Jede Änderung im Geschäftsprozess muss umgehend in der unterstützenden Software abgebildet werden können. Professionelle System-Entwicklung basiert auf praxiserprobten Vorgehensmodellen und modernsten Engineering-Methoden, Frameworks und Architecture-Blueprints. Mit der Fusion-Middleware-Produktpalette bietet Oracle Lösungen, um diesen Anforderungen nachkommen zu können.

Forms goes SOA

Stefan Jüssen, Trivadis GmbH

Der Artikel beschreibt einen möglichen Ansatz, wie Oracle-Forms-Applikationen mit Service-orientierten Architekturen und Lösungen integriert werden können. Hintergrund ist, dass die getätigten Investitionen eines Unternehmens in bestehende Applikationen geschützt bleiben [1]. Aber kann Oracle Forms diese Kriterien erfüllen? Lassen sich Oracle-Forms-Applikationen schnell und einfach anpassen?

Grundlagen

Warum ist ein SOA-basierter Lösungsansatz überhaupt sinnvoll? Warum sollen Services lose gekoppelt werden, anstatt dass man sie direkt aufruft? Es gibt zahlreiche Beschreibungen, wie ein Web-Service direkt in verschiedenen Oracle-Forms-Versionen aufzurufen ist. Für die Version Forms 11*g* existiert eine Anleitung bei Oracle [2]. Alle Beschreibungen haben folgenden Ansatz gemeinsam:

- Web-Service erstellen
- Web-Service-Proxy erstellen
- Web-Service-Proxy verteilen (Deployment)
- Anpassungen im Oracle-Forms-Module vornehmen, um den Web-Service ausführen zu können (Klassen in den Pfad aufnehmen und Java-Code importieren)
- Ausführen der importierten Java-Methoden

Der Ansatz ist mit viel Aufwand für die Integration eines einfachen Services verbunden und entspricht nicht dem Prinzip, so wenig Logik wie möglich in der Präsentationsschicht zu enthalten.

Es gibt folgende andere Techniken, um einen Web-Service zu integrieren:

- Einsatz des Datenbank-Adapters der SOA-Suite (Inbound / Outbound)
- Einsatz von „External Events“ über „Advanced Queuing“

Welche dieser Möglichkeiten am besten geeignet ist, kann anhand der Interaktion zwischen dem Benutzer und der Applikation festgelegt werden. Drei mögliche Lösungsansätze – One-Way-Call mit Datenbank-Adapter, Synchroner Aufruf mit UTL_HTTP und Asynchroner Aufruf mit Advanced Queuing – werden hierbei beschrieben. Die Lösungsansätze zeigen die Integration der Web-Services sowie die jeweilige Beeinflussung der bestehenden Applikation auf, die so gering wie möglich sein sollte [3].

One-Way-Call mit Datenbank-Adapter

Von einer „One-Way-Message“ (Fire-and-Forget) spricht man, wenn der Client eine Nachricht an einen Service sendet, ohne dass eine Antwort erwartet wird. Ein Bestellprozess ist ein typisches Beispiel für ein solches Szenario. Eine Bestellung wird in der Regel nicht in wenigen Sekunden abgewickelt. Alle Bestellungen werden durch eine Forms-Applikation erfasst und in einer Datenbank gespeichert. Für die Integration ist ein Datenbank-Adapter möglich, der die eingehenden Bestellungen identifiziert und die weitere Ausführung abwickelt.

Dazu ein Beispiel: Im bestehenden Prozessablauf (siehe Abbildung 1) wird die Auslieferung manuell erfasst. Der Versand wird mit einer E-Mail bestätigt. Der Benutzer hat keine detaillierten Informationen über die Ausführung des Prozesses.

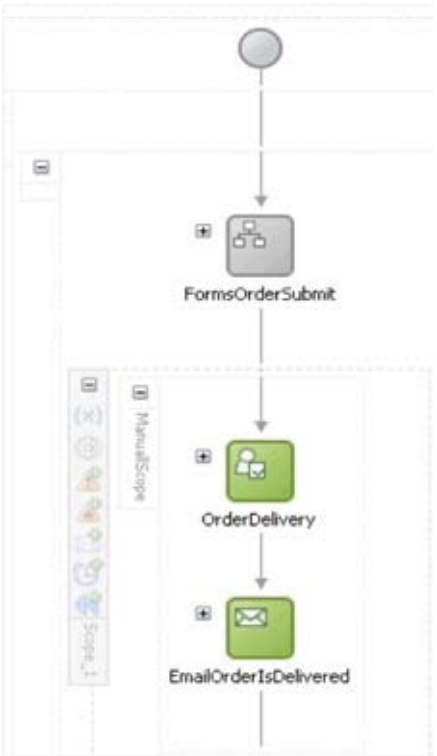

Abbildung 1: Bestehender Prozessablauf

Bei diesem Lösungsansatz hat die Integration eines Web-Services keine Auswirkungen auf die Forms-Applikation, da er von dem Datenbank-Adapter der SOA-Suite kontrolliert wird. Die eingehenden Bestellungen werden durch einen Datenbank-Adapter abgefragt, der weitere Prozesse anstößt. Im angepassten Prozessablauf (siehe Abbildung 2) werden mehrere E-Mails an den Benutzer gesendet, die ihn über den Ablauf informieren. Um die Informationen über den Prozessablauf zu versenden,

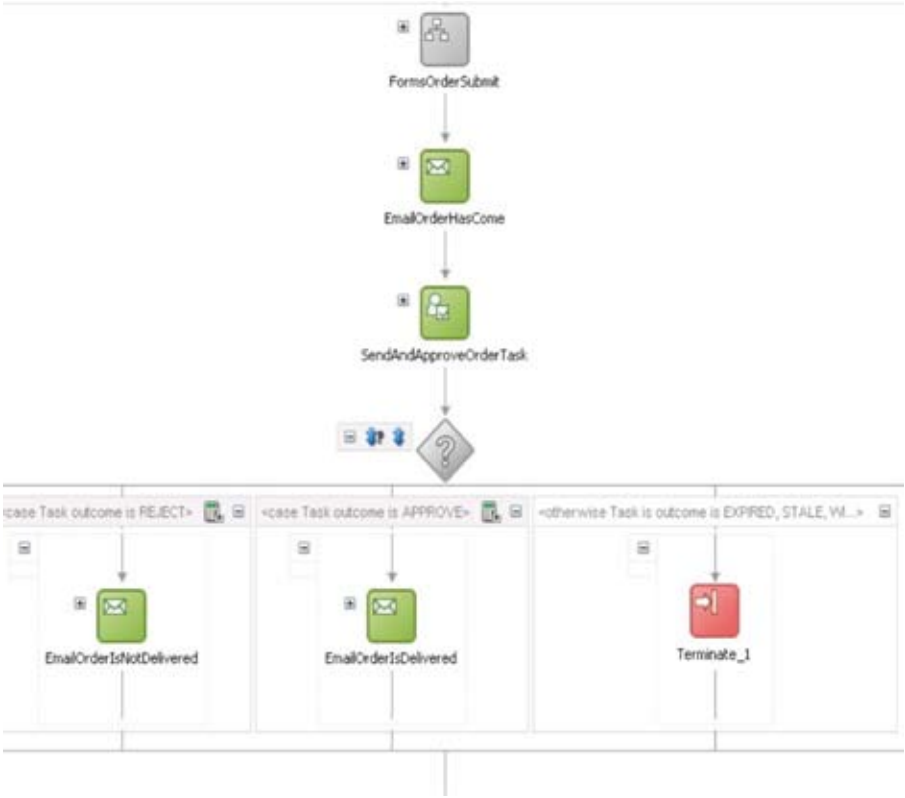

Abbildung 2: Angepasster Prozessablauf

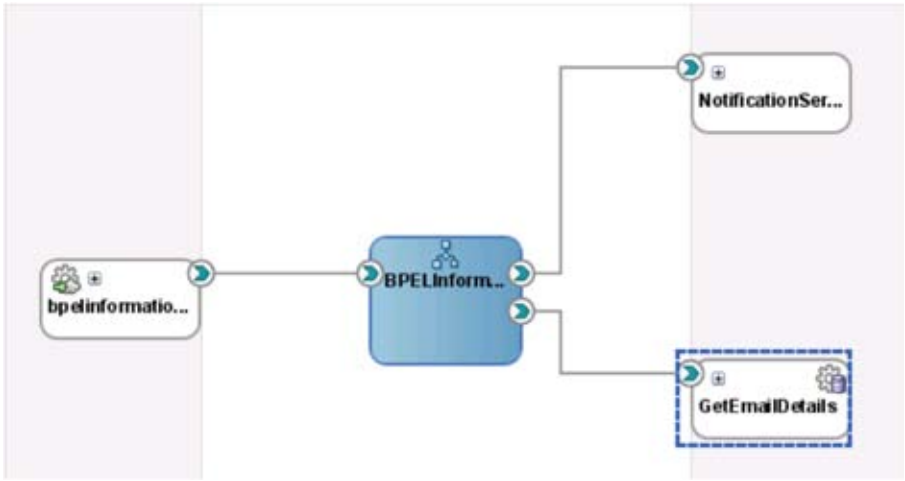

Abbildung 3: Informations-Service

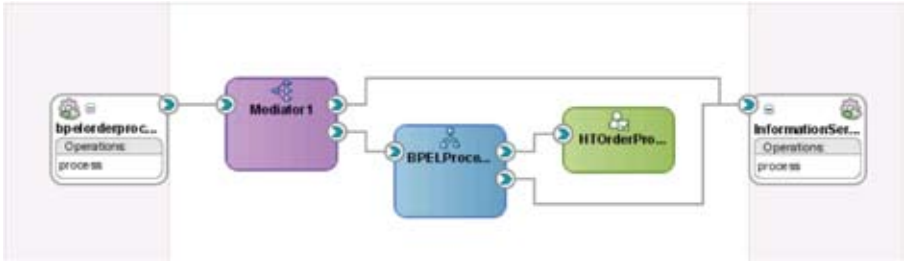

Abbildung 4: Bestellservice

bietet sich ein weiterer Service an. Der zentrale Informations-Service an dieser Stelle hat Vorteile für zukünftige Erweiterungen und verbessert die Wiederverwendbarkeit. Die Versendung der Informationen per SMS könnte eine Alternative oder Ergänzung zum E-Mail-Versand sein. Die SOA-Suite bietet dafür zahlreiche Möglichkeiten.

Der Informations-Service (siehe Abbildung 3) wird durch zwei Parameter gesteuert. Die Bestellnummer und der E-Mail-Typ sind ausreichend, um die übrigen Informationen zu beschaffen. Die Implementierung kann über eine Funktion in der Datenbank „GetEmailDetails“ erfolgen, die die benötigten Daten selektiert und zurückgibt.

Der Prozessablauf der Bestellungen (siehe Abbildung 4) wird durch einen BPEL-Prozess kontrolliert, der einen Bestellservice und einen Human Task ansteuert.

Der Human Task (siehe Abbildung 5) bietet die Möglichkeit, die eingehenden Bestellungen anzunehmen oder abzulehnen. Wenn eine Auftrag nicht ausgeführt werden kann, bekommt der Kunde eine entsprechende Nachricht. In der Abfertigungsabteilung liefern Sensoren den Status der Anfragen, sodass fertige Bestellungen identifiziert werden und Angestellte sich auf die nicht erledigten konzentrieren können.

Der Hauptprozess (siehe Abbildung 6) fragt in regelmäßigen Abständen nach neuen Bestellungen, die in einer Tabelle in der Datenbank abgelegt werden. Er aktiviert den Bestellservice. Falls dieser Prozess im gesamten Unternehmen publiziert wurde, kann er auch von anderen Applikationen genutzt werden.

Bei einer synchronen Ausführung sendet der Client eine Nachricht an einen Service und wartet auf eine Antwort. Buchungen und Reservierungen für Flüge oder Hotels sind typische Szenarien.

Beispiel für einen Buchungsprozess

Der Service für eine Buchung kann von Oracle Forms aus am einfachsten durch eine Prozedur in der Datenbank angesprochen werden. Die

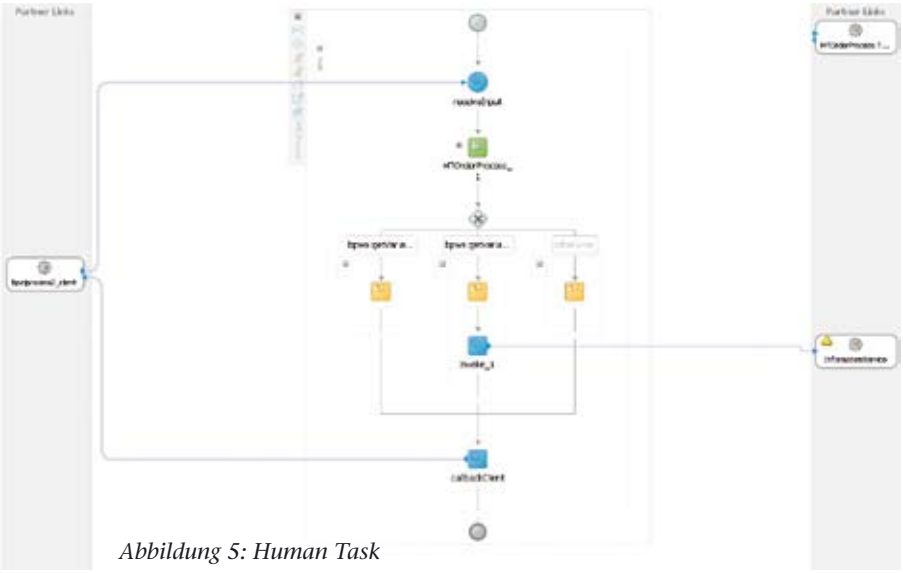

Abbildung 5: Human Task

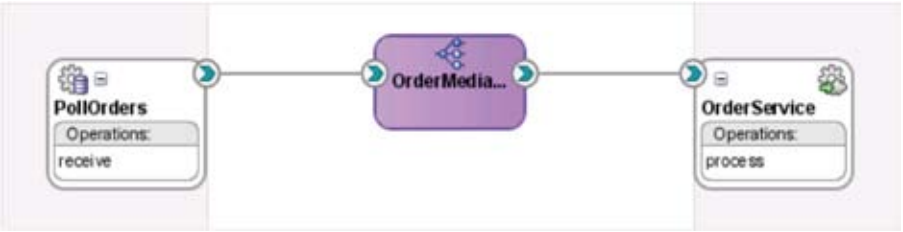

Abbildung 6: Hauptprozess der Bestellungen

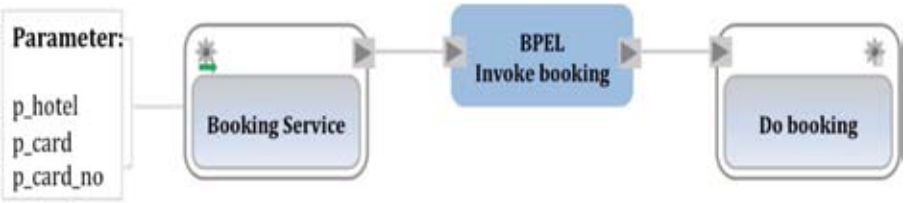

Abbildung 7: Service für Buchungen

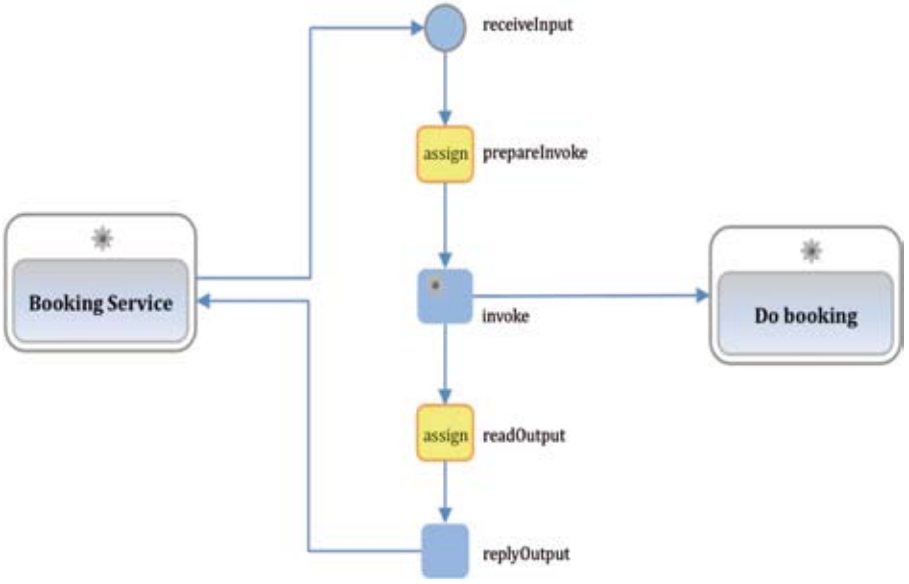

Abbildung 8: Detaillierter BPEL-Prozess

Prozedur implementiert den Service und gibt das Ergebnis zurück. In diesem Beispiel erwartet der Service die Parameter „HotelName“, „CreditCardType“ und „CreditCardNumber“. In Abbildung 7 ruft ein BPEL-Prozess eine Datenbank-Prozedur auf, die den eigentlichen Prozess für die Buchungen ausführt.

Der Service für Buchungen ist in Abbildung 8 detailliert dargestellt. Daraus wird ersichtlich, dass die Aktivität „invoke“ die Buchung synchron aufruft. Das Ergebnis der Buchung wird an den Service zurückgegeben.

Ausführung des Services über eine Datenbank-Funktion

Eine einfache Möglichkeit, den Web-Service in der Datenbank zu konsumieren, bietet das Package UTL_HTTP. Die Verwendung wird in Listing 1 deutlich. Das Ergebnis des Service-Aufrufs wird dabei als XMLTYPE zurückgegeben und kann entsprechend verarbeitet werden.

Schließlich wird die Funktion aus dem oberen Beispiel im Oracle-Forms-Modul aufgerufen.

```
BEGIN
  message (
    booking.book_hotel (,Hotel
California‘, ,Visa‘,
                        ,1234-
1234-1234-1234‘));
END;
/
Result: OK
```

Asynchroner Aufruf mit Advanced Queuing

Bei einer asynchronen Verarbeitung (siehe Abbildung 9) ruft der Client einen Service auf und erwartet eine Antwort. Der Client wird dadurch allerdings nicht blockiert, sondern kann in der Applikation weiterarbeiten. Vertreter dieses Szenarios sind vor allem zeitaufwändige Berechnungen, wie zum Beispiel eine Fourier-Analyse. Der Client initialisiert eine Berechnung, ruft diese auf und möchte über das Ergebnis informiert werden.

Oracle Forms 11*g* kann Benachrichtigungen für External Events abonnieren und darauf reagieren, wenn diese durch die Oracle-Datenbank-Funktionalität „Advanced Queuing (AQ)“ publiziert werden. Forms kann einen asynchronen Prozess initiieren, in dem eine Nachricht in eine Queue gestellt wird. Wenn der Prozess beendet wird, zündet ein „WHEN EVENT RAISED“-Trigger im Forms-Modul, der mit diesem Ereignis verbunden ist. Die Forms-Applikation kann dann die Ergebnisse anzeigen. Andere Technologien wie JMS oder BPEL können ebenfalls Nachrichten an Advanced Queuing übergeben. Dadurch ist Forms in der Lage, auch asynchrone Services zu verarbeiten und auf externe Ereignisse zu reagieren.

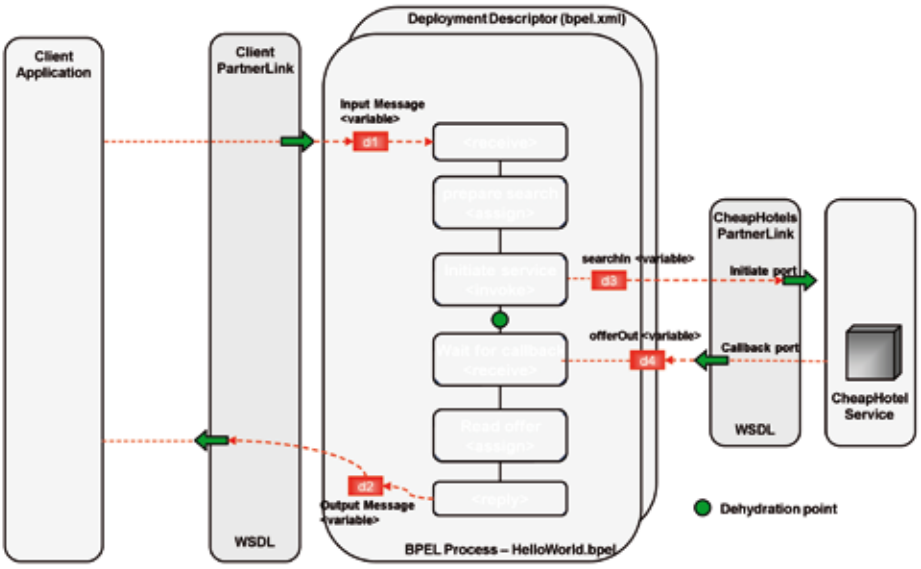

Abbildung 9: Asynchroner Service-Aufruf

Fazit

Besonders in heterogenen Umgebungen bieten Web-Services eine gute Möglichkeit, allgemeine Funktionalität zur Verfügung zu stellen. Forms-Applikationen lassen sich mit solchen Web-Services integrieren.

Referenzen

[1] Daniel Liebhart, Application Development, Trivadis
http://www.trivadis.com/de/solutions/application-development.html

[2] Calling a Web service from Oracle Forms 11*g*
http://www.oracle.com/technetwork/developer-tools/forms/webservices-forms-11g-094111.html

[3] Guido Schmutz, Oracle Service Bus vs. Oracle Enterprise Service Bus vs. BPEL, 08.06.2009
http://www.trivadis.com/uploads/tx_cabagdownloadarea/osb_vs_oesb_bpel_02.pdf

Stefan Jüssen
Trivadis GmbH
stefan.juessen@trivadis.com

```
FUNCTION book_hotel (p_hotel VARCHAR2, p_card VARCHAR2,
                     p_card_no VARCHAR2)
IS
   ...
   resp              XMLTYPE;
BEGIN
   soap_req :=
    ,<soapenv:Envelope .... ,
 || , <soapenv:Header/> ,
 || ,  <soapenv:Body> ,
 || ,   <bpel:process> ,
 || ,     <bpel:hotel>‘ || p_hotel || ,</bpel:hotel>‘
 || ,     <bpel:creditcard>‘ || p_card || ,</bpel:creditcard>‘
 || ,     <bpel:card_no>‘ || p_card_no || ,</bpel:card_no> ,
 || ,   </bpel:process> ,
 || ,  </soapenv:Body> ,
 || , </soapenv:Envelope> ,;
   http_req := UTL_HTTP.begin_request (,http: ... ,, ,POST‘,
                                       ,HTTP/1.1‘);
   UTL_HTTP.set_header (http_req, ,Content-Type‘, ,text/xml‘);
   UTL_HTTP.set_header (http_req, ,Content-Length‘,
                                  LENGTH (soap_req) );
   UTL_HTTP.set_header (http_req, ,SOAPAction‘, ,process‘);
   UTL_HTTP.write_text (http_req, soap_req);
   http_resp := UTL_HTTP.get_response (http_req);
   UTL_HTTP.read_text (http_resp, soap_resp);
   UTL_HTTP.end_response (http_resp);
   resp := XMLType.createXML (soap_resp);
   resp := resp.EXTRACT (,/env:Envelope/env:Body/node()‘,
                         ,xmlns:env=“http ... „‘);
   ...
END;
/
```

Listing 1