanInteractions>
  <!--Abbildung People - Generic Human Roles//-->
  <b4p:peopleAssignments>
    <htd:genericHumanRole>
      <htd:from>.</htd:from>
     </htd:genericHumanRole>
  </b4p:peopleAssignments>
  ...
  <!--eigentliche Aktivitäten//-->
  <bpel:extensionActivity>
    <b4p:peopleActivity name=".." ... >... </b4p:peopleActivity>
  </bpel:extensionActivity>
  ...
</bpel:process>
```

Listing 2: Grundgerüst BPEL4People

localTask Lokale Tasks verweisen auf einen Human Task der über kein Webservice-Interface verfügt. (2/3)

remoteTask Remote Tasks verweisen alternativ dazu auf einen Human Task mit Webservice-Interface. (4)

Völlig analog wird mit den Notifications verfahren. Innerhalb der <scheduledActions> können noch verzögerte Ausführungen oder Ablaufzeiten der Aktivitäten spezifiziert werden. Eine detailliertere Betrachtung an dieser Stelle überschreitet aber den Rahmen der Arbeit. Abschließend zeigt Listing 5 noch einen einfachen Inline Human Task, der den Sachverhalt kurz zusammenfasst.(BPEL4People, 2007, S. 20)

```
<b4p:peopleAssignments>
  <b4p:processStakeholders>
     <htd:from logicalPeopleGroup="group1">
       <htd:argument name="..."> ... </htd:argument>
     </htd:from>
  </b4p:processStakeholders>
  <b4p:businessAdministrators>
    <htd:from> <htd:literal>
    <htd:entity xsi:type="htd:organizationalEntity">
      <htd:users>
        <htd:user>Anne</htd:user>
        <htd:user>Paul</htd:user>
        <htd:user>Mary</htd:user>
      </htd:users>
    </htd:entity>
    </htd:literal> </htd:from>
  </b4p:businessAdministrators>
</b4p:peopleAssignments>
```

Listing 3: Beispiel <peopleAssignment>

3.4 Zusammenfassung

BPEL4People übernimmt selbstverständlich die Vor- und Nachteile von BPEL. Die Einbettung des ebenfalls sehr umfangreichen Standards WS-HT erhöht die Komplexität aber nochmals deutlich.(Abb. 2)

```
<bpel:extensionActivity>
  <b4p:peopleActivity name="..."
     inputVariable="..."
     outputVariable="..."
     isSkipable="xsd:boolean"
     ...>
     ...
     <!--Alternativ//-->
     ( <htd:task>...</htd:task>
     | <b4p:localTask>...</b4p:localTask>
     | <b4p:remoteTask>...</b4p:remoteTask>
     | <htd:notification>...</htd:notification>
     | <b4p:localNotification>...</b4p:localNotification>
     | <b4p:remoteNotification>...</b4p:remoteNotification>
     )
     <b4p:scheduledActions> ... </b4p:scheduledActions>
     ...
  </ b4p:peopleActivity>
</bpel:extensionActivity>
```

Listing 4: Grundgerüst <peopleActivity>

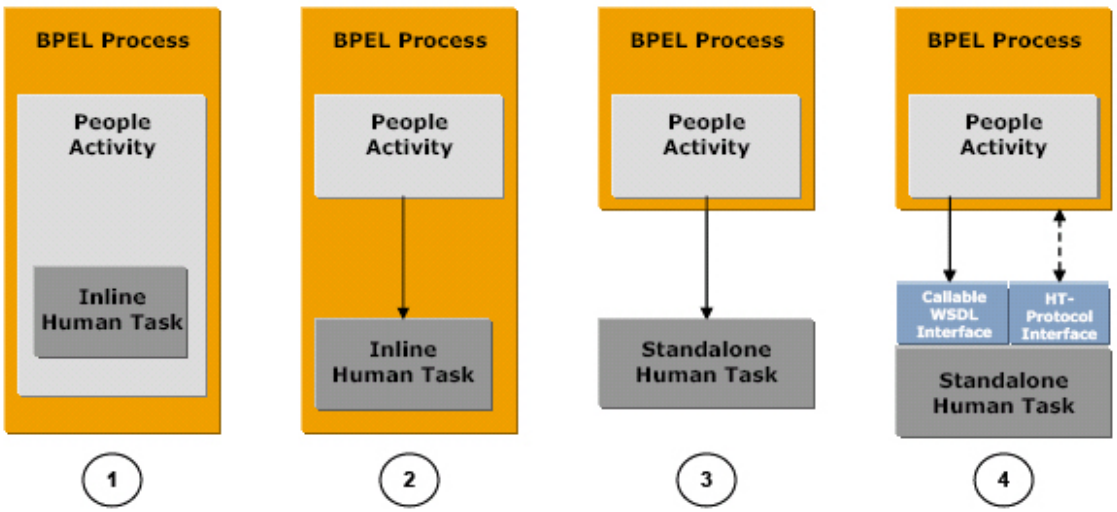

Abbildung 1: Konstellationen von Tasks in BPEL4People

```
<b4p:peopleActivity
  inputVariable="candidates"
  outputVariable="vote"
  isSkipable="yes">
  <htd:task>
    <htd:peopleAssignments>
      <htd:potentialOwners>
        <htd:from>$voters/users/user[i]</htd:from>
      </htd:potentialOwners>
    </htd:peopleAssignments>
  </htd:task>
  <b4p:scheduledActions>
    <b4p:expiration>
      <b4p:documentation xml:lang="en-US">
        This people activity expires when not completed
        within 2 days after having been activated.
      </b4p:documentation>
      <b4p:for>P2D</b4p:for>
    </b4p:expiration>
  </b4p:scheduledActions>
</b4p:peopleActivity>
```

Listing 5: Beispiel Inline Human Task

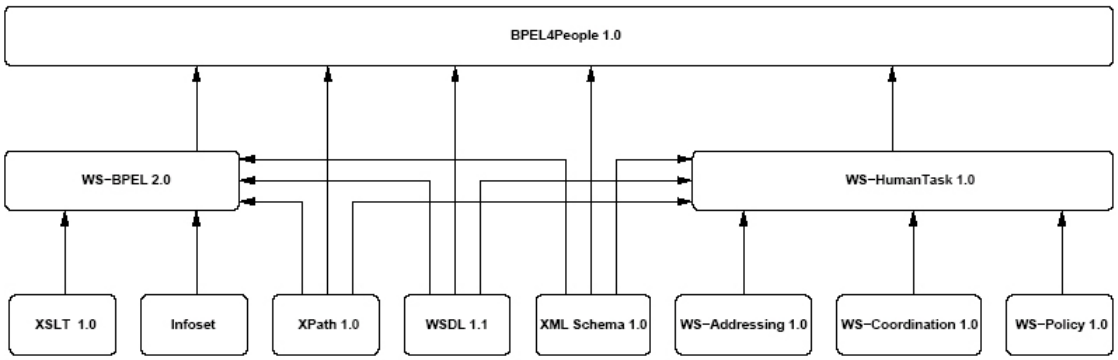

Abbildung 2: WS-Hierachie

4 BPMN

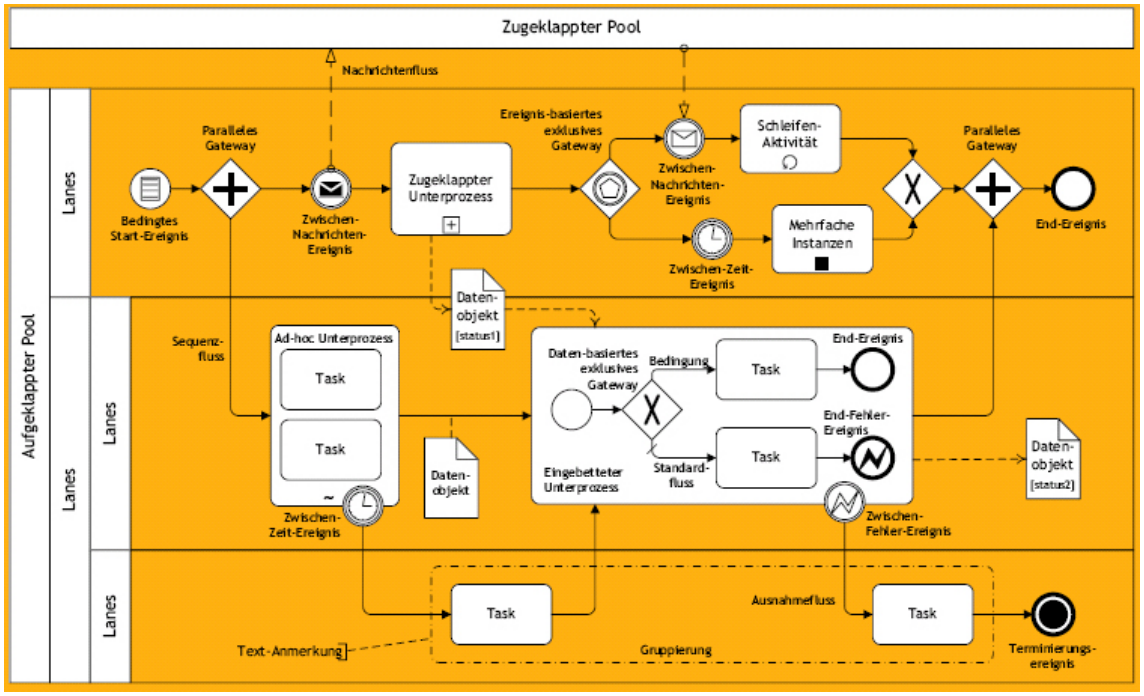

Abbildung 3: Übersicht BPMN

Aus den letzten beiden Abschnitten dürfte ersichtlich geworden sein, dass BPEL in seiner XML-Repräsentation äußert anspruchsvoll und damit auch fehleranfällig ist. Im täglichen Betrieb wird daher meist zunächst zu der Business Process Modeling Notation gegriffen. Nach einer ausführlichen Modellierung, auch mit Fachexperten aus den Geschäftsbereichen, kann eine semiautomatische Generierung von BPEL-Prozessen erfolgen. Die Abbildung 3 zeigt einen Überblick über Syntax und Semantik der BPMN und soll an dieser Stelle auch genügen. Die gesamte Abbildung ist unter [4] zu finden. Ein schönes Einstiegstutorial bekommt man unter [5]

Da BPEL zunächst nur als Ausführungssprache vorgesehen war, ist das Mapping zwischen BPMN und BPEL nicht bidirektional. Diesen Sachverhalt greift die im nächsten Abschnitt vorgestellt Sprache XPDL auf.

[4] http://bpt.hpi.uni-potsdam.de/pub/Public/BPMNCorner/BPMN1_1_Poster_DE.pdf
[5] http://www.bpmn.org/Documents/OMG%20BPMN%20Tutorial.pdf

5 XPDL

5.1 Allgemein

Die XML Process Definition Language ist, wie der Name schon vermuten lässt, ebenfalls eine auf XML basierte Ausführungssprache. Seit 1993 wird XPDL von der Workflow Management Coalition (WfMC) entwickelt und liegt derzeit in der Version 2.1 vom Oktober 2008 vor. Im Gegensatz zu BPEL liegt der Schwerpunkt aber nicht auf Interaktion von Webservice, vielmehr sollte bei der Entwicklung auch ein Austauschformat für diverse Prozesssprachen entstehen[6]. Kernelement dieser Forderung ist die vollständige Unterstützung der grafischen Business Process Modeling Notation (BPMN). Somit ist XPDL auch eine transitionsbasierte Sprache(XPDL, 2008, S. 8f)

5.2 Metamodell

Um das Mapping zwischen XPDL und BPMN zu vereinfachen wurde XPDL auf einen gemeinsames Metamodell aufgesetzt. Abbildung 4) (XPDL, 2008, S. 18ff) stellt dieses in einer Übersicht dar. Das Metamodell enthält die für die BMPN typischen Swimlanes und Pools, die einzelne Akteure gruppieren bzw. voneinander abgrenzen. Aktivitäten, Transitionen und viele weitere Elemente sind direkt an der BMPN angelehnt. Interessant in diesem Modell, ist noch die Möglichkeit, eine Blockaktivität (im Sinne von BPEL-Blöcken) verwenden zu können.

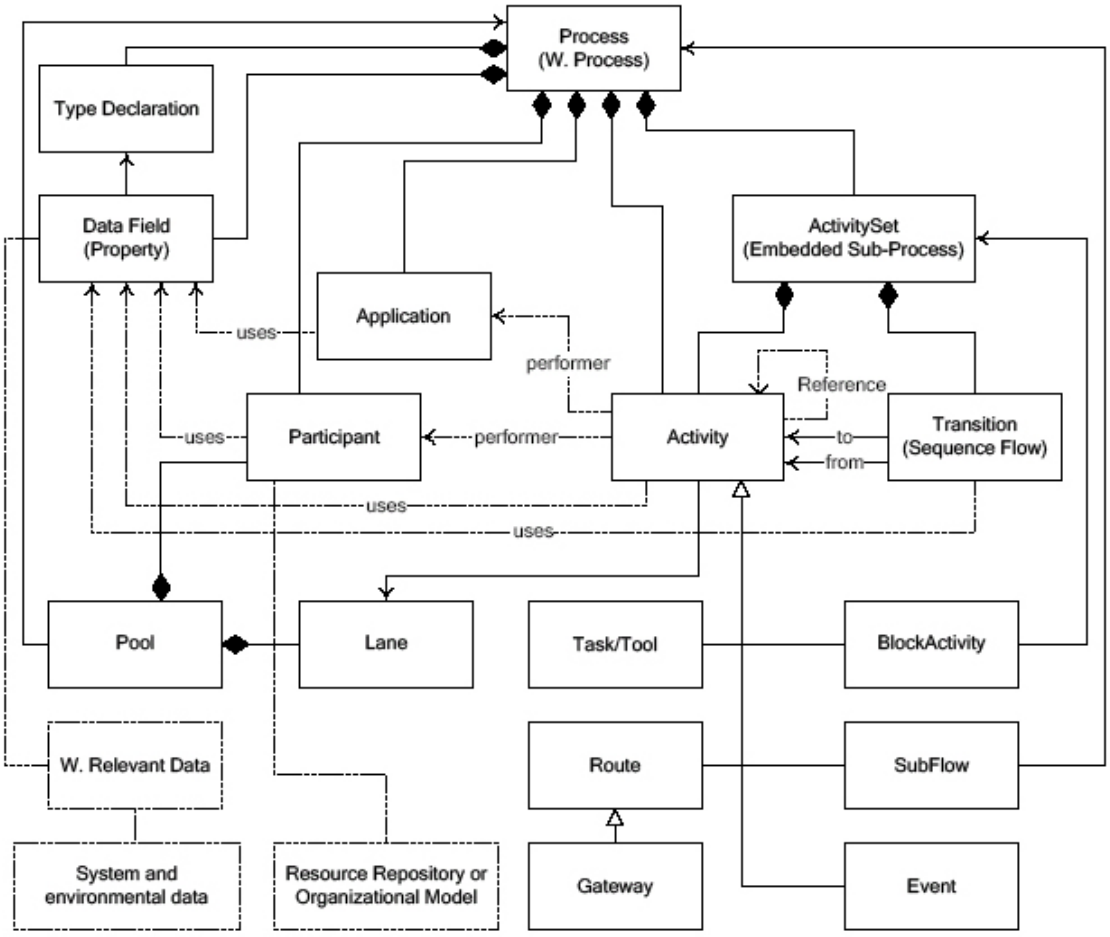

Abbildung 4: Metamodell der XPDL-Prozessdefinition

[6]Vgl. http://www.wfmc.org/xpdl.html

5.3 Bestandteile eines XPDL-Prozesses

Aufgrund des Umfangs von XPDL und der Mischung aus grafischen Annotationen und Prozessablauf wird im Folgenden nur auf einige Elemente der Sprache eingegangen. Listing 6 zeigt die Struktur eines XPDL-Dokuments(XPDL, 2008, S. 169ff).

5.3.1 ConformanceClass

Die Angabe der ConformanceClass gibt an, inwieweit die vorliegende Definition über Element der BPMN verfügen darf bzw. wie mit Blöcken umgegangen wird.(XPDL, 2008, S. 44f) Mögliche Werte für die BPMNModelPortabilityConformance sind NONE, SIMPLE, STANDARD und COMPLETE. Welche Elemente diesen Klassen zufallen, kann unter (XPDL, 2008, S. 46) nachgelesen werden. Die GraphConformance wird in die folgenden drei Mengen unterteilt:

NON-BLOCKED Kein Blöcke

LOOP-BLOCKED Menge azyklischer Graphen

FULL-BLOCKED Blöcke im Sinne von BPEL

5.3.2 Pool

Innerhalb der Sektion <pools> können diverse <lanes> und Annotationen zu grafischen Darstellung eingebettet werden. Lanes können verschachtelt werden und ihnen wird ein Performer zugeordnet, der i.Allg. einer Rolenzuordnung entspricht.(XPDL, 2008, S. 53ff)

5.3.3 Participants

Participants im Sinne von XPDL sind Ressourcenmengen, Ressourcen, Organisationseinheiten, Rollen, Menschen oder das System. Role und Ressourcen sind hier als abstrakte Aktoren zu verstehen.(XPDL, 2008, S. 158ff)

5.3.4 Activities

Activities werden nochmals in Task/Application, Subprozesse, Gateways und Events unterteilt. Eine detaillierte Auflistung sämtlicher möglicher Gateways ist in (XPDL, 2008, S. 81ff) und der Events in (XPDL, 2008, S. 93ff) zu finden. Interessanter sind die Subprozesse bzw. Blockaktivitäten, die den Aufbau eines reduziblen Graphen ermöglichen. Dabei springt die Ausführung in den Startzustand des Subprozesses und anschließend bei Erreichen einer Exit-Aktivität zurück in den aufrufenden Prozess. Aktivitäten können weiterhin eine Implementierung (und damit einzelne Tasks) und ausgehende Transitionen enthalten (Listing 6).(XPDL, 2008, S. 70ff)

5.3.5 Application

Application deklariert eine Liste von Anwendungen und Tools, die innerhalb eines Prozesses aufgerufen werden. Dies dient zur Abstraktion einer konkreten Implementierung oder Plattform und wird dann vom Object Manager konkret instantiiert.(XPDL, 2008, S. 49ff)

5.3.6 Transitions

Der Abschnitt Transitionen dient wieder der grafischen Darstellung. Transitionen können dabei visuell um Conditions erweitert werden.

5.4 Human Task

Menschliche Interaktionen sind bereits ein Kernelement von XPDL und erfordern keine zusätzlichen Erweiterungen. Ein Human Task kann dabei durch einen Task mit dem Element <taskUser> für den allgemeinen Fall(XPDL, 2008, S. 120ff) (bzw. <taskManual> für Aufgaben außerhalb des Workflows(XPDL, 2008, S. 115ff)) eingebunden werden. Dieser Task enthält dann einen Verweis auf den Performer. Weitere beschreibende Attribute können innerhalb der Aktivität angegeben werden.

5.5 Zusammenfassung

Ein großer Nachteil wird wohl in der Praxis die fehlenden Unterstützung für Konzepte wie Transaktionen, Fehler- und Exceptionsbehandlung sein. Ein graphbasierter Ansatz neigt weiterhin dazu schnell unübersichtlich zu werden, bietet dagegen natürlich aber auch eine ungemeine Flexibilität im Vergleich zu den blockstrukturierten Ansatz von BPEL. Die integrierte Unterstützung menschlicher Interaktionen über die Webservices hinaus und die Definition von Performern und Participants sind äußerst positive Merkmale von XPDL. Ein Feature welches bisher nicht erwähnt wurde, ist die Möglichkeit der Simulation von Prozessen durch vordefinierte Standardeingaben. Dies findet in größeren Prozessen sofort seine Berechtigung zum Testen der Arbeitsabläufe.

```
<Package name="sample process" ...>
  <PackageHeader><XPDLVersion>2.0</XPDLVersion>...</PackageHeader>
  <!--Angabe über die BPMN-Unterstützung//-->>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <!--Typdeklaration und Import//-->
  <TypeDeclarations> ... </TypeDeclarations>
  <Participants>
    <Participant Id="DBConnection">
      <ParticipantType Type="SYSTEM"/>
    </Participant>
  </Participants>
  <!--Grafische Darstellung//-->
  <Pools>
    <Pool Process="1" Id="2" BoundaryVisible="false">
      <Lanes/><NodeGraphicsInfos> .. </NodeGraphicsInfos>
    </Pool>...
  </Pools>
  <WorkflowProcesses>
    <WorkflowProcess Id="2" Name="EORDER">
      <ProcessHeader/>
      <FormalParameters> ... </FormalParameters>
      <Applications>
        <Application Id="checkData">
          <FormalParameters>...</FormalParameters>
        </Application>
      </Applications>
      <DataFields/><ActivitySets/>
      <Activities>
        <Activity Id="12" Name="Check Data">
          <Implementation>
            <Task>
              <TaskApplication Id="checkData">
                <ActualParameters> ...</ActualParameters>
              </TaskApplication>
            </Task>
          </Implementation>
          <TransitionRestrictions>
            <TransitionRestriction>
              <Split Type="XOR"><TransitionRefs>
                  <TransitionRef Id="17"/><TransitionRef Id="23"/>
              </TransitionRefs></Split>
            </TransitionRestriction>
          </TransitionRestrictions>
          <NodeGraphicsInfos> ... </NodeGraphicsInfos>
        </Activity>
      </Activities>
      ...
      <Transitions>
        <Transition Id="16" Name="" From="10" To="12">
          <ConnectorGraphicsInfos> ... <ConnectorGraphicsInfos>
        </Transition>
      </Transitions>
    </WorkflowProcess>
  </WorkflowProcesses>
  <ExtendedAttributes>...</ExtendedAttributes>
</Package>
```

Listing 6: Grundgerüst XPDL-Dokument

6 jPDL

6.1 Allgemein

Die Java Process Definition Language verfolgt wiederum einen anderen Weg. Die Knoten und Transitionen werden dabei wieder in einen XML-Format gespeichert. Die ausführenden Anweisungen werden aber an die verschiedenen Arten von Knoten angehängt und z.B. durch eine Verknüpfung zu einer Java-Klasse vorgenommen. jPDL ist Bestandteil des JBoss-Frameworks jBPM und ist somit auf Arbeitsabläufe für die Java-EE-Plattform ausgerichtet. Aufgrund der Java-Nähe wird für jPDL auch eine umfangreiche Ecpliseintegration angeboten. Im Folgenden wird immer die Version 3.2 betrachtet. Eine Version 4 ist derzeit in Arbeit und enthält einige Änderungen, die hier nicht berücksichtigt werden können. Weiterhin wird hier auch bloß die Seite der Prozessdefinition verfolgt, die Ausführung durch Erweiterungen in Java ist hier nicht berücksichtigt.

6.2 Bestandteile eines jPDL-Prozesses

jPDL in der Version 3 unterliegt einem vergleichsweise simplen Schema (Listing 7). Ab der Version 4 scheint es etwas umfangreicher zu werden und auch z.B. Annotationen für die grafische Aufbereitung zu enthalten. (JPDL, 2009, S. 88ff, 142ff)

6.2.1 Swimlanes

Swimlanes in jPDL sind ähnlich den Swimlanes der BPMN und spezifizieren Prozessrollen. Wird einem User eine Swimlane zugeordnet, kann er Nachrichten empfangen, die aus der Zuweisung eines Tasks entstehen (s. Listing 7 Task „do-something“) (JPDL, 2009, S. 113f)

6.2.2 Knotentypen

jPDL bietet diverse Knoten bzw. Zustände an. Neben den Start-, End- und beliebigen Zwischenzuständen existieren Knoten zum Aufteilen von Transitionen (<fork>), Zusammenfügen von Transitionen (<join>) und für Entscheidungen (<decision>). Weiterhin gibt es noch einfache Knoten (<node>) und Task-Knoten (<task-node>). Wie die Knotenarten gegeneinander abzugrenzen sind und wie man „eigene“ Knotenarten durch das class-Attribut erstellen kann, ist in (JPDL, 2009, S. 90f) nachzulesen. Um Teilprozesse zu gruppieren gibt es noch das Element <super-state>.

6.2.3 Transitionen

Jeder Knotentyp kann beliebige Transitionen enthalten. Diese referenzieren einfach den Namen des Zielknotens. Transaktionen können auch Aktionen enthalten.

6.2.4 Actions/Events

Aktionen enthalten ein class-Attribut, welches auf eine Java-Klasse verweist. Diese Klasse wird beim Erreichen der Aktion ausgeführt. Damit bietet sich z.B. die Möglichkeit Transitionen um Aktionen zu erweitern, die nicht direkt in den Businessprozess gehören (z.B. Datenbankupdates). Innerhalb der Knoten können während des Prozessablaufes Events ausgelöst werden, die eine detaillierte Einschränkung für das Ausführen von Aktionen in einem Zustand zulassen.(JPDL, 2009, S. 93ff)

6.2.5 Tasks

Bei der Ausführung der Prozesse werden aus den Tasks Taskinstanzen erzeugt und den Tasklisten einzelner Benutzer oder Gruppen hinzugefügt. Die Verteilung der Tasks kann wieder durch einen eigenen Handler innerhalb des class-Attributes spezifiziert werden.

6.3 Weitere Features

Exception-Handler Die Exception-Handler gibt es nur im Sinne der Java-Exceptions, aber nicht für Fehler, die innerhalb des Graphen auftreten.(JPDL, 2009, S. 97)

E-Mail Innerhalb eines Prozesses kann direkt eine E-Mail mit den Element <mail> oder <mail-node> versandt werden. Auch innerhalb der Assignments führt das notify-Attribute ggf. dazu eine Mail an den entsprechenden User zu senden. Weiterhin können auch die <reminder> E-Mails auslösen.(JPDL, 2009, S. 132f)

Business Calender/Timer Der Businesskalender ist ein sehr interessantes Feature, welches die Definition der Tages- und Wochenarbeitszeit oder von Feiertagen ermöglicht. Anhand dieser Daten kann dann ein <timer> z.B. einen Tag vor einem Feiertag eine bestimmte Aktion auslösen oder Timeouts in Prozessen definiert werden.(JPDL, 2009, S. 120f, 130f)

6.4 Zusammenfassung

Für jPDL kann man die Einschätzung zu XPDL fast vollständig übernehmen. jPDL kann aber durch das schlanke Schema und hohe Flexibilität überzeugen. Durch die enge Kopplung an Java kann jedes Element beliebig erweitert werden und auf eine für den Prozess nötige Aktivität angepasst werden. Der Geschäftsprozess kann zunächst durch einen Fachexperten modelliert werden und im Anschluß durch einen Software-/Prozessingenieur um konkrete Ausführung erweitert werden. Dieser Ansatz könnte auch im Rahmen der MDA zum Zuge kommen. Die Human Tasks sind schlüssig und kompakt in den Prozess integriert und durch das Task-Management auch in Interaktion mit der Außenwelt einfach zu verwenden.

7 Process Engines

BPEL und XPDL können von einer ganzen Reihe von Engines ausgeführt werden. Aufgrund der steten Entwicklung sei hier keine Übersicht, sondern lediglich Verweise zu anderen Listen erwähnt. Eine sehr umfangreiche Auflistung für XPDL ist unter [7] zu finden. Einige BPEL-Engines sind unter [8], [9] oder [10] zu finden.

Bei jPDL sieht es derzeit etwas überschaubarer aus. Der einzige Process Engine ist derzeit der jBPM[11] von JBOSS.

[7] http://www.wfmc.org/xpdl-implementations.html
[8] http://de.wikipedia.org/wiki/BPEL#BPEL_Engines
[9] http://elib.uni-stuttgart.de/opus/volltexte/2006/2908/pdf/FACH_0063.pdf
[10] http://en.wikipedia.org/wiki/Comparison_of_BPEL_engines
[11] http://www.jboss.org/jbossjbpm/

```
<process-definition xmlns="urn:jbpm.org:jpdl-3.2">
  <swimlane name="lane1"/>
  <swimlane name="user"/>
  <super-state ...> </super-state>
  ...
  <start-state name="1" swimlane="lane1">
    ...
    <transition to="2"/>
  </start-state>
  ...
  <!--Teilung von Transitionen//-->
  <fork>...</fork>
  <join>...</join>
  <decision>...</decision>
  <!--Aufgaben//-->
  <task-node name="2">
    <task name="do-something" ...>
      <assignment class="....MyAssignmentHandler" swimlane="user"
                                                        notify='yes'>
        <reminder duedate="2␣business␣days" repeat="2␣business␣hours"/>
      </assignment>
    </task>
    <transition to="3">
      <action class='....MyActionHandler' />
    </transition>
  </task-node>
  ...
  <state name="3">
    <event type='node-enter'>
      <action class='....MyActionHandler' />
    </event>
    <event type='node-leave'>
      <action class='....MyActionHandler' />
    </event>
    <transition to='end'/>
  </state>
  ...
  <end-state name="end"/>
  <!--Exception aus Javaklassen//-->
  <exception-handler/>

</process-definition>
```

Listing 7: Grundgerüst jPDL-Dokument

8 Zusammenfassung

Zusammenfassend lässt sich festhalten, dass alle drei Sprachen eine andere Intention verfolgen: BPEL mit BPEL4People als Prozessausführungssprache im Webserviceumfeld, XPDL als Austauschformat für verschiedene BPMN-Anwendungen und jPDL mit den Schwerpunkt der Integration von Java in Businessprozessen bzw. für Speziallösungen.

Es gilt also, sich vor der Wahl einer dieser Sprachen intensiv Gedanken über die Zielstellung eines Projekts zu machen. Jede Sprache kann in ihrem Bereich überzeugen und alle sind auch noch nicht auf dem Höhepunkt ihrer Entwicklung angekommen. Leider scheint durch eine falsche Zielstellung in den Gremien und Communitys eine Konkurrenz zwischen den Sprachen aufzukeimen, die es in der Realität eigentlich nicht gibt. Die Spezialisierung führt hier zu einer Daseinsberechtigung für alle drei Sprachen.

Abbildungsverzeichnis

1 Konstellationen von Tasks in BPEL4People 8
2 WS-Hierachie 9
3 Übersicht BPMN 10
4 Metamodell der XPDL-Prozessdefinition 11

Listings

1 Grundgerüst BPEL-Dokument 3
2 Grundgerüst BPEL4People 6
3 Beispiel <peopleAssignment> 7
4 Grundgerüst <peopleActivity> 8
5 Beispiel Inline Human Task 9
6 Grundgerüst XPDL-Dokument 14
7 Grundgerüst jPDL-Dokument 17

Literatur

[BPEL4People 2007] BPEL4People: *WS-BPEL Extension for People (BPEL4People)*. Version 1.0. : SAP/IBM (Veranst.), 2007. – https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/30c6f5b5-ef02-2a10-c8b5-cc1147f4d58c

[JPDL 2009] JPDL: *Java Process Definition Language*. Version 3.2.3. http://www.jboss.com: JBoss (Veranst.), 2009. – http://docs.jboss.com/jbpm/v3.2/userguide/pdf/jbpm-jpdl.pdf

[WSBPEL 2007] WSBPEL: *Web Services Business Process Execution Language*. Version 2.0. http://www.oasis-open.org: OASIS (Veranst.), 2007. – http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[WSDL 2007] WSDL: *Web Services Description Language (WSDL)*. Version 2.0. http://www.w3.org: W3C (Veranst.), 2007. – http://www.w3.org/TR/wsdl20/

[WSHT 2007] WSHT: *Web Services Human Task (WS-HumanTask)*. Version 1.0. : , 2007. – https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/a0c9ce4c-ee02-2a10-4b96-cb205464aa02

[XPDL 2008] XPDL: *XML Process Definition Language*. Version 2.1. http://www.wfmc.org: WfMC (Veranst.), 2008. – http://www.wfmc.org/index.php?option=com_docman&task=doc_download&gid=132&Itemid=72