Goede strategie is het halve werk

Migreren van langlopende BPEL-processen

Om niet voor verrassingen komen te staan, is een goede strategie voor aanvang van een BPEL-project onontbeerlijk. Zeker wanneer het in een project gaat om langlopende BPEL's. We nemen een fictief project en kijken naar de problemen die daarbij ontstaan en de uiteindelijk gekozen oplossing om het project succesvol voort te zetten.

Het project was vol goede moed gestart en de eerste processen waren – weliswaar wat onwennig – gemodelleerd in BPEL. Onwennig, omdat het projectteam niet gewend was om zo in processen te denken. Toch kwamen de voordelen duidelijk aan het licht: de gebruikers konden goed meepraten tijdens het ontwerp en ze herkenden hun eigen processen. Onwennig ook, omdat dit programmeren van processen uiteindelijk een heel technische aangelegenheid is en een beetje lijkt op het aanleren van een nieuwe taal, maar dan op een iets andere manier. Enige twijfel was er bij de eerste stappen, maar het projectteam raakte steeds meer vertrouwd met het werk en er werden mooie processen gesmeed. De eerste van negen releases was zo goed als klaar.
De testers morden steeds als hun lopende processen weer waren verdwenen, nadat nieuwe versies waren opgeleverd. Ze konden toch niet verwachten dat hun processen konden blijven draaien? Na een paar maanden kon de eerste release opgeleverd worden en dat bracht uiteraard blije gezichten en knallende kurken met zich mee.

Daardoor geïnspireerd pakte het projectteam de draad weer op en werd naar de tweede release toegewerkt. Grote problemen bleven uit en alle processen deden hun werk naar behoren. Vlak voor de oplevering van de tweede release aan de testers was er een vergadering over de live-gang ervan. Werden er toch nog problemen verwacht? De BPEL-mannen zagen die gelukkig niet, maar er moesten nog wel wat dataconversies komen.
De gebruiker van de financiële afdeling had nog een aantal vragen. In de eerste release waren zijn processen nog maar gedeeltelijk opgeleverd en het overige deel stond klaar in de tweede release. Hij vroeg zich af wat er ging gebeuren met de processen die al waren gestart en nu dus van een nieuwe versie konden worden voorzien. De BPEL-mannen zagen weinig problemen.

Probleem

Toch bleek hier wel een groot probleem te zitten. De BPEL-versie die gebruikt was, kende geen mechanisme om eenvoudig lopende versies van een proces te vervangen door nieuwe. Dit probleem werd binnen het project bekend als 'inflight migration'.
De mogelijkheid om een bestaande (flying) instantie van een proces, dat dus gestart is en nu op een bepaalde processtap staat te wachten, over te zetten naar de nieuwe versie van dat proces was gewenst en wel zo dat de instantie weer op dezelfde plek (of een gewenst alternatief) stond te wachten. De BPEL-mannen kwamen er na vele discussies en brainstormsessies achter dat er een aantal oplossingen waren voor dit probleem. Ze hadden de projectleider wel vast verteld dat er nog een forse change aan zat te komen.

Oplossing 1: Laten uitlopen

Dit is eigenlijk geen oplossing voor het probleem. Het betekent simpelweg dat je de lopende processen niet van een nieuwe procesdefinitie voorziet en ze laat aflopen met hun huidige definitie. Op termijn zou dit wel een goede oplossing kunnen zijn, als het project wat stabieler zou zijn, maar op dit moment waren de nieuwe BPEL-processen nog aan te grote wijzigingen onderhevig. Het project was immers pas bij release twee van de negen.

De BPEL-mensen konden zich wel voorstellen dat bepaalde wijzigingen die op termijn werden doorgevoerd alleen maar hoefden te worden doorgevoerd voor nieuw startende processen. Bijvoorbeeld als wetgeving vanaf een bepaald moment wijzigt. Het betekent wel dat de oplossing die uiteindelijk gekozen zou worden de mogelijkheid moest hebben om lopende processen ongemoeid te laten. Deze oplossing bleek dus niet de juiste.

Oplossing 2: Een universeel proces met sub-aanroepen

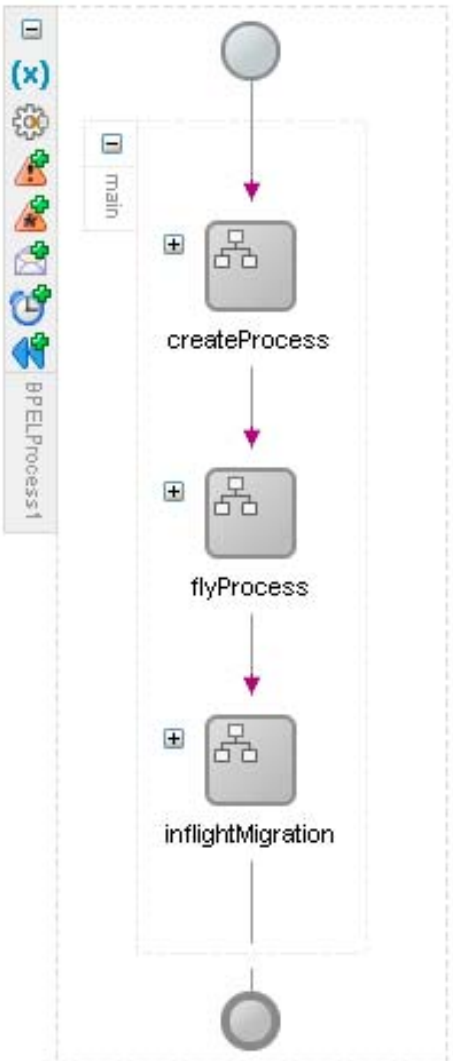
Deze oplossing kwam eigenlijk als vanzelf voort uit de eerste. Was het niet logisch om één generiek proces uit te denken, dat gedurende de komende acht releases niet meer zou wijzigen en die subprocessen aanroept die wel konden wijzigen, maar die slechts kort waren en dus niet meer hoefden te worden gemigreerd.
Het was een goede kandidaat om als 'dé oplossing' uit de bus te komen. In wat workshops werd met key-gebruikers bepaald hoe de rest van de processen er uit moesten zien, in grote stappen. Het ging goed, maar de gebruikers waarschuwden toch wel vaak dat zij niet honderd procent overtuigd waren, dat in dit grote proces geen wijzigingen meer zouden optreden. Het risico was te groot. Ook deze oplossing bleek niet de juiste.

Oplossing 3: Afsterven en Herstarten

Eén van de BPEL-ers had uitgevonden dat je geautomatiseerd je lopende processen kon opsporen en dat je kon uitlezen in welke processtap deze zich bevonden. Dus kon een mechaniek worden bedacht waarbij je alle lopende BPEL-processen opspoort en vervolgens per BPEL-proces de oude versie stopt en een nieuwe versie start.
Dit werkte! Dit was de oplossing! Snel en voortvarend werd het patroon bedacht van deze nieuwe BPEL, zodat het proces kon worden herstart. Al snel boekte het team succes met dit mechanisme in Test, want de testers kregen een nieuwe versie van het BPEL-proces en hun lopende versie van dit proces was meteen gemigreerd naar deze nieuwe versie.
De oplossing was als volgt gebouwd: per nieuw proces werd een migratie-BPEL gestart. Deze spoort alle lopende BPEL-processen van een bepaald type op. Per lopend proces werd het proces uitgelezen, vervolgens beëindigd, een korte tijd gewacht waarna een nieuwe versie werd opgestart en vervolgens op de juiste plek neergezet. Het werkte ideaal, was best een grote ingreep, maar na een paar pizza-avonden was de live-gang van release twee gered.

In het weekend voor de live-gang voerde het team de BPEL-migratie uit. Er was niet met een kloon van productie getest, dus spannend was het wel. Het viel tegen en eigenlijk ook weer mee. Eerst leek het erop dat de processen waren vastgelopen. Al snel ontdekte het team dat de processen niet vastgelopen waren, maar dat bij enkele honderden processen het wachten op het afsterven van een proces veel tijd in beslag nam. Die tijd zorgde ervoor dat het afsterven en opstarten een fors aantal uren duurde. Maar het werkte wel.
Heerlijk een goede live-gang. In de week na het spannende weekend werd er nog wel veel nagepraat. Hoe kwam het eigenlijk dat die migratie van BPEL zo lang duurde? Al snel bleek dat het team had ingebouwd dat het wachten op het afsterven van een BPEL-proces was geprogrammeerd op vier seconden. Al rekenend was de conclusie dat als je later een paar miljoen van deze processen moest migreren, je snel op een aantal weken migratietijd uitkomt, ook al zou je die vier seconden terugbrengen naar één of minder.
Dit alternatief werkte, maar alleen bij lage aantallen of bij veel tijd bij de overstap naar een nieuwe versie. De tijd voor de migratie was echter maar een paar dagen, dus de huidige oplossing was vele dagen te lang. Terug naar de tekentafel.

Er werd al snel bedacht dat de huidige oplossing best zou kunnen werken als niet in een weekend moest worden gemigreerd, maar een proces pas gemigreerd hoefde te worden als het weer opgestart wordt. Een korte check kijkt of er dan een nieuwe versie is en start zichzelf zo op. Geen migratietijd, alleen een korte tijd extra als een proces werd opgestart.
Wat verder nadenken over deze oplossing gaf wel een aantal problemen. Hoe weet je nu zeker dat uiteindelijk alle processen



Een voorbeeld van een willekeurig BPEL-proces.

zijn gemigreerd? De interne auditor wilde wel een voor- en achterafrapportage van het aantal processen, wat echter in deze oplossing niet mogelijk is. Daarnaast konden sommige processen zo lang stil liggen dat ze een paar releases overslaan. Dat leidt ertoe dat je veel verschillende versies van een proces kunt hebben 'draaien' wat als ongewenst werd beschouwd.
Er moest een andere oplossing worden bedacht.
Er werd een workshop georganiseerd met interne en externe specialisten (ook van Oracle) waar het probleem werd besproken. Daar kwamen twee alternatieven uit:

Oplossing 4: BPEL proces op basis van masterdata

Op Oracle Technet circuleerde deze oplossing al een tijdje. Je brengt je BPEL's terug tot een BPEL die als processtap steeds uit Metadata (xml's) ophaalt wat zijn volgende stap moet zijn. Doordat je deze metadata kunt wijzigen, kun je elke gewenste wijziging doorvoeren op elk gewenst moment.
Deze oplossing werd snel door de aanwezige gebruikers aangevochten. Waar konden zij dan hun proces terug zien? Dat was juist het voordeel van BPEL. Moesten ze het uit een XML halen of waren er dan tools die, zoals BPEL PM, het proces van de metadata kon laten zien?
Deze oplossing leek goed voor processen die extreem flexibel moesten zijn maar was geen oplossing voor deze langlopende processen.

Oplossing 5: Kortlopende BPEL processen met behulp van States

"Waar staan die processen eigenlijk op te wachten?" Het was tijdens de lunch dat de facilitator van de workshop die vraag stelde. "Op het antwoord van een andere service of proces," was het antwoord. "Staan al die processen dan in het geheugen van de server?" vroeg een ander. "Nee deze worden in de dehydration store opgeslagen door de BPEL proces manager," zei één van de BPEL-programmeurs, en aanvullend "Netjes in de database, dat heeft Oracle goed geregeld."
"Maar kunnen we dan in die database niet wat zaken aanpassen, zodat er met de nieuwe versie van het proces wordt verdergegaan, gewoon door een aantal updates?" Het klonk als een goede oplossing. "Nee, je mag die tabellen uitlezen, maar als je er op gaat muteren verlies je alle support en dat wil je niet."
"Maar als we dan zelf dat proces van opslaan overnemen, dan hebben we ook controle over wat we in dit traject doen." De broodjes werden vergeten en al snel zaten de deelnemers voor een whiteboard waarop deze variant werd uitgedacht.

Alle processen blijven bestaan, echter na elke stap in het proces, die zorgde voor een lange wachttijd, zou het proces opslaan waar deze gebleven was (opslaan van zijn state) en afsterven. Een externe regisseur bepaalt bij het optreden van een event welk proces moet worden opgestart. Het proces neemt het event, en bepaalt of hij al gedraaid heeft voor het onderwerp van het event. Als dat zo is dan haalt hij zijn gegevens op en start met het event op de juiste plaats in het proces. Voordeel hiervan is dat je in het BPEL-proces nog steeds het hele proces terugziet (de gebruiker herkent het) en elk proces draait nog maar heel kort.
Aan het eind van de workshop stond er een draft van deze oplossing. In de weken daarna werd een prototype gemaakt van de regisseur, wat nog best een lastige klus was. Er werd een locking mechanisme aan toegevoegd, want je mag een proces maar één keer voor één onderwerp hebben draaien. Events die tegen een lock aan liepen werden geparkeerd.
In de opvolgende pilot toonde de regisseur aan dat het mechaniek werkte en werd dit de nieuwe strategie voor het implementeren van BPEL's. Voldaan en na veel rework werd het project weer voortgezet. Hoe hadden ze dit nu van te voren moeten weten?

Inflight migration tool

Oracle is momenteel bezig om aan BPEL PM een inflight migration tool toe te voegen. Dit kan veel van de beschreven problemen voorkomen en een goed alternatief zijn. Hoewel ook een aantal vragen nog moet worden beantwoord, voordat je zeker bent dat het een goed alternatief is:

Hoe definieer je de migratie?
Kun je conversies uitvoeren (van data of extra data) tijdens de migratie?
Hoe lang kost de migratie per proces?
Kun je ook rapportages krijgen van deze migraties?
Hoe map je de bron-processtappen naar de doel-processtappen, kan dat voorwaardelijk?

Conclusie

De in dit project gekozen oplossing hoeft niet overal een oplossing te zijn, de andere alternatieven kunnen situationeel ook een goede oplossing zijn en gewoon werken. De hier gekozen oplossing is technologisch complex. Bezint eer ge begint!

Dit artikel is geschreven op basis van ervaringen in verschillende projecten. De ervaringen zijn echt, het beschreven project is bedacht.

Ruben Spekle (ruben.spekle@capgemini.com) is Vakgroepleider Oracle Fusion Middleware en Architecture binnen Capgemini en binnen SOA projecten werkzaam als Software Architect.