Edwin Jongsma
Accenture Technology Solutions
Met dank aan George Stere.

Businessprocessen in een wereld van webservices

BPEL4WS in Microsoft BizTalk 2004

Steeds meer bedrijven raken overtuigd van de universele communicatiemogelijkheden van webservices. De standaardisatie van webservicesprotocollen is een belangrijke stap in de richting van applicatie-integratie. Deze stap wordt nu genomen. De tijd is aangebroken om over de volgende stap te gaan nadenken. Voor systeemintegratie heb je namelijk meer nodig dan webservices. Het aanroepen van een webservice is een activiteit die meestal onderdeel uitmaakt van een businessproces.

Vele bedrijven zijn de laatste jaren actief met de uitrol van ERP, CRM, HRM en andere systemen. De stroomlijning van bedrijfsprocessen binnen deze systemen is nog steeds aan de orde van de dag. De toekomst zit er in businessprocessen te definiëren die de scope van deze systemen overstijgen, omdat hier bedrijfseconomisch voordeel valt te behalen. Dit wordt ook wel Business Process Automation (BPA) genoemd: BPA binnen een bedrijf (EAI), maar ook tussen bedrijven (B2B). Dit is precies het terrein van BizTalk. Met de komst van BizTalk 2000 introduceerde Microsoft Orchestration, een oplossing voor BPA. BizTalk Orchestration is gebaseerd op Microsoft's XLANG-specificaties. Met de komst van BizTalk 2004 is het mogelijk gebruik te maken van een open standaard voor BPA: Business Process Execution Language for web services (BPEL4WS, vanaf nu BPEL genoemd).

Dit artikel bestaat op hoofdlijnen uit de volgende delen:
1. Geschiedenis van de BPEL-standaard
2. Context: wat zijn businessprocessen?
3. Inhoudelijk: het BPEL-model, een beschrijving van de taal
4. Conclusie.

Geschiedenis

De eerste BPEL-versie werd in augustus 2002 gepubliceerd door BEA, IBM en Microsoft. Pas in april 2003 werd de specificatie aangeboden bij OASIS (Organisation for Advancement of Structured Information) om op deze manier een nog bredere acceptatie te krijgen als een open standaard. BPEL wordt gezien als een fusie van WSFL van IBM en XLANG van Microsoft. In het standaardiseringcomité van BPEL zijn de volgende partijen vertegenwoordigd:
- BEA Systems
- IBM Corporation
- Microsoft Corporation
- SAP AP en
- Siebel Systems.

Sun is een belangrijke concurrent, omdat zij een andere standaard positioneren om businessprocessen te modelleren: Web Services Choreography Interface en Business Process Management Language (WSCI / BPML).

Businessprocessen

Webservices zijn op zichzelf staande entiteiten. BPEL is ervoor bedoeld deze isolatie te doorbreken door te specificeren hoe verscheidene webservices een logisch onderdeel van een groter geheel kunnen vormen; een businessproces. BPEL is een extensie van het webservices-interactiemodel. Een businessproces beschrijft de volgorde waarin webservices worden aangeroepen en hoe data tussen deze webservices met elkaar worden gedeeld. Webservices zijn 'stateless', terwijl BPEL 'stateful' langlopende businessprocessen beschrijft.

Met behulp van Microsoft BizTalk Server 2004 kun je businessprocesapplicaties bouwen. Een dergelijke applicatie bestaat uit twee delen:
1. Flow logica van de businessprocesapplicatie geschreven in XLANG/S. Je kunt er ook voor kiezen om je orchestration 'BPEL-compliant' te maken.
2. Functionele logica van de applicatie zelf ontsloten via een webservices-interface.

Het voordeel van deze tweedeling is flexi-

biliteit van de applicatie, doordat de delen 'loosely coupled' zijn. Loose coupling is 'goed koopmansgebruik' voor een beheersbare softwarearchitectuur. Het businessproces kan eenvoudig worden uitgebreid met de aanroep van nieuwe webservices, of bestaande webservices kunnen worden vervangen door nieuwe met een zelfde interface. De interfacedefinities van webservices aanroepen liggen opgeslagen in Web Services Definition Language (WSDL). Wanneer je in 'BPEL-compliant' modus werkt en jouw applicatie als BPEL-file exporteert, dan wordt de informatie van een BPEL-proces opgeslagen in een BPEL-bestand en een WSDL-bestand.

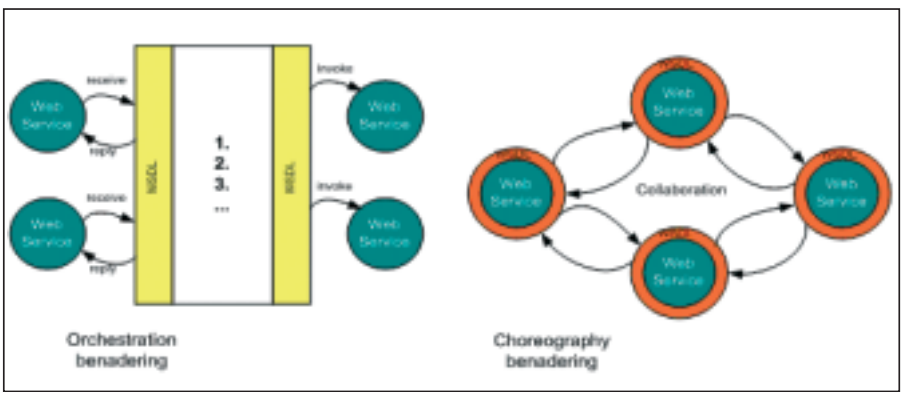
Businessprocessen moeten niet worden verward met transacties. Voor transacties bestaan aparte webservices-standaarden: WS-Transaction en WS-Coordination. BPEL en WS-Transaction zijn complementair. Transacties worden geïmplementeerd door locking-, commit- en rollback-mechanismen. Transacties worden in korte tijd uitgevoerd, 'locks' kunnen dus snel worden losgelaten. De BPEL-scope daarentegen is langlopend, neem bijvoorbeeld het businessproces van een hypotheekaanvraag dat weken kan duren. Locking is derhalve geen optie. Om 'rollbacks' te kunnen doen, zullen in een BPEL-wereld compenserende activiteiten moeten worden ondernomen. Compenserende activiteiten zullen nodig zijn om de consequenties van eerder aangeroepen webservices te corrigeren. Een www.reizen.com-bezoeker sluit bijvoorbeeld een reis af waarbij een hotel, een auto en een vliegreis worden geboekt. In het boekingsproces van een reis, waarbij de webservices van Hilton en Avis worden aangeroepen, gaat er iets fout in de laatste stap bij de aanroep van de KLM-webservices. De auto is echter al besteld bij Avis. Er zal dus een compenserende aanroep naar Avis moeten worden ondernomen om de bestelling ongedaan te maken. Error-handling en compensation-handling vormen dus kritische onderdelen van de BPEL-standaard.

BPEL ondersteunt twee soorten businessprocessen:

1. Executeerbare processen: orchestration
2. Abstracte processen: choreography

Het verschil tussen deze twee soorten processen moet worden genuanceerd, alleen al doordat abstracte processen wel degelijk executeerbaar kunnen zijn. Orchestration impliceert een dirigent die de muzikanten (lees webservices) aanstuurt, terwijl bij choreography de flow van het proces wordt opgepakt door de verschillende dansers (lees webservices) zelf. Bij choreography ligt de nadruk op de berichten die tussen de webservices aan elkaar worden doorgegeven. In het geval van orchestration hoeven webservices zich er niet van 'bewust' te zijn dat zij onderdeel uitmaken van een businessproces. Afbeelding 1 geeft een overzicht.

De bron van orchestration kan gevonden worden in op regels gebaseerde procesautomatisering zoals je die aantreft bij productiebedrijven, bijvoorbeeld een assemblagefabriek voor auto's. De bron van choreography ligt meer in de wereld van op menselijke interactie gebaseerde workflow, zoals die bijvoorbeeld wordt toegepast in Office en Sharepoint. Beide vormen van workflow groeien naar elkaar en vinden elkaar in BPEL. Sun hanteert voor beide vormen een aparte concurrerende standaard: Web Services Choreography Interface en Business (WSCI) & Process Management Language (BPML). In de BPEL 1.1 specificaties zijn aparte extensies gedefinieerd voor orchestration en choreography. Deze specificaties overstijgen de scope van dit artikel. BizTalk Server 2004 hanteert met name het orchestration-model.



Afbeelding 1. Overzicht twee benaderingen: Orchestration en Choreography

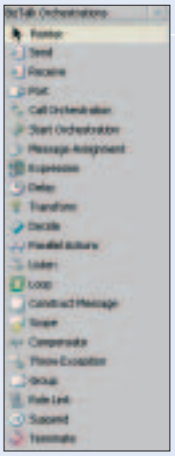
BPEL-model

BizTalk 2004 is volledig geïntegreerd met Visual Studio .NET. Gezien de nauwe link van BPEL met webservices is deze integratie niet alleen logisch, maar ook heel praktisch. De BizTalk Messaging Manager, de Orchestration Designer, maar ook de editor en de mapper zijn volledig geïntegreerd met Visual Studio. In een BizTalk-project kunnen je mappings, schema's en orchestrations aanmaken. Vanuit een orchestration-file kun je BPEL genereren, zie afbeelding 2. Ook is het mogelijk BPEL te importeren.

Afbeelding 2. Het genereren van BPEL vanuit een orchestration-file

BPEL definieert een op XML gebaseerd model om businessproces-interacties te beschrijven. Ook al is het met BizTalk 2004 niet nodig om te weten hoe BPEL-script is opgebouwd, het is leerzaam een kijkje onder de motorkap te nemen. De root tag van BPEL-script is <process>. Hoe kan het ook anders. De belangrijkste secties binnen process zijn:

1. <variables>. Binnen deze sectie wordt de 'state' van het proces vastgehouden in variabelen. In deze sectie worden de parameters van de webservices (waarmee wordt gecommuniceerd) gedefinieerd. Vergelijk dit met de membervariabelen in een class. Stateful in BizTalk betekent niet dat de variabelen gedurende het gehele langlopende proces in het geheugen worden bewaard; dat zou een schaalbaarheidprobleem opleveren. State wordt door BizTalk tijde-



BPEL Primitieve activiteiten	BizTalk orchestrations	BPEL structuuractiviteiten	BizTalk Orchestrations
<empty>		<pick>	Listen
<terminate>	Terminate	<scope>	Scope
<catch>		<while>	Loop
<throw>	Throw exception	<switch>	Decide
<assign>	Message Assignment	<flow>	Parallel actions
<wait>	Delay	<sequence>	
<reply>	Send	<links>	
<receive>	Receive		
<invoke>	Send		
<compensate>	Compensate		

Afbeelding 3. Opsomming BPEL-activiteiten, BizTalk Orchestrations en relaties

lijk in de database opgeslagen totdat het proces weer voortgang vindt - (dehydration en rehydration).

2. <partnerLinks>. In deze sectie worden de partners gedefinieerd die in het businessproces participeren. Een partnerlink verwijst naar een WSDL-file via een zogenaamde portType. PortType is een abstracte WSDL interfacedefinitie. De daaraan gekoppelde 'concrete' interfacegegevens worden opgeslagen in ports. PortType en ports zijn met elkaar verbonden via 'bindings'. In BizTalk kan naast SOAP ook gebruik worden gemaakt van file, HTTP, message queueing en SMTP bindings. Met BizTalk 2004 kun je meer orchestreren dan BPEL toestaat. Zo kun je bijvoorbeeld .NET-componenten aanroepen. BizTalk's XLANG-specificaties zijn derhalve een superset van BPEL.

3. Activiteiten. Er zijn verscheidene activiteiten die het businessproces uitmaken. Dit zijn met name activiteiten waarmee webservices worden aangeroepen, en de flow van het businessproces wordt doorlopen. Welke activiteiten mogelijk zijn wordt hieronder nader uiteengezet.

4. <faultHandler>. Net als in een normale programmeertaal ontbreekt in BPEL ook de error handler niet. Alleen vanuit een error handler kunnen compensationHandlers worden aangeroepen. Een compenserende transactie is een oplossing voor het feit dat 'locking' onmogelijk is gedurende een langlopend proces. Ze kan gezien worden als een rollback voor de langere termijn. Dit impliceert dat webservices die in de rollback worden aangeroepen slim genoeg moeten zijn om compensaties uit te voeren; bijvoorbeeld een reservering of bestelling ongedaan maken.

Activiteiten

Er zijn twee soorten activiteiten: primitieve en structuuractiviteiten. Primitieve activiteiten kunnen worden vergeleken met commando's in een programmeertaal en structuuractiviteiten met taalconstructies zoals een loop. Afbeelding 3 is een opsomming van de BPEL-activiteiten, de mogelijke BizTalk orchestrations en de relaties tussen beide.

De volgende in afbeelding 3 vermelde BizTalk Orchestrations zijn niet conform de BPEL-specificaties: call/start orchestration, transform en suspend. Met behulp van 'role link' kun je aangeven welke rol de <partnerLink> vervult. Met construct message en expressions kun je <variables> bepaalde waardes meegeven.

Met behulp van <receive> geef je het businessproces de opdracht te wachten op een inkomend webservices-bericht. Elk proces begint met ten minste één <receive> activiteit om geïnstantieerd te worden. Met behulp van <reply> kan het businessproces reageren op een inkomend bericht. Met behulp van <receive> en <reply> kun je webservices-interactie afhandelen (request-response). Met <invoke> kun je zelf webservices aanroepen, met of zonder response. <assign> wordt gebruikt om een nieuwe waarde aan een variabele toe te kennen. Voor simpele waardebewerkingen kan gebruikt worden gemaakt van Xpath-expressies, bijvoorbeeld om een tellertje op te hogen. Met behulp van <throw> kun je een faultHandler activeren. Dit gebeurt op vergelijkbare wijze in C# en Visual Basic .NET. <wait> maakt het mogelijk het businessproces voor een bepaalde periode of tot een bepaalde tijd stil te laten staan. Het doel van de <empty>-activiteit is om fouten te onderdrukken, door de orchestration engine te instrueren niets te doen als de faultHandler afgaat. Met behulp van <sequence> kun je activiteiten na elkaar uitvoeren, met <flow> tegelijkertijd. Door middel van <links> kun je volgordelijkheid bepalen binnen een <flow>. <receive> is vergelijkbaar met <pick>, het verschil is dat je met <pick> verscheidene berichten kunt onderscheppen en kunt aangegeven hoe lang de process-engine op een bericht wacht alvorens verder te gaan. Een proces moet altijd beginnen met <pick> of <receive>. De <switch> kan het beste worden vergeleken met een 'select case statement', terwijl <while> voor zich spreekt. De aanwezigheid van deze activiteiten zorgt ervoor dat BPEL zich een taal mag noemen. Weliswaar geen uitgebreide taal met objectgeoriënteerde faciliteiten, maar wel een overzichtelijke taal die zich laat uittekenen. De kracht ervan zit in de eenvoud; dit zou de manier waarop wij heden ten dage programmeren op de kop kunnen zetten: van bedenkers van samenhangende objectstructuren naar losse objecten. BPEL is de weg die deze objecten met elkaar verbindt.

<scope> laat zich het beste vergelijken met een functie binnen een class. Binnen een scope kun je je eigen variabelen en faultHandlers met bijbehorende compensaties creëren. Buiten de scope definieer je globale variabelen. Scopes kun-

```
<?xml version="1.0"?>
<process xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:q1="airline"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   name="TravelAgency.TravelAgencyOrchestration"
   targetNamespace="airlineBP"
   xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">

  <partners>
    <partnerLink name="Customer"
      partnerLinkType="CustomerType"
      myRole="portRole" />
    <partnerLink name="Airline"
      partnerLinkType="q1:AirlineType"
      partnerRole="portRole" />
  </partners>

  <variables>
   <variable name="ItineraryMessage" messageType=
      "TravelAgency_itineraryMessage" />
   <variable name="TicketsMessage" messageType=
      "TravelAgency_ticketsMessage" />
  </containers>

  <sequence>
   <receive    partnerLink="Customer"
       portType="itineraryPT"
       operation="sendItinerary"
       variable="ItineraryMessage"
       createInstance="yes" />
    <invoke    partnerLink="Airline"
      portType="ticketOrderPT"
      operation="requestTickets"
      inputVariable="ItineraryMessage"
      outputVariable="TicketsMessage" />
  </sequence>
</process>
```

Afbeelding 4. BPEL-script

nen binnen elkaar worden 'genest'. Met <compensate> kun je een compensationHandler binnen een 'geneste' scope aanroepen. Het is namelijk mogelijk dat de 'geneste' scope reeds correct is beëindigd, terwijl de hoger liggende scope de faultHandler ingaat. Daarom kun je <compensate> ook alleen binnen faultHandlers aanroepen. De oplettende lezer zal begrijpen dat het voor zichzelf sprekende <terminate>-commando alleen voorkomt in een orchestration, en niet in een choreography scenario. Met deze informatie in handen moet het mogelijk zijn om het BPEL-script in afbeelding 4 te begrijpen. Dit script is trouwens een vereenvoudigd resultaat van afbeelding 2.

Tot slot zijn er nog twee belangrijke BPEL-onderdelen die niet onbesproken mogen blijven: <eventHandlers> en <correlations>. Het doel van een eventhandler is messages te kunnen ontvangen, terwijl het proces wordt uitgevoerd - gelijktijdig. Een eventhandler kan ook worden gebruikt als een timer die na een bepaalde tijd af gaat. In de praktijk wil je na de <invoke> van een externe applicatie een <reply> ontvangen, al is het alleen al een acknowledge dat je gestuurde informatie is verwerkt. Een <invoke> waar een <reply> op wordt verwacht, maakt gebruik van correlatie om binnenkomende berichten te koppelen aan het verstuurde bericht. Met behulp van correlation kun je een 'request-response' implementeren. Een voorbeeld van een <correlationSet> is klant_id en ordernummer. Deze unieke gegevens kunnen worden gebruikt in het proces van een boekbestelling waar een tijd gewacht moet worden op een terugkomend bericht van een partner die de bestelling in verwerking neemt. <correlationSets> zijn gekoppeld aan parameters in de berichten die worden uitgewisseld. Deze parameters maken deel uit van de bij BPEL behorende WSDL-files. Er hoeft veelal geen gebruik te worden gemaakt van correlation, omdat een aanwezige berichtentransportinfrastructuur berichten automatisch laat correleren. <correlation> zorgt ervoor dat ook berichten kunnen worden gekoppeld die verstuurd zijn over niet gecorreleerde asynchrone transportmechanismen. Data in berichten kunnen bestaan uit twee delen: applicatiedata en metadata, in het geval van SOAP <body> en <header>. Correlatiegegevens worden meestal meegestuurd als metadata van een bericht, maar je kunt ook op zinvolle inhoudelijke data correleren, zoals op ordernummer in een orderbericht.

Conclusie

Bepaalde bedrijfsprocessen komen bij meer bedrijven voor. Het voordeel van BPEL is het hergebruik van standaard businessprocesschema's. Een BPEL-specificatie is in principe platformonafhankelijk. Zo kan BPEL - ontwikkeld met Microsoft BizTalk - worden uitgevoerd door een andere BPA engine en andersom. Standaardisatie zorgt ervoor dat platformgrenzen vervagen en integratie steeds eenvoudiger wordt. In de praktijk zal het zo'n vaart niet lopen. Om succesvol te zijn is BPEL afhankelijk van webservices. Webservices hebben zeker momentum, maar de organisaties die hun business al ontsluiten op deze wijze zijn nog in de minderheid. BPEL is dus voor vele bedrijven nog toekomstmuziek. Het kan echter op korte termijn in specifieke gevallen al heel pragmatisch worden toegepast.

Nuttige internetadressen

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbiz2k2/html/bpel1-1.asp
http://dev2dev.bea.com/technologies/webservices/BPEL4WS.jsp
http://www-106.ibm.com/developerworks/webservices/library/ws-bpel
http://ifr.sap.com/bpel4ws/
http://www.siebel.com/bpel