Smart Work am DPMA – Erfolgsfaktoren für die Realisierung

Stefan Kühnlein
IBM Deutschland Enterprise Application Solutions GmbH
München

Schlüsselworte:

BPEL- Testframework, BPEL-Java-API, Error Hospital, Fussion Middleware

Einleitung
Am 1. Juni 2011 wurde im Deutschen Patent- und Markenamt die Elektronische Schutzrechtsakte (ElSA) in Betrieb genommen. Mit der Einführung von ElSA wurde der zentrale Verwaltungsserver durch eine moderne SOA-Anwendung und einem Rich-Client abgelöst.
Die Elektronische Schutzrechtsakte ermöglicht ca. 1200 Mitarbeitern eine effiziente und ergonomische Bearbeitung der ca. 70000 Patent- und 17000 Gebrauchsmusteranmeldungen im Jahr zu verarbeiten. Die Bearbeitung aller Anträge übernehmen 60 zum Teil hochkomplexe, ineinander greifende Geschäftsprozesse, die in mehr als 250 BPEL-Prozesse abgebildet sein.
Dieser Vortrag erläutert die Erfolgfaktoren, mit der die BPEL-Prozesse realisiert wurden. Hierzu gehören u.a.:

- die Verwaltung der BPEL-Instanzen unter Verwendung der Java-API
- die Verwendung von Fachprozess-Triggern zum erneuten Start von Fachprozessen.
- die Verwendung des Error-Hospitals zur Realisierung von Sperren
- die Verwendung von Design-Patterns zur Realisierung der BPEL-Prozesse
- der Test von BPEL-Prozessen mit BPEL-JUnit

Verwaltung der BPEL-Instanzen unter Verwendung der JAVA-API
Im Rahmen der Analyse der fachlichen Anforderungen wurden die ca. 60 Geschäftsprozesse identifiziert und modelliert. Die Modellierung dieser Geschäftsprozesse erfolgte mit Hilfe der ereignisgesteuerte Prozesskette (EPK). Aus der Modellierung heraus ergaben sich für diese BPEL-Instanzen die verschiedensten Status wie „*gestartet*“, „*gestoppt*“, „*abgebrochen*“ und „*beendet*“.
Um diese fachlichen Anforderungen umzusetzen wurde die Servicekomponente „*Verfahrensverwaltung*“ spezifiziert. Mit Hilfe dieser Servicekomponente können BPEL-Prozesse gestartet, gestopped, angehalten und wieder fortgesetzt werden. Diese Servicekomponente ruft asynchron einen RoutingService auf den ESB auf, welcher das Logging und die Transformation der Nachricht vornimmt. Der RoutingService ruft wiederum über eine technische Servicekomponente den BPEL-Prozess auf.

Hierzu stellt der BPEL-Prozessmanager folgende APIs zur Verfügung:

- com.oracle.services.bpel.task: Schnittstelle für die Interaktion mit den Benutzer
- com.oracle.bpel.client: Schnittstelle für den Zugriff auf Server-Funktionalität
- com.oracle.bpel.client.auth: Interfaces für die Authentifizierung
- com.oracle.bpel.client.dispatch: Schnittstelle für die Zugriffe auf die Prozesse
- com.oracle.bpel.client.util: Hilfsklassen für HTML- und SQL-Interaktionen
- com.collaxa.xml: Hilfsklassen für XML und XPath
- com.collaxa.common.util: Schnittstelle für den Zugriff auf die Performance Statistik

Verwendung von Fachprozesstriggern zum erneuten Start von Fachprozessen
In dem komplexen fachlichen Umfeld von ElSA gibt es zwischen Geschäftsprozessen gewisse Abhängigkeiten. So darf z.B. eine Patentprüfung nur durchgeführt werden wenn die zugehörige Patentanmeldung auch abgeschlossen ist. Stellt nun der Patentprüfungsprozess fest, dass die zugehörige Patentanmeldung noch nicht abgeschlossen ist, so müsste dieser ständig den Status der Patentanmeldung prüfen. Dies würde jedoch die Performance des Systems erheblich beeinträchtigen.
Um diese Beeinträchtigung zu Vermeiden wurden die Fachprozesstrigger unter Verwendung des Observer Patterns implementiert. Stellt der Patentprüfungsprozess fest, dass die Patentanmeldung noch nicht abgeschlossen ist, so wird ein entsprechender Eintrag in der Datenbank erzeugt. Dieser Eintrag enthält unter anderem die Startnachricht mit der ein neuer BPEL-Prozess gestartet wird und somit die Patentprüfung fortsetzt. Im Gegenzug prüft der Prozess für die Patentanmeldung am Ende, ob ein Patentprüfungsprozess fortzusetzen ist.
Die Implementierung der Fachprozesstrigger erfolgte im ProzessTriggerManager der folgende Methoden zur Verfügung stellt:

- *Registriere Fachprozess*
  Registriert für einen Fachprozess eine Nachricht, mit der der Fachprozess beim Eintreten des Events gestartet wird.
- *Starte Fachprozess*
  Startet den Prozess mit der registrierten Nachricht
- *Lese Fachprozesstrigger*
  Ermittelt die registrierten Fachprozesstrigger für einen Fachprozess

Verwendung des Error-Hospitals zur Realisierung von Sperren
Üblicherweise laufen Prozesse innerhalb einer Akte unbeeinflusst nebeneinander bzw. sind fachlich miteinander gekoppelt .Jedoch existierten auch eine Reihe von Prozessen, die beispielsweise essentielle, gemeinsame Strukturen ändern und damit ggf. Einfluss auf andere Prozesse und Verfahren nehmen
Für derartige, kritische Prozesse muss sichergestellt sein, dass:

- Kein Vorgang innerhalb eines möglicherweise beeinflussten Prozesses mehr gestartet werden kann, solange der kritische Prozess aktiv ist, d.h. bei einem neu gestarteten Prozess würde die Bearbeitung des ersten Vorganges verhindert, bei einem existierenden Prozess die Bearbeitung des laut Prozessdefinition nächsten Vorganges.
- Der kritische Prozess erst dann mit seinem nächsten Vorgang fortfahren kann, wenn kein beeinflusster Prozess sich noch innerhalb eines Vorganges befindet.

Die Implementierung der Sperren erfolgte in der Sperrenverwaltung, die folgende Methoden zur Verfügung stellt:

- *Bearbeitungssperre liefern*
  Liefert alle Bearbeitungssperren der Akte oder zu einer Fachprozessart.
- *Bearbeitungssperre setzen*
  Erzeugt für den Fachprozess eine Bearbeitungssperre. Existiert zu einem gleichnamigen Fachprozess der Akte eine Sperre, wo wird dieser ebenfalls gesperrt.
- *Bearbeitungssperre löschen*
  Überprüft, ob für einen Fachprozess eine Bearbeitungssperre existiert und löscht ggf. diese.

Die Realisierung der Sperrenkomponente erfolgt mit Hilfe des Error-Hospitals. Stellt der Dienst „*Bearbeitungssperre setzen*“ fest, dass bereits ein gleichnamiger Fachprozess der eine Sperre gesetzt hat, so wird eine Exception geworfen und der zugehörige BPEL-Prozess landet im Error-Hospital. Der Dienst „*Bearbeitungssperre löschen*“ prüft ob für die Akte ein gleichnamiger Fachprozess im Error Hospital liegt. Ist dies der Fall, so wird der BPEL-Prozess aus dem Error-Hospital fortgeführt.

Verwendung von Design-Patterns zur Realisierung der BPEL-Prozesse
Im Rahmen eines Forschungsprojektes wurde 1999 von der Eindhoven University of Technology und der Queensland University of Technology die Workflow Pattern Initiative gegründet. Ziel dieser Initiative war es, die Grundlegenden Anforderungen, die während der Geschäftsprozessmodellierung auf einer wiederkehrenden Grundlage entstehen, zu beschreiben. Im ersten Schritt dieses Forschungsprojektes wurden 20 Entwurfsmuster entworfen, die den Kontrollfluss von Workflow Systemen beschreibt. Die formale Beschreibung dieser 20 Entwurfsmuster erfolgte mit Hilfe von Petri Netzen (CPN). Inzwischen wurden weitere 23 neue Entwurfsmuster identifiziert

Sequence Pattern
Das Sequence Pattern beschreibt die Reihenfolge der Ausführung innerhalb eines Prozesses. Erst wenn die vorherige Aktivität erfolgreich ausgeführt wurde, kann die nächste Aktivität ausgeführt werden. Mit Hilfe des Sequence Patterns lassen sich in der Prozessmodellierung Blöcke mit zusammenhängenden Aktivitäten modellieren, die ohne weitere Bedingungen miteinander verknüpft sind.

Parallel Split Pattern
Mit Hilfe des Parallel Split Patterns wird die Aufteilung in zwei oder mehrere parallele Zweige beschrieben. Die aufgetrennten Zweige können zu einem späteren Zeitpunkt getrennt von einander enden oder mit Hilfe des Synchronization Pattern wieder zu einem gemeinsamen Hauptzweig zusammengeführt werden.
In BPEL wird dieses Pattern durch eine <*flow*> Aktivität abgebildet. Wird in der Realisierung des Parallel Split Pattern keine Parallelisierung benötigt, können die Aktivitäten auch sequentiell ausgeführt werden. Durch die Serialisierung der Aktivitäten kann unter umständen auf Correlations verzichtet werden.

Synchronization Pattern
Mit Hilfe des Synchronization Patterns lassen sich zwei oder mehrere parallele Zweige in einen gemeinsamen Ast zusammen führen. Die Aktivität im Ast wird erst dann ausgeführt, wenn alle Aktivitäten in den parallelen Zweigen ausgeführt sind. Diesem Pattern geht in der Regel das Parallel Split Pattern voraus.
In BPEL wird dieses Pattern durch eine <*flow*> Aktivität abgebildet. Wie auch beim Parallel Split Pattern, kann eine sequentielle Ausführung der Aktivitäten erfolgen.

Exclusive Choice Pattern
Mit Hilfe des Exclusive Choice Patterns kann der Hauptpfad in zwei oder mehrere Pfade aufgesplittet werden. Anhand des Ergebnisses der vorangegangenen Aktivität wird der entsprechende Pfad durchlaufen.
In BPEL wird dieses Pattern durch eine <*switch*> Aktivität abgebildet werden. Mit Hilfe einer <*otherwise*> Aktivität kann sichergestellt werden, dass auch immer ein Zweig durchlaufen wird.

Simple Merge Pattern
Mit Hilfe des Simple Merge Patterns lassen sich zwei oder mehrere unterschiedliche Zweige eines Prozesses zu einem Zweig verschmelzen ohne Synchronisation der einzelnen Zweige. Durch die Verwendung des Simple Merge Patterns in der Modellierung können gemeinsame Aufgaben von unterschiedlichen Zweigen zusammengefasst werden, so dass diese nicht redundant modelliert werden müssen.
In BPEL wird dieses Pattern durch eine <*switch*> Aktivität oder durch die Verwendung einer <*flow*> Aktivität abgebildet.

Advanced Branching and Synchronisation Patterns
Die Gruppe der Advanced Branching und Synchronisation Patterns umfasst eine Reihe von Design Patterns die eine komplexere Aufspaltung und Zusammenführung von Aktivitäten innerhalb von Geschäftsprozessen beschreibt. In der einschlägigen Fachliteratur wird hier eine Reihe von Patterns aufgeführt. Allerdings lassen sich nicht alle Patterns mit BPEL realisieren und werden im Weiteren nicht näher betrachtet.

Multi Choice Pattern
Das Multi Choice Pattern bietet die Möglichkeit für einen Prozess einen oder mehrere parallele Zweige gleichzeitig auszuführen. Die Entscheidung welcher der verschiedenen Zweige ausgeführt werden soll ist vom Ergebnis der vorherigen Aktivität abhängig. Das Multi Choice Pattern entspricht im Wesentlichen dem Exclusive Choice Pattern; allerdings können hier mehrere Zweige parallel ausgeführt werden.

In BPEL wird dieses Pattern ebenfalls durch eine <*flow*> Aktivität abgebildet. Zur Überprüfung, ob der Zweig ausgeführt werden soll, ist in jedem Zweit eine <*switch*> Aktvität notwendig. Da in BPEL dieses Pattern nicht sicherstellt, dass auch wirklich ein Zweig ausgeführt wurde, sollte im Structured Synchronizing Merge Pattern geprüft werden, ob auch wirklich mindestens ein Zweig ausgeführt wurde. Alternativ können die Aktivitäten der einzelnen <*flow*> Aktivitäten auch sequentiell abgearbeitet werden, sofern dies keine Nachteile auf den Geschäftsablauf des Prozesses haben.

Structured Synchronizing Merge Pattern
Das Structured Synchronizing Merge Pattern kommt zum Einsatz wenn das Multi Choice Pattern angewendet wurde. Mit Hilfe des Structured Synchronizing Merge Pattern wird die Ausführung von ein oder mehreren parallelen Zweigen in einem gemeinsamen Ast zusammengefasst. Wurde im Multy Choice Pattern nur ein Zweig angesprochen, so ist keine synchronisation erforderlich.

In BPEL ist dieses Pattern nur mit dem Multi Choice Pattern zusammen anwendbar. Die Synchronisierung der verschiedenen Zweige erfolgt durch die <*flow*> Aktivität automatisch. Es sollte nach der Synchronisierung lediglich geprüft werden, ob auch wirklich eine Aktivität ausgeführt wurde.

Iteration Patterns
Die folgenden Patterns befassen sich mit Wiederholungen von Aktivitäten in einem Geschäftsprozess. In BPEL werden nur das Structured Loops Pattern unterstützt. Das Arbitrary Cycle Pattern und das Recursion Pattern werden von BPEL nicht unterstützt.

Structured Loops
Mit den Structured Loops Pattern können Aktivitäten oder Subprozesse beliebig oft wiederholt werden. Die Prüfung ob eine Aktivität oder ein Subprozess wiederholt werden kann, kann entweder am Anfang oder am Ende des Structured Loops Pattern erfolgen.
In BPEL wird dieses Pattern mit Hilfe der <*while*> Aktivität realisiert. Bei der <*while*> Aktivität wird die Bedingung zur Wiederholung am Anfang geprüft. Prüfungen, die am Ende des Patterns geprüft werden, müssen in BPEL ebenfalls durch die <*while*> Aktivität realisiert werden.

Cancellation Patterns
Verschiedene Patterns wie z.B. das Structured Synchronizing Merge Pattern verwenden verschiedene Konzepte bezüglich des Abbruchs von Aktivitäten bzw. Geschäftsprozessen. Die verschieden Formen des Exceptionhandlings von Prozessen basieren ebenfalls auf Basis der Cancellation Patterns.

Test von BPEL-Prozessen mit dem BPEL Testframework
Mit dem BPEL Testframework des BPEL Prozess Manager lassen sich wiederholbare Testfälle für BPEL-Prozesse erstellen und ausführen. Mit diesen Testfällen lassen sich die Interaktionen zwischen den BPEL-Prozessen und den Web-Services simulieren. Auf diese Art und Weise kann sichergestellt werden, dass die BPEL-Prozesse mit den Web-Services interagieren können und in einer Produktionsumgebung zur Verfügung gestellt werden können.
Das BPEL Testframework stellt ein automatisiertes Test Framework zur Verfügung, mit der wiederholbare Tests von BPEL Prozessen erstellt und ausgeführt werden können.

Das BPEL Testframework stellt folgende Funktionen zur Verfügung:

- Nachbildung der Interaktionen mit den Web-Service Partnern
- Überprüfung von Prozessaktionen
- Berechnung des Prozentsatzes der Source-Code-Ausführung in Bezug auf die Ausführung der einfachen Aktivitäten
- Generierung eines Testfalles aus dem Audit-Trail einer abgeschlossenen Instanz
- Erstellung von Berichten über die Testergebnisse

Im BPEL-Testframework des BPEL Prozess Managers sind folgende Konzepte integriert:´

- Test Case
  Das BPEL Testframework unterstützt sowohl Unit-Test als auch Composite-Test. Bei Unit-Tests werden das Verhalten aufrufenden Web-Services emuliert. Im Gegensatz wird bei dem Compsite-Tests der Web-Service mit den Testdaten ausgeführt.
- Test Suite
  Eine Test Suite besteht aus einer Sammlung von einem oder mehreren Test Cases.
- Emulierung
  Mit der Emulierung kann das Verhalten der Web-Services nachgebildet werden, die bei der Ausführung des Prozesses aufgerufen werden.
- Assertions
  Mit den Assertions können sowohl der Ablauf als auch die Daten überprüft werden.
- Process Code Coverage
  Mit Hilfe der Process Code Coverage ermittelt das Testframework den Grad der Abdeckung bei Ausführung der Test Suite.
- JUnit Unterstützung

Mit dem BPEL-Testframework wird dem Testteam ein geeignetes Werkzeug für die Erstellung und wiederholbare Ausführungen zum Testen von BPEL-Prozessen zu Verfügung gestellt.
Mit Hilfe der Emulierung des Verhaltens von Web-Services kann frühzeitig mit der Erstellung und Ausführung von Testfällen begonnen werden. Für die Tests von BPEL-Prozessen müssen nur die Schnittstellen der verwendeten Web-Services bekannt sein.
h

Stefan Kühnlein
IBM Deutschland Enterprise Application Solutions GmbH
Hollerithstr. 1
D-81929 München

Telefon: +49 (0) 160-88 48 611
E-Mail Stefan.Kuehnlein@de.ibm.com
Internet: http://www.ibm.com/de/de