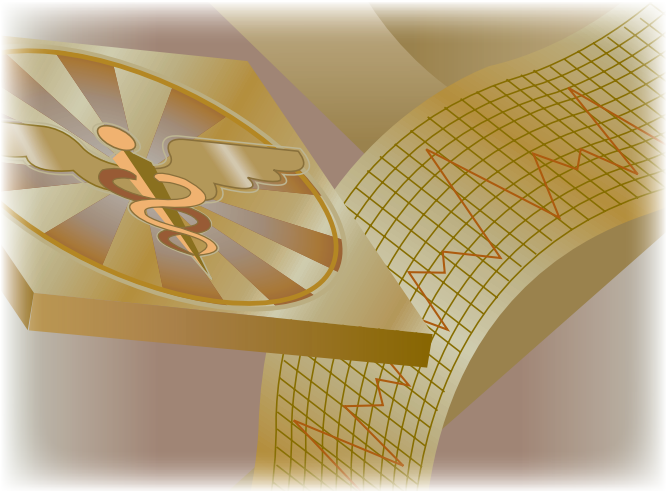
Es muss nicht immer BPEL sein

Welche Sprache spricht BPM?

Bernd Rücker

Business Process Management (BPM) ist zurzeit in aller Munde. Aus Sicht der IT gibt es zwei sehr spannende Aspekte: Das Schaffen einer gemeinsamen Sprache von Fachabteilung und Technikern (Business-IT-Alignment) sowie die Automatisierung der Prozesssteuerung (Process Execution). Für letzteres propagieren alle großen Hersteller heute BPEL, in der Projektpraxis ist das jedoch nicht immer der Königsweg. Der Artikel beleuchtet daher Grundlagen sowie verschiedene Ansätze und Sprachen für die Prozessautomatisierung.



Warum Prozessautomatisierung?

Geschäftsprozessmodellierung ist in Fachbereichen bereits ein alter Hut. Wenn nicht schon seit der gut vermarkteten Idee des Business Process Re-Engineering wurden spätestens im Rahmen des Qualitätsmanagements reihenweise Projekte gestartet, die Prozessmodellierung als Inhalt hatten. Der Hauptfokus lag dabei stets auf der Dokumentation von Geschäftsprozessen.

Fast alle Unternehmen sind aber heute dazu gezwungen, ihre Prozesse auch entsprechend in der IT abzubilden. Die Idealvorstellung ist dabei, dass der dokumentierte Prozess 1:1 in einer Software abgebildet ist, die dann eine entsprechende Prozesssteuerung übernehmen kann. Man spricht hierbei von „Process Execution". Dies erlaubt es, das etwas geschundene Verhältnis von Geschäftsprozessen und Softwaresystemen vom Kopf auf die Füße zu stellen: Nicht mehr der Mensch muss wissen, dass er zu bestimmten Zeiten gewisse Aufgaben zu erfüllen hat, sondern das Softwaresystem sagt es ihm. Um dies zu ermöglichen, wird eine sogenannte Business-Process-Engine benötigt. Diese ist in der Grundidee mit Workflow-Management-Systemen vergleichbar.

Die Prozessmaschine kennt den Prozess und kann dadurch sowohl menschliche Interaktion, das so genannte Human Task Management, als auch Softwaresysteme automatisiert ansteuern, was heute meist über Services im Rahmen einer SOA geschehen soll. Dies ist in Abbildung 1 visualisiert. Es werden also nicht die kompletten Prozesse automatisiert, sondern lediglich deren Steuerung.

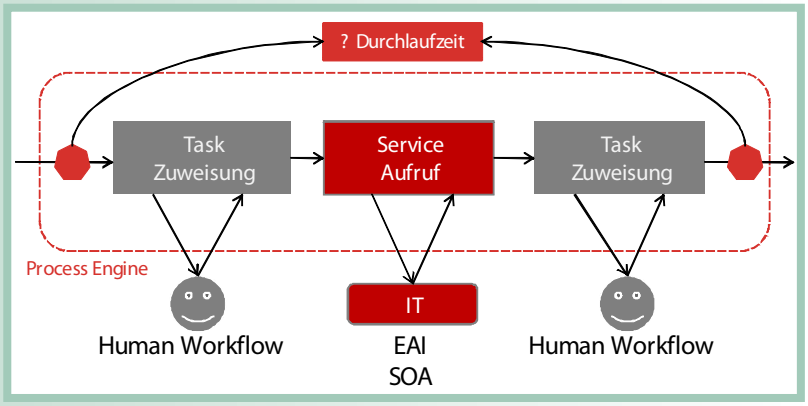


Abb. 1: Prozesssteuerung mittels Prozessmaschine

Prozessautomatisierung auf Knopfdruck?

Leider weckte das Marketing großer Hersteller Erwartungen, die noch nicht erfüllbar sind. Fachabteilungen können ihre Prozesse noch nicht selbstständig modellieren und auf Knopfdruck ausführbar machen. Stand der Dinge sind eher technische Prozessmodelle, die aber zumindest graphisch repräsentiert sind und eine Diskussionsgrundlage für Techniker und Fachbereich bilden. Mitunter lassen sich schon technische und fachliche Modelle so miteinander verknüpfen, dass bei fachlichen Änderungen schnell die technischen Anforderungen ermittelt werden können.

Die Vision geht dabei in die Richtung eines Round-Trip-Generierungsansatzes von fachlichem zu technischem Modell, das Stichwort hier lautet „von der Business Process Modeling Notation (BPMN) zur Business Process Execution Language (BPEL)". Die Technik ist zumindest schon soweit, dass die Prozesssteuerung in vielen Projekten einen echten Mehrwert birgt, den es zu heben gilt. Auch wenn es die Fachabteilungen nicht selbst tun, der Prozessfluss kann oft schon ohne Änderungen am Java-Code (oder ähnlichem) verändert werden. Dies erhöht die so oft geforderte Agilität der Prozesse bedeutend.

Ein ganz großer Gewinn ergibt sich auch durch mögliches Messen von Prozesskennzahlen oder durch den von der Engine bereitgestellten Überblick, wie viele Prozesse in welchem Status stehen. Diese Übersicht in „konventionellen" Anwendungen zu erreichen, erfordert nicht wenig projektspezifischen Aufwand. Die Prozessmaschine bringt die Funktionalität hingegen direkt mit. Nicht zu vergessen ist auch, dass sich sowohl Architekten, Entwickler aber auch Anwender an das prozessorientierte Denken gewöhnen können.

Welche Sprache spricht BPM?

Fast alle Prozessmaschinen auf dem Markt lesen Prozessmodelle als XML-Dateien ein. Diese Prozessdefinitionen können nun aber in unterschiedlichen Sprachen definiert werden. Der Standard, der sich in diesem Bereich eindeutig durchgesetzt hat, heißt „Business Process Execution Language" (BPEL). Da alle großen Hersteller hinter ihm stehen, bezweifelt keiner mehr seine führende Stellung. BPEL kommt jedoch nicht ohne Nachteile ins Haus und es gibt oft Sprachen oder proprietäre Lösungen, die für das jeweilige Projekt deutlich besser geeignet sind. Aber starten wir doch von Anfang an ...

Business Process Execution Language (BPEL)

Die Motivation für BPEL, das bereits seit über fünf Jahren auf dem Markt ist, war nicht etwa Geschäftsprozessautomatisierung, sondern Orchestrierung von Services. Unter Orchestrierung versteht man das Zusammenstellen von Services zu neuen, mächtigeren Services. Geht man über einfache Kommunikationsmuster hinaus, liefern also beispielsweise Services asynchron eine Antwort, so reichen einfache Interaktionsmuster nicht mehr aus. Man spricht hierbei übrigens von „Message Exchange Patterns“ (MEP). Dadurch wird eine zentrale Steuerung zur Orchestrierung notwendig, die beispielsweise auch Wartezustände unterstützt. Hierfür wurde in der Umgebung der Webservices BPEL entwickelt.

BPEL war also „nur“ als Orchestrierungssprache für Webservices entworfen. Diesen Job macht die Sprache auch sehr gut. Hauptelemente sind folglich Konstrukte zum Aufrufen von Webservices (Invoke) oder zum Anbieten eines Webservice (Receive). Zwischen diesen Aufrufen können Prozessvariable als XML gehalten und manipuliert werden (Assign). Daneben stellt BPEL strukturierende Elemente bereit, die jedem Programmierer bekannt vorkommen, beispielsweise Schleifen oder If-Konstruktionen. Fortgeschrittene Konzepte, wie Ausnahmebehandlung oder mächtige Korrelationsmechanismen (um eingehende Service-Calls dem richtigen Prozess zuzuordnen), runden die Sprache ab. Listing 1 zeigt einen einfachen BPEL-Prozess, der in Abbildung 2 graphisch zu sehen ist.

Schaut man sich diesen BPEL-Prozess oder überhaupt BPEL genauer an, so gibt es einige Auffälligkeiten. Zuallererst, dass BPEL eigentlich eine Programmiersprache ist. Zwar ist BPEL-Code in XML geschrieben, aber letztendlich trotzdem Programmcode. Diese Beobachtung wird durch die zweite Auffälligkeit noch gestärkt: BPEL ist blockorientiert. Der Prozess in Abbildung 2 ist kein Graph, die Logik ist stattdessen wie in Programmiersprachen ohne GOTO in Blöcke gepresst. Nun möchte ich nicht behaupten, dass ich das GOTO in Programmiersprachen vermisse, jedoch werden Geschäftsprozesse alles andere als blockorientiert modelliert. Fachabteilungen können dann auch (wohl zu Recht) nichts mit diesen Diagrammen anfangen. Hier herrschen Visio- oder ARIS-Diagramme vor. Und diese arbeiten mit Graphen, die auch Rücksprünge oder ähnliches erlauben. Dies ist übrigens eines der Probleme, warum die Generierung von BPEL-Prozessen aus fachlichen Modellen und vor allem die Unterstützung eines Roundtrips in der Praxis sehr schwierig ist.

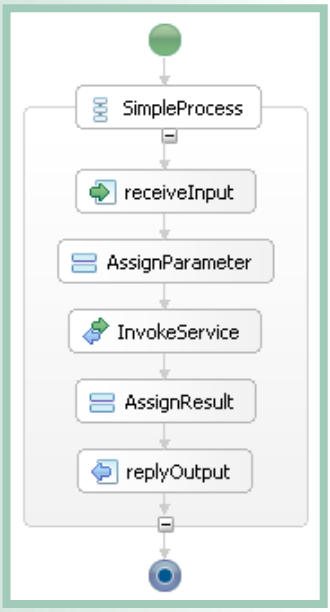


Abb. 2: Beispiel-BPEL-Prozess

Dies alles zeigt, dass BPEL nicht zur Geschäftsprozessmodellierung gedacht war. Da die Sprache aber durch Marketing genau für diesen Zweck auch positioniert wurde, fehlte zumindest noch ein wichtiges Detail: Menschliche Interaktion, also Human Task Management. Dies ist inzwischen unter dem Namen BPEL4People ebenfalls standardisiert, sodass die Sprache zumindest mächtig genug ist, Geschäftsprozesse auszudrücken.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpws:process name="Simple"
              targetNamespace="http://simple.camunda.com"
  xmlns:tns="http://simple.camunda.com" ... >
  <bpws:import location="Simple.wsdl"
               namespace="http://simple.camunda.com"/>
  <bpws:partnerLinks>
    <bpws:partnerLink myRole="SimpleProvider" name="client"
                      partnerLinkType="tns:Simple"/>
  </bpws:partnerLinks>
  <bpws:variables>
    <bpws:variable messageType="tns:SimpleRequestMessage"
                   name="input"/>
    <bpws:variable messageType="tns:SimpleResponseMessage"
                   name="output"/>
  </bpws:variables>
  <bpws:sequence name="SimpleProcess">
    <bpws:receive name="receiveInput"
      createInstance="yes"
      operation="process"
      partnerLink="client"
      portType="tns:Simple"
      variable="input"/>
    <bpws:assign name="AssignParameter"
                 validate="no">...</bpws:assign>
    <bpel:invoke
      name="InvokeService"
      operation="someOperation"
      inputVariable="Request"
      outputVariable="Response"
      partnerLink="SomePartnerLink"/>
    <bpws:assign name="AssignResult" validate="no">...</bpws:assign>
    <bpws:reply name="replyOutput"
      operation="process"
      partnerLink="client"
      portType="tns:Simple"
      variable="output"/>
  </bpws:sequence>
</bpws:process>
```

Listing 1: Beispiel-BPEL-Prozess

Aus technischer Sicht arbeitet BPEL mit WSDL-Schnittstellen für ein- und ausgehende, also angebotene und konsumierte, Services. Prozessdaten liegen als XML vor, Transformationen werden durch XSLT unterstützt. Nicht durch BPEL adressierte Aspekte wie Sicherheit oder Transaktionssteuerung werden durch andere WS-*-Standards adressiert. Somit wird momentan an einer standardisierten Webservice-SOA-Infrastruktur gearbeitet, die aber noch sehr im Fluss ist. Viele Standards, wie beispielsweise WS-Addressing, das sich auch um zustandsbehaftete Services kümmert, sind noch sehr instabil und häufig Änderungen unterworfen.

In BPEL modellierte Prozesse können auf einer beliebigen BPEL-Engine ausgeführt werden. Wie so oft ist das Wechseln der Engine nicht immer trivial, aber grundsätzlich möglich. Viele Engines unterstützen heute bereits die aktuelle Version 2.0 des offiziell WS-BPEL genannten Standards.

In der Praxis ist BPEL zweischneidig zu sehen. Einerseits ist es ein gesetzter Standard mit großer Industrieunterstützung und einer großen Auswahl an Tools. Die Anbindung heterogener Systemlandschaften geht dank Webservice-Technologien einfach vonstatten. Auf der anderen Seite sind BPEL-Prozesse relativ komplex und es wird umfangreiches Know-how benötigt, neben BPEL an sich auch XML, XSLT, X-Path, XML-Schema, WSDL und diverse WS-*-Standards. Dieses Know-how ist in vielen Projekten noch nicht vorhanden und momentan recht schwer einzukaufen. Auch liegen

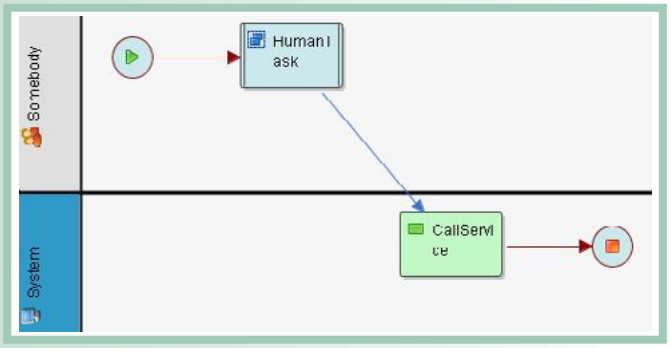


Abb. 3: Beispiel-XPDL-Prozess

noch wenige Erfahrungen mit größeren BPEL-Projekten vor, die echtes BPM als Ziel haben.

Aus diesen Gründen ist es aus meiner Sicht in vielen Projekten sinnvoll, sich zumindest einmal umzuschauen, was es für Alternativen zu BPEL gibt.

XML Process Definition Language (XPDL)

Eine weitere Möglichkeit, Geschäftsprozesse über eine standardisierte Sprache abzubilden, ist die „XML Process Definition Language“ (XPDL). Der Standard ist ähnlich alt wie BPEL und wurde von der „Workflow Management Coalition“ (WfMC) standardisiert, die Organisation, die 1995 auch das bekannte Workflow-Referenzmodell veröffentlichte. Die Idee von XPDL war die Definition einer Lingua Franca für Geschäftsprozesse, also einer Standardsprache für verschiedenste Business-Process- oder Workflow-Engines. Dementsprechend definiert die Sprache nur relativ wenige Elemente, dafür aber viele Erweiterungspunkte. So können beliebige Anforderungen über XPDL-Erweiterungen abgedeckt werden.

Im Gegensatz zu BPEL werden Prozesse in XPDL als Graphen modelliert, was für Geschäftsprozesse viel natürlicher ist. Dies kann man auch in Abbildung 3 sehen, die einen XPDL-Beispielprozess zeigt. Serviceaufrufe können in Prozessen unabhängig von deren Implementierung definiert werden und durch ein durch die Engine bereitgestelltes Anwendungs-Repository aufgelöst werden.

```
<WorkflowProcess Id="Simple" Name="Simple">
  <Participants>
    <Participant Id="Somebody">
      <ParticipantType Type="ROLE"/>
    </Participant>
    <Participant Id="System">
      <ParticipantType Type="SYSTEM"/>
    </Participant>
  </Participants>
  <Activities>
    <Activity Id="HumanTask">...</Activity>
    <Activity Id="CallService">
      <Implementation>...</Implementation>
      <Performer>System</Performer>
    </Activity>
  </Activities>
  <Transitions>
    <Transition Id="HumanTask_CallService" From="HumanTask"
                To="CallService"/>
  </Transitions>
</WorkflowProcess>
```

Listing 2: Beispiel-XPDL-Prozess

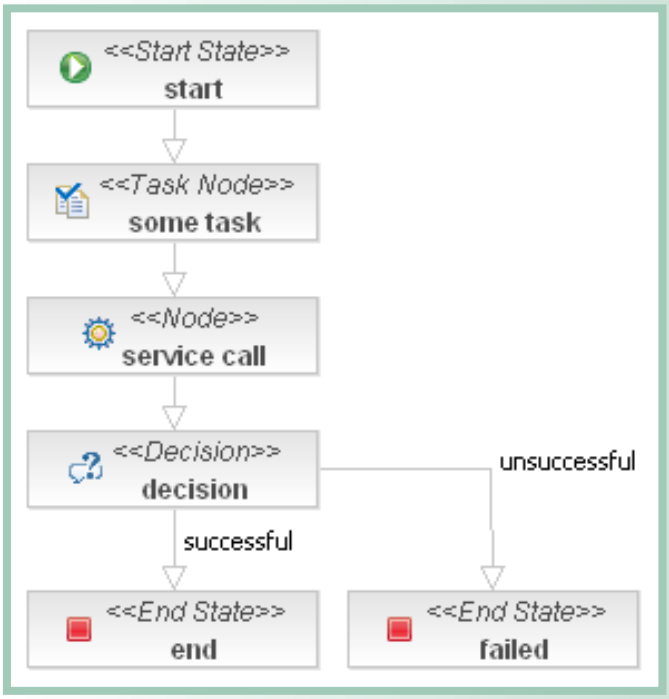


Abb. 4: Beispiel-jBPM-Prozess

Die Offenheit von XPDL führt in der Praxis aber auch dazu, dass verschiedene Engines sehr unterschiedlich in der Mächtigkeit sind. Viel Funktionalität und Logik ist in Erweiterungen versteckt, sodass Prozesse normalerweise nicht ohne weiteres auf eine andere Engine übertragen werden können. Portabel ist meist lediglich die Prozessstruktur. Interessant ist in diesem Zusammenhang, dass XPDL sich gerade als Austauschformat für Diagramme der „Business Process Modeling Notation“ (BPMN) positioniert, sodass diese Diagramme zwischen verschiedenen Tools ausgetauscht werden können, also wie XMI bei der UML. Ob sich dies durchsetzen kann, bleibt abzuwarten. Allerdings ist auch kein anderer Standard in Sicht, der die Austauschbarkeit der Diagramme gewährleisten könnte.

Auf jeden Fall ist XPDL ein recht verbreiteter Standard, den sowohl viele kommerzielle als auch Open-Source-Engines unterstützen. Um genau zu sein, bieten die Tools normalerweise einen XPDL-Import und -Export an. Vor allem Produkte, die Human-Centric-Workflow-Management fokussieren, unter-

```
<process-definition name="Simple">
  <start-state name="start">
    <transition to="some task"/>
  </start-state>
  <task-node name="some task">
    <transition to="service call"/>
  </task-node>
  <node name="service call">
    <transition to="decision"/>
  </node>
  <decision name="decision">
    <transition to="end" name="successful"/>
    <transition to="failed" name="unsuccessful"/>
  </decision>
  <end-state name="end"/>
  <end-state name="failed"/>
</process-definition>
```

Listing 3: Beispiel-jBPM-Prozess

stützen XPDL. Es gibt in diesem Bereich sehr leistungsfähige Implementierungen, die oft deutlich geeigneter als BPEL-Engines sind. Um aber die Leistungsfähigkeit und den Nutzen im eigenen Projekt beurteilen zu können, muss man sich die jeweiligen Engines anschauen und evaluieren, der Standard XPDL hilft dabei leider wenig.

JBoss jBPM: Open Source Process Execution

Man kann auch auf eine proprietäre Geschäftsprozessmaschinen setzen. Es gibt einige im Markt, die meisten davon sind auch durchaus interessant, wobei einige nur für spezielle Anforderungen ausgelegt sind. Nun stoßen proprietäre Produkte ohne BPEL-Unterstützung schnell auf Ablehnung. Ein Kompromiss kann der Einsatz von Open-Source-Software sein, die zwar auch proprietär, aber wenigstens quelloffen verfügbar ist. JBoss jBPM ist eine solches Projekt [jBPM], das sich momentan stark steigender Beliebtheit erfreut und mit JBoss/RedHat im Rücken sogar professionellen Support anbieten kann.

Der Prozess wird in jBPM wie in XPDL als Graph modelliert (s. Abb. 4). Dabei kennt jBPM verschiedene Knotentypen, die entsprechend unterschiedliches Verhalten aufweisen, beispielsweise automatische Entscheidungen, Human Tasks, Forks oder Joins. jBPM ist sehr an Java orientiert, dies bedeutet, dass Prozessvariablen reine Java-Objekte sind, die komplette API zur Steuerung der Engine in Java geschrieben ist und Serviceaufrufe im Prozess über kleine Java-Code-Stücke realisiert werden. Die Engine selbst ist durch einfache Java-Klassen implementiert, um Persistenz kümmert sich Hibernate. Dies macht die Maschine extrem flexibel, da sie in jeder Umgebung, mit oder ohne Application Server und auch mit oder ohne Datenbank zum Einsatz kommen kann.

Befindet man sich in einem Umfeld mit vielen Java-Anwendungen, kann die Engine sehr einfach in die Anwendungslandschaft eingebettet werden. Sind inhomogene Umgebungen zu integrieren, so kann man sich mit relativ einfachen Mitteln eine Infrastruktur schaffen, in der man Webservices orchestriert, ähnlich wie bei BPEL. Alternativ kann auch der JBoss ESB integriert werden, um die Inhomogenität vor der Prozessmaschine zu verstecken.

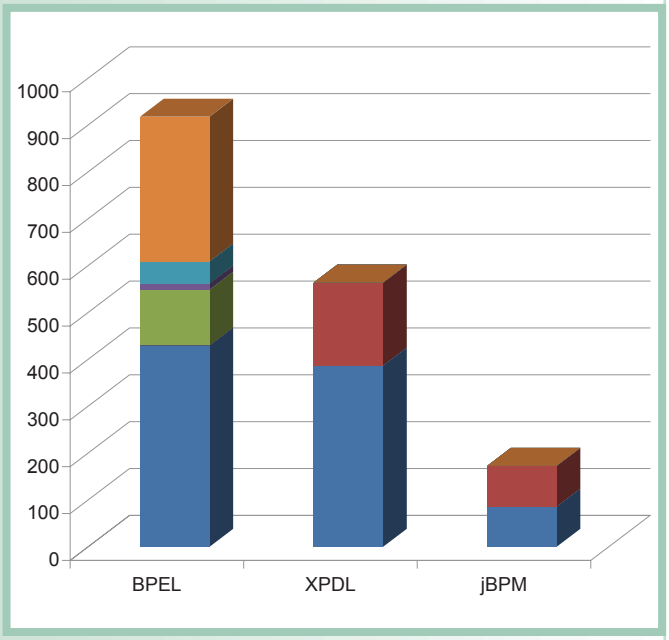


Abb. 5: LOC-Vergleich eines beispielhaften Ticketprozesses

Ein großer technischer Vorteil der Lösung mit jBPM ist, dass die Transaktionsmechanismen der Java-Welt verwendet werden können, beispielsweise der „Java Transaction Service“ (JTS). So kann der Aufruf der Prozessmaschine inklusive einiger Serviceaufrufe in einer atomaren Transaktion ausgeführt werden. Im Gegensatz dazu werden in BPEL die Service-Aufrufe und die Prozesssteuerung jeweils für sich atomar transaktionsgesichert. Dies bedeutet, ein späteres Zurückrollen des Prozesses erfordert es, vorher ausgeführte Aktionen rückgängig zu machen. Ein einfaches Zurückrollen der Transaktion genügt dort nicht.

Die Erfahrung in unseren Trainings zeigt, dass Java-Entwickler sich relativ schnell in die jBPM-Umgebung einarbeiten können, im Gegensatz zu BPEL-Produkten. Neben Java, XML und Grundlagen des BPM sind kaum weitere technische Grundlagen notwendig. Das größte Problem ist dann meist die Umstellung auf die Prozessorientierung. Dies sollte übrigens nicht unterschätzt werden, trifft aber auf alle Prozesssteuerungsprojekte zu.

Somit ist JBoss jBPM durchaus eine ernstzunehmende Alternative für prozessorientierte Anwendungen, vor allem wenn man sich in einem Java-lastigen Umfeld bewegt oder viel Java-Know-how im Projekt hat. Übrigens unterstützt jBPM heute bereits BPEL, auch wenn die BPEL-Unterstützung in meinen Augen noch nicht uneingeschränkt zu empfehlen ist.

Vergleich?

Die Sprachen BPEL, XPDL und jBPM ausführlich zu vergleichen, würde leider den Rahmen dieses Artikels sprengen. Einen Anhaltspunkt kann aber ein einfacher Ticketprozess geben, der im Rahmen einer unserer Schulungen entstanden ist. Abbildung 5 zeigt den Codeumfang (Lines of Code, LOC) für die drei unterschiedlichen Implementierungen. Auch wenn Kennzahlen wie LOC mit absoluter Vorsicht zu genießen sind, zeigen sie eine gewisse Tendenz.

Im Übrigen möchte ich noch erwähnen, dass BPEL für gewisse Problemstellungen eine exzellente Wahl darstellen kann, eben vor allem für die automatisierte Orchestrierung von Webservices. Auch haben die Standardisierungsbemühungen von BPEL sowie BPEL4People sehr viele gute Ideen und Innovationen hervorgebracht. Somit möchte ich keinesfalls BPEL verteufeln, man sollte es aber auch nicht als Königsweg ansehen und blind in jedem Projekt einsetzen.

Ausblick: Process Virtual Machine

Momentan gibt es verschiedene Überlegungen, dass es möglicherweise ungünstig ist, sich auf nur eine Prozesssprache festzulegen. Denn verschiedene Anwendungen in verschiedenen Domänen müssen doch recht unterschiedliche Probleme lösen. Allerdings haben geschäftsprozessorientierte Anwendungen immer sehr ähnliche Grundprobleme zu lösen.

Die „Process Virtual Machine“ (PVM) von JBoss [PVM], zurzeit in der Implementierung im Rahmen von jBPM 4, oder auch die „Microsoft Workflow Foundation“ [MWF] gehen daher einen anderen Weg. Definiert werden nur die notwendigen Kernabstraktionen, um eine Prozesssprache zu entwickeln. Die eigentliche Sprache kann dann aufbauend auf diesen Abstraktionen relativ einfach gehalten werden. Dies ermöglicht es, dass eine Engine BPEL, XPDL und eine

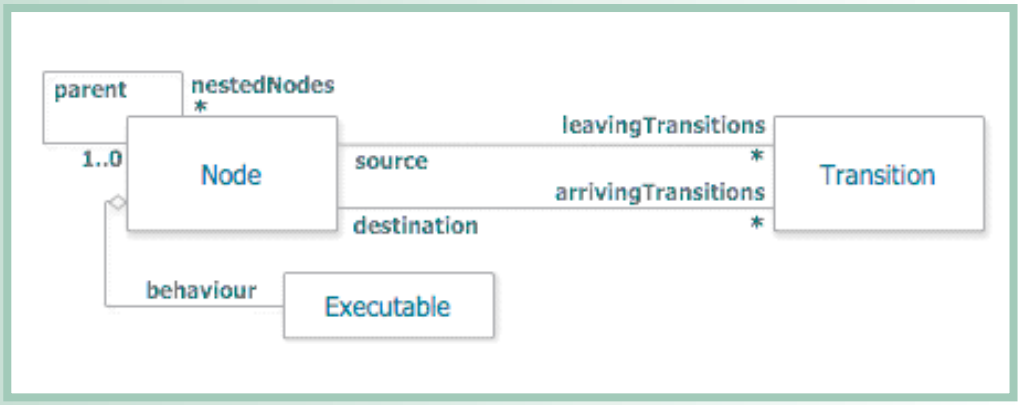


Abb. 6: JBoss PVM

eigene Sprache direkt unterstützt, aber auch die einfache Definition von eigenen Sprachen für spezielle Domänen. Diese Idee ist aus modellgetriebenen Ansätzen als Domain Specific Language (DSL) bekannt.

Im Rahmen der JBoss PVM sind die wichtigsten Elemente Nodes (hierarchisch verbunden) und Transitionen zwischen den Nodes (s. Abb. 6). Damit kann bereits ein beliebiger Geschäftsprozess als Graph modelliert werden. Das Verhalten der Nodes wird dann je nach Prozesssprache unterschiedlich vorgegeben oder gar projektspezifisch implementiert.

Die Verlagerung der Diskussion weg von der besten Sprache hin zur verbesserten Unterstützung unterschiedlicher Sprachen ist sicherlich ein sehr guter Ansatz. Prinzipiell war ja auch die Idee hinter XPDL mit den Erweiterungspunkten nicht weit weg davon. Und dass Microsoft dieselbe Idee verfolgt, untermauert die Tragfähigkeit des Konzeptes. Ein Blick auf die PVM oder die Microsoft Workflow Foundation lohnt sich also allemal.

Ausblick: BPMN

Im Bereich des BPM hat sich inzwischen auch ein Standard zur fachlichen Modellierung von Geschäftsprozessen durchgesetzt, die „Business Process Modeling Notation“ (BPMN). Zurzeit wird dabei an der Problemstellung geforscht, aus fachlich modellierten Diagrammen direkt ausführbare Sprachen wie BPEL, XPDL oder jPDL zu generieren, ein Ansatz ganz im Sinne der Model Driven Architecture (MDA). Auch wenn das Thema momentan noch sehr am Anfang steht, sind bereits prototypische Implementierungen zu finden und es lohnt sich, am Ball zu bleiben.

Wenn diese Vorgehensweise irgendwann technisch reif ist und sich durchsetzt, wovon ich heute ausgehe, dann wird die Wahl der tatsächlichen Ausführungssprache noch unbedeutender. So gibt es heute bereits „Zero-Code“-Ansätze, die BPMN direkt ausführen sollen, wie es beispielsweise SAP in seinem Galaxy-Projekt plant. Derzeit sind diese Ansätze aber leider noch weit weg von der Produktionsreife.

Fazit

Die Prozessautomatisierung mit einer Geschäftsprozessmaschine stellt in vielen Projekten einen großen Mehrwert dar, unabhängig vom verwendeten Standard oder vom konkreten Produkt. XPDL hat eine breite Unterstützung als Prozessaustauschformat, der Großteil an Funktionalität ist allerdings über proprietäre Erweiterungen umgesetzt. BPEL ist tatsächlich zwischen Produkten portabel und hat sich als Automatisierungsstandard momentan durchgesetzt, dafür ist es aber technisch für viele Projekte nicht die beste Wahl. Dementsprechend sollte man sich gut überlegen, wie wichtig der Standard wirklich ist und wie viel mehr beispielsweise die Verwendung von BPEL gegenüber anderen Lösungen kosten darf. Alternativen sind genügend vorhanden und auch die Open Source Community hält spannende Lösungen wie etwa JBoss jBPM bereit.

Insgesamt wird man also im eigenen Projekt um eine Evaluierung der infrage kommenden Standards, Technologien und Produkte nicht herumkommen. Dabei darf man beispielsweise auch die eigene Erfahrung, Know-how im Projekt, vorhandene Systemlandschaft und die wirklichen Anforderungen nicht vergessen. Der Hype um das Thema BPM führt zwar zu vielen Informationen und Publikationen, diese sind aber oft von geringer Qualität. Echte praktische Erfahrungen stellen sich erst nach und nach ein. Erschwerend kommt hinzu, dass das Schlagwort BPM sehr viele unterschiedliche Facetten aufweist. Eine gesunde Mischung aus „learning by doing“, Selbststudium, Beratung und Ausbildung kann helfen. Die Prozessautomatisierung lohnt aber trotz des steinigen Weges zum Ziel auf jeden Fall!

Links

[jBPM] JBoss jBPM, `http://www.jboss.org/jbossjbpm/`
[MWF] Microsoft Windows Workflow Foundation, `http://netfx3.com/content/WFHome.aspx`
[PVM] The Process Virtual Machine, `http://docs.jboss.com/jbpm/pvm/article/`

Bernd Rücker ist Geschäftsführer der camunda services GmbH. Er verfügt über mehrjährige Projekterfahrung als Softwarearchitekt, Coach, Berater, Trainer und Entwickler im Umfeld von BPM und SOA sowie deren praktischer Umsetzung. Er ist Autor eines EJB3-Buches, zahlreicher Fachartikel, Sprecher auf Konferenzen sowie Committer im JBoss-Projekt jBPM. Zum Thema des Artikels hält er eigene Seminare.
E-Mail: bernd.ruecker@camunda.com.