Smart Work am DPMA – Aspekte der Weiterentwicklung einer SOA-Anwendung

Stefan Kühnlein
IBM Deutschland Enterprise Application Solutions GmbH
München

Schlüsselworte:

SOA, BPEL, Fussion Middleware, Dehydration-Store

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-Architektur 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.
Mit der Einführung einer solchen komplexen Anwendung muss sich Entwicklungsteam mit der Weiterentwicklung und der Abwärtskompatibilität von BPEL-Prozessen und den zugehörigen Artifakten auseinander setzen.
Die Umsetzung einer Änderungsanforderung oder eines Bugfixes und das spätere Einbringen in das produktive System können nach vier Vorgehensweisen erfolgen.

Sicherstellung der Abwärtskompatibilität

Die neue Version einer Funktionalität arbeitet mit Eingabedaten, die mit dem produktiven Stand der Anwendung erzeugt wurden (bzw. von diesem geliefert werden) und hinterlässt Ausgabedaten, die mit dem produktiven Stand der Anwendung weiterverarbeitet werden können.

Parallelbetrieb von bisheriger und neuer Funktionalität

Das neue Release der Anwendung enthält die Funktionalität in der bisherigen und der neuen Version. Eine vorgeschaltete Logik entscheidet, welche Version in einer konkreten Situation angesprochen werden muss. Nachgelagerte Teile der Anwendung können die von beiden Versionen hinterlassenen Daten weiterverarbeiten (Abwärtskompatibilität hinsichtlich Ausgabedaten). Um die langfristige Wartbarkeit des Systems sicherzustellen, muss zu einem späteren Zeitpunkt die alte Version der Funktionalität entfernt werden.

Migration

Die neue Version einer Funktionalität ersetzt die bestehende Funktionalität. Vor der Aktivierung der neuen Version der Funktionalität wird der Datenbestand so angepasst, dass diese darauf arbeiten kann. Nachgelagerte Teile der Anwendung können die von beiden Versionen hinterlassenen Daten weiterverarbeiten (Abwärtskompatibilität hinsichtlich Ausgabedaten).

Eskalation der Änderung

Die neue Version einer Funktionalität ersetzt die bestehende Funktionalität. Alle Aufrufer der Funktionalität werden nach einer der vier Vorgehensweisen angepasst.

Aspekte der Weiterentwicklung von Web-Services
Im Rahmen der Weiterentwicklung einer serviceorientierten Anwendung, bzw. durch die Fehlerhebung an einem WebService, kann sich die Funktionalität eines WebServices ändern. Durch die lose Kopplung der WebServices in einer serviceorientierten Architektur kann der Service-Consumer direkt diese Änderung ohne weitere Wartezeiten direkt nutzen. Eine direkte Verwendung des geänderten WebServices kann jedoch nur erfolgen, wenn sich die Schnittstelle des WebServices nicht ändert bzw. sich die Funktion im Rahmen der Weiterentwicklung für alle Service-Consumer bereitgestellt werden kann. Ist dies nicht der Fall, so müssen mehrere Versionen dieses Services zur Verfügung gestellt werden. Die älteren Versionen können erst gelöscht werden, wenn alle Service-Consumer die neuste Version verwenden.
Aktuelle Standards für WebServices haben jedoch keine explizite Unterstützung für die Verwaltung der verschiedenen Versionen. Dieses Problem muss durch die Bereitstellung entsprechender Pattern erfolgen.
Im Rahmen der Weiterentwicklung von PatGbm wurden für die Webservices folgende Pattern identifiziert:

Hinzufügen einer neuen WebService-Methode
Muss im Rahmen der Weiterentwicklung einem WebService eine neue Methode hinzugefügt werden, so sind bzgl. der Versionierung keinerlei Aspekte zu berücksichtigen. Erst nach der Bereitstellung des neuen Services kann dieser verwendet werden.

Änderung der Implementierung einer WebService-Methode
Bei der Änderung der Implementierung einer WebService-Methode ist zu überprüfen, ob es sich um eine abwärtskompatible Änderung handelt.
Bei einer abwärtskompatiblen Änderung einer WebService-Methode liefert die geänderte Methode bei gleichen Eingabeparametern die gleichen Ausgabedaten. In diesem Fall kann die Änderung direkt vorgenommen werden und steht nach der Bereitstellung allen Service-Consumern zur Verfügung.
Bei einer nicht abwärtskompatiblen Änderung einer WebService-Methode ist die neue Funktionalität in einer neuen WebService-Methode zu implementieren. Nach der Bereitstellung des neuen WebServices müssen die Service-Consumer angepasst werden.

Änderung der Schnittstelle einer WebService-Methode
Um geforderte Änderung an der Funktionalität einer WebSerivce-Methode zu implementieren, muss die Schnittstelle einer WebService-Methode angepasst werden. Da WebServices keine Überladung von Methoden erlaubt, muss die neue Funktionalität in einer neuen WebService-Methode implementiert werden. Nach der Bereitstellung des neuen WebService müssen die Service-Consumer angepasst werden. Sind die neuen Parametern vom Typ String und keine Pflichtfelder, so kann unter Umständen eine Erweiterung der bestehenden WebService-Methode vorgenommen werden.

Änderungen von Container-Parameter bzw. Container-Rückgabe in der Schnittstelle einer Web-Service-Methode
Um die Anzahl der Parameter für die WebService-Methoden zu reduzieren bzw. um „mehrere“ Ergebnisse zurück zu liefern wurden Parameter bzw. Rückgabewerte in ElSA in Container zusammengefasst. Eine Änderung dieser Parameter bzw. Rückgabewerte entspricht einer Anpassung der Schnittstelle einer WebService-Methode. Im Rahmen der Umsetzung müssen neue Container-Objekte und eine neue WebService-Methode implementiert werden. Nach der Bereitstellung des neuen WebServices müssen ebenfalls die Service-Consumer angepasst werden.

Aspekte der Weiterentwicklung von BPEL-Prozessen
Im Rahmen der Weiterentwicklung bzw. der Fehlerbehebung von BPEL-Prozessen führ das Deployment eines BPEL-Prozesses immer zu einer neuen Version. Daher muss in einer Produktiven Umgebung der Parameter ServerMode auf den Wert „*production*“ gesetzt werden. Dieser Parameter verhindert, dass ein BPEL-Prozess immer in einer neuen Version bereitgestellt wird und somit laufende und noch nicht abgeschlossene Instanzen in den Status „*stale*“ versetzt werden.
Im Rahmen der Fehlerbehebung von BPEL-Prozessen muss entschieden werden, ob der bereits gestartet BPEL-Prozess weiter ausgeführt werden kann. Ist dies nicht der Fall, so ist zu prüfen, ob eine Migration im BPEL-Prozessmanager durchgeführt werden kann. Ist eine Migration nicht möglich, so ist ggf. der BPEL-Prozess zu beenden und dieser in der neusten Version zu starten.

Aspekte der Weiterentwicklung von Human Tasks
Für die Erfassung der notwendigen Daten werden den Sachbearbeitern und Prüfern entsprechende Wizards zur Verfügung gestellt. Die Übermittlung der Parameter und Ergebnisse erfolgt über SDOs und den zugehörigen XSD. Analog zu den WebServices müssen auch im Rahmen der Weiterentwicklung die Abwärtskompatibiltät berücksichtigt werden.
Bei einer abwärtskompatiblen Änderung eines Human Tasks wird der Wizard mit denselben Eingabeparametern gestartet und liefert die gleichen Ausgabedaten. In diesem Fall kann die Änderung direkt vorgenommen werden und stehen nach der Verteilung der geänderten Anwendung zur Verfügung.
Bei einer nicht abwärtskompatiblen Änderung eines Human Tasks muss durch das Entwicklungsteam genauer geprüft werden, welche Änderungen notwendig sind. In diesem Fall ist analog zu den WebServices eine redundante Implementierung notwendig.

Verwaltung einer produktiven SOA-Anwendung
Jeder gestartete BPEL-Prozess – unabhängig ob dieser erfolgreich ausgeführt wurde – wird in der Datenbank gespeichert. Zu jedem BPEL-Prozess werden folgende Informationen in der Datenbank gespeichert:

- Metadaten der Payload
- Metadaten des Callbacks
- Payload des Callbacks
- Metadaten des Aufrufs
- Audit Information der Instanz

Durch das ständige Starten von BPEL-Prozesse in einer Produktionsumgebung wächst die Größe der Datenbank exponentiell an, was zu gewissen Performance-Engpässen führt. Aus diesen Gründen sind in der Produktionsumgebung folgende administrative Tätigkeiten notwendig:

- Archivierung aller erfolgreich abgeschlossenen BPEL-Prozesse
- Löschen aller erfolgreich zugestellten XML-Nachrichten
- Löschen veraltert Instanzen
- Recovery fehlgeschlagener Instanzen

Hierzu stellt die Oracle BPEL-Prozessmanager Konsole eine benutzerfreundliche webbasierte Benutzeroberfläche für das Verwaltung und das Debugging von BPEL-Prozessen zur Verfügung.
In einer produktiven Umgebung müssen die Administratoren zusätzlich eine Reihe von Aufgaben durchführen. Hierzu gehören unter anderem die Überwachung der Prozesse, das Recovery der fehlgeschlagenen Prozesse und das Löschen von bereits erfolgreich ausgeführten Prozessen.
Für eine effiziente Durchführung der Aufgaben stell der BPEL-Prozessmanager eine API für den Zugriff auf den Dehydration-Store zur Verfügung, der den Status der Prozesse speichert. Diese API bietet eine umfassende Reihe von Klassen zum Finden, Archivieren Löschen von Instanzen, das

Löschen von Callback-Nachrichten sowie die Abfrage des Status von Domänen, von Prozessen und Instanzen.

Archivierung abgeschlossener Instanzen
Wie bereits erwähnt werden alle Informationen einer Instanz im Dehydration-Store gespeichert. Die Speicherung der Informationen einer abgeschlossenen Instanz erfolgt in den Tabellen CUBE_INSTANCE und CUBE_SCOPE. In der Tabelle CUBE_INSTANCE werden die Header-Informationen, Domäne, Erstellungsdatum, Status, Priorität, Titel usw. gespeichert. In der Tabelle CUBE_SCOPE werden z.B. die Werte der Variablen gepeichert.
In einer Produktionsumgebung müssen hunderte von Prozessinstanzen archiviert werden, bevor die Informationen gelöscht werden. Die automatische Archivierung der Prozessinstanzen kann wie folgt druchgeführt werden:

```
public static int archiveInstances(Locator locator, String processId, int
keepdays) throws ORABPELAccessException {
   try {
      WhereCondition wc = WhereConditionHelper.whereInstancesClosed();
      WhereCondition tmpWhere = new WhereCondition();
      NonSyncStringBuffer buf = new NonSyncStringBuffer();
      if (!"*".equals(processId)) {
         buf.setLength(0);
         tmpWhere.setClause(buf.append(" AND ").append(
            SQLDefs.AL_ci_process_id).append(" = ? ").toString());
         tmpWhere.setString(1, processId);
         wc.append(tmpWhere);
      }
      Calendar cal = Calendar.getInstance();
      cal.add(Calendar.DATE, -keepdays);
      buf.setLength(0);
      tmpWhere.setClause(buf.append(" AND ").append(
              SQLDefs.AL_ci_modify_date).append(" <= ? ").toString());
      tmpWhere.setTimestamp(1, cal.getTime());
      wc.append(tmpWhere);
      IBPELDomainHandle domain = locator.lookupDomain();
      return domain.archiveInstances(wc, true);
   } catch (ServerException se) {
         throw new ORABPELAccessException(se.getMessage());
   }
}
```

Löschen von Callback- und Invoke-Nachrichten
Für jeden Callback (receive, onMessage, ...) auf dem eine Instanz wartet wird im Rahmen des Invokes ein entsprechende Subscription gespeichert. Beim Eintritt einer Nachricht versucht der DeliveryLayer diese zu der ausgehenden Nachricht an Hand der gespeicherten Subscription zu korrelieren. Nach erfolgreicher Zustellung der Nachricht bleiben die Subscriptions in der Datenbank gespeichert und müssen über die Stored Prozedure `collaxa.delete_ci(cikey)` gelöscht werden.
Dies gilt auch für alle XML-Callback und XML-Invoke Nachrichten.

Erneutes Ausführen von fehlerhaften Instanzen
BPEL-Prozesse können aus den verschiedensten Gründen in ihrem Ablauf scheitern. Hierzu gehören unter anderem die Eingabe von falschen Werten, fehlerhafter Prozessdesign oder nicht verfügbare Web-Services. In diesem Fall wird der BPEL-Prozess nicht erfolgreich abgeschlossen und endet mit dem Status „*fault*“. Scheitert der BPEL-Prozess auf Grund eines fehlerhaften Designs, so kann der

Prozess nicht erneut gestartet werden. In diesem Fall muss der BPEL-Prozess korrigiert und neu deployed und gestartet werden.
In allen anderen Fällen kann über die Konsole des BPEL-Prozessmanagers erneut gestartet werden. Im Idealfall muss der Aufruf nur wiederholt werden und der BPEL-Prozess wird erfolgreich zu Ende geführt. Andernfalls muss die Invoke- oder Callback-Nachricht korrigiert werden. Dies kann ebenfalls über die Konsole des BPEL-Prozessmanagers erfolgen.

Kontaktadresse:

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