De duiveltjes in Oracle's roadmap

Interview met Massimo Pezzini

Op Oracle Fusion Middleware Forum in Amsterdam in november 2008 hield Massimo Pezzini, vice president en distinguished analyst bij Gartner Research, een lezing met de titel 'Application Platform Futures: SOA, Cloud and XTP'. De lezing ging onder meer over Extreme Transaction Processing (XTP), en andere ontwikkelingen in de nabije toekomst, maar hij stipte ook andere zaken aan. Optimize praatte met hem.

Hoe internationaal georiënteerd ook, aan sommige dingen merk je toch dat Massimo Pezzini in de eerste plaats een Italiaan is. Spaanse wijn bij de lunch is hem een gruwel ('Franse wijn kan ik toleren, maar Spaanse wijn, nee') en een lezing in een Nederlandse kerk die zelfs nog gebruikt werd, kan hem al evenmin bekoren ('Praten over SOA en applicatie-integatie in een kerk, dat geeft me toch een heel vreemd gevoel'). Gelukkig bleek Pezzini echter vooral een zeer aangename gesprekspartner, die vrijelijk van gedachte wisselde over vrijwel elk denkbaar onderwerp, vooral wanneer het IT-gerelateerd was.

Pezzini: "Wat is een applicatie in een SOA-omgeving?"

Voorafgaand aan het interview gaf Pezzini een presentatie. Daarin gaf hij een visie op SOA en op de ontwikkelingen op het gebied van softwareontwikkeling en softwarearchitectuur in de komende jaren. Een van de trends die hij voorzag was Extreme Transaction Processing. Kennelijk zijn we hard op weg naar de realisering daarvan, want volgens Pezzinni zou XTP al rond 2012 realiteit moeten zijn. Uiteraard is Oracle één van de partijen – Pezzini sprak tenslotte op een Oracle-bijeenkomst – die zeer waarschijnlijk te zijner tijd een geschikt platform zal bieden voor XTP. Een belangrijk onderdeel van een dergelijk systeem is gedistribueerde caching, zoals Oracle die sinds de overname van Tangosol onder de naam Coherence aanbiedt. De rol van SOA binnen XTP roept vragen op over de haalbaarheid van XTP. Volgens Pezzini is daarvoor – behalve gedistribueerde caching - een snel volwassen wordend onderdeel nodig: de registry repository. Maar waarom is die noodzakelijk? Is een ESB niet genoeg?

Pezzini: 'Wat is een applicatie in een SOA-omgeving? Een hoop bewegende documenten. Je hebt in een gemiddelde SOA misschien wel veertig of vijftig services die je bij runtime gebruikt. Je hebt de services, de applicaties die de services consumeren, de services zijn geassocieerd met een beschrijving van de services zelf zoals het WSDL-document dat de interface van de service beschrijft, je hebt procesmodellen. Vaak heb je ergens services en een workflow BPM tool die ze met elkaar verbindt. Daarnaast is er additionele complexiteit, doordat sommige services over meerdere applicaties heen gedeeld worden. Een gebruikelijk voorbeeld betreft de service *customer information*. Dat is software die je kunt aanroepen om de naam en andere data van klanten te krijgen. Zeer waarschijnlijk gaan zeer veel applicaties die services delen. Wat is dus een applicatie in een SOA-omgeving? Het is moeilijk te zeggen. Vroeger was een applicatie een stuk software dat je kon deployen, beheren. In een SOA-omgeving gaat de notie van een applicatie steeds meer verloren, want er zijn heel veel bewegende delen en sommige daarvan zijn ook gedeeld door verschillende gebruikers over verschillende applicatieomgevingen. Het beheren van de lifecycle van die applicaties is dus behoorlijk gecompliceerd,

omdat je al die kleine bewegende delen in de gaten moet houden. Vroeger had je een grote verzameling COBOL-code en dat was de applicatie. In een SOA-omgeving heb je een verzameling services, maar die breiden zich vrij langzaam uit. Je hebt ook de front-end applicatie die die services consumeert en die beweegt veel sneller. Stel je een Siebel-applicatie voor. Je hebt een portal, voor employees of een e-commerce-applicatie voor consumenten, en deze front-end applicaties consumeren back-end services die klantdata, productdata, facturering en dergelijke verzorgen. Die services veranderen heel langzaam, omdat ze standaardfunctionaliteit bevatten die niet zo snel verandert. Maar het front-end verandert voortdurend, omdat je steeds meer nieuwe producten en diensten hebt. Je hebt al die bewegende dingen over meerdere locaties, dus je hebt ook iets nodig dat bijhoudt wat er allemaal gebeurt. Dat is de registry repository. Wat is het? In wezen een database die informatie over de services opslaat. Die informatie gaat dan om welke services je hebt, hoe ze eruit zien, welke capaciteiten ze implementeren in termen van functionaliteit, kwaliteit van service in termen van availability, performance. De database slaat echter ook informatie op over de applicaties die de services consumeren. Voor iedere front-end applicatie zou je misschien willen weten welke services die applicatie consumeert en vice versa: je zou ook van iedere service willen weten welke applicaties hem gebruiken. Je wilt in de repository alle metadata opslaan met betrekking tot die applicaties, zoals de WSDL-documenten en het BPEL-script dat de processen beschrijft. Op één enkele plaats, in één repository heb je dan alle informatie die je nodig hebt om de lifecycle van je applicatie te beheren."

"Die informatie wordt gedurende de gehele lifecycle van de applicatie gebruikt. Op design time bijvoorbeeld, wanneer je een nieuwe applicatie bouwt en de registry wilt doorzoeken om te kijken welke services je opnieuw wilt gebruiken. Als je een service portal voor klanten wilt bouwen bijvoorbeeld, ga je kijken of er al een service is die klantinformatie aanbiedt. Je gebruikt de registry repository op design time, op development time, bij het beheer. Stel dat je een service moet wijzigen omdat je een bug gevonden hebt. Je moet dan weten welke applicatie die service gebruikt."

"Je hebt die informatie ook op runtime nodig, de ESB heeft informatie nodig waar de service is, SOAP payload die heen en weer gaat enzovoorts. De registry repository kan zelfs op runtime worden gevoed, onder meer met statistische data. Je wilt bijvoorbeeld weten welke services het meest gebruikt worden, je wilt de availability kennen, de performance. De ESB kan dus de registry repository niet alleen gebruiken, maar ook van gegevens voorzien. Ze werken dus samen gedurende de gehele lifecycle van een applicatie."

"Bij twintig services zien de meeste mensen de noodzaak van een registry repository niet in. Wanneer je er echter vijftig of honderd hebt, dan ziet het er anders uit. Het wordt dan echt moeilijk om op de hoogte te blijven van wat er gebeurt met de services, dus je hebt dan gewoon een registry repository nodig. Maar de database op zichzelf is niet genoeg, je hebt ook processen nodig, die ervoor zorgen dat de inhoud van de repository actueel is, een organisatie die het ondersteunt, SOA-governance met andere woorden."

Zijn er al volwassen producten die dit idee ondersteunen?

"De eerste registry repository-producten begonnen twee, drie jaar geleden op de markt te verschijnen. Systinet van HP, Centrasite van Software AG, Websphere Registered Repository van IBM, BEA en Oracle hadden eveneens toen al producten. Er is zelf open source technologie, Galaxy van MuleSource."

En hoe volwassen zijn ze?

"Tegenwoordig hebben de meest succesvolle producten twee, driehonderd klanten. Laten we zeggen dat deze producten nog steeds volwassen aan het worden zijn, maar ze zijn goed genoeg voor de minder complexe SOA-initiatieven. Bedrijven met de meest succesvolle langer lopende SOA-initiatieven, ik denk dan aan tien jaar of zo, hebben dit concept zelf uitgewerkt. Sommigen zijn nu bezig om te kijken of ze kunnen migreren van hun zelfgebouwde oplossingen naar commerciële oplossingen. Ik denk dat deze producten waarschijnlijk nog een paar jaar nodig hebben om krachtig genoeg te worden om de meest complexe omgevingen te ondersteunen."

Waar laten ze nog te wensen over?

"Een voorbeeld is het federatie-probleem. We hebben ontdekt dat het bij grote organisaties, of zelfs bij organisaties van gemiddelde grootte, heel moeilijk is om één SOA te implementeren. Je hebt verschillende business units, afdelingen, en het houden van één consistente set van technologieën en governanceprocessen over zo'n grote organisatie is heel moeilijk. Het probleem is dus dat we meerdere SOA-initiatieven zullen hebben die naast elkaar bestaan binnen één organisatie, en op een zeker moment moet je die integreren of federeren. Een probleem is: hoe federeer je meerdere registry repository's? In een afdeling heb je bijvoorbeeld SAP en gebruik je een SAP registry repository, in een andere afdeling heb je Oracle met de Oracle registry repository. Het probleem is: hoe zorg je ervoor dat die twee registry repositories informatie kunnen uitwisselen? Van een SAP-applicatie kun je data uit een Oracle database halen, maar hoe haal je data uit de ene registry repository naar de andere? Daar zijn geen standaarden voor."

Idealiter heb je maar één registry repository.

"Ja idealiter wel, maar stel je een groot bedrijf als ING voor. Ze hebben heel veel verschillende business units en bedrijven. Dat al die verschillende eenheden dezelfde ESB, dezelfde registry

repository en dezelfde tools hebben is een utopie. Dat gaat nooit gebeuren. Het probleem is dat er geen standaarden zijn. Je moet nu zelf point to point verbindingen gaan schrijven, wat vrij problematisch is. Maar dat is één van de problemen waarvan we mogen verwachten dat die waarschijnlijk opgelost zullen zijn in twee of drie jaar. Maar voor 'standaard' SOA zijn de commerciële producten die nu verkrijgbaar zijn al goed genoeg."

U sprak tijdens uw lezing over caching-technologie, die XTP mogelijk zal maken. Denkt u dat dit – gecombineerd met de hardware-veranderingen – tot nog veel grotere veranderingen in de manier waarop we software ontwikkelen ten gevolge zal hebben?

"Absoluut. Als je kijkt naar de fundamentele trends in hardware: je hebt 64 bit, wat betekent dat je enorme hoeveelheden geheugen kunt adresseren. Verder heb je multicore, dus in een kleine chip heb je vier of meer computers die parallel werken. Dan zijn de kosten van geheugen ook nog eens heel erg gering en de netwerken zijn zeer snel. Vanuit het hardware-perspectief is dat een paragdigma-verandering. We zijn gewend geweest om applicaties te bouwen die sequentieel waren. We praten al heel lang over parallel computing. Maar een paar jaar geleden betekende het nog dat je enorme hoeveelheden geld moest uitgeven. Nu is het commodity."

"De basisaannamen voor computing zijn veranderd. Vroeger was geheugen schaars en duur en rekenkracht eveneens. Mijn laptop is waarschijnlijk krachtiger dan vele mainframes. Mijn Blackberry is sneller dan mainframes van vijf jaar geleden. Rekenkracht is goedkoop en commodity, grote hoeveelheden geheugen eveneens, het wordt dus waarschijnlijk tijd om opnieuw na te gaan denken over een aantal van de fundamentele principes van computing. Parallel computing, in memory caching, gedistribueerde computing, het is allemaal commodity. Van al die trends kan echter nauwelijks of niet geprofiteerd worden door de softwarearchitectuur van nu. We hebben dus een verandering nodig in de softwarearchitectuur. Ik geloof oprecht dat het XTP-concept het softwarefundament vormt dat het mogelijk maakt om te profiteren van veel van deze hardwaretrends. Als je erover nadenkt, het caching-concept maakt het mogelijk om gedistribueerde applicaties parallel te laten werken. Je kunt meerder applicaties op dezelfde multcicore server laten draaien, óf je kunt applicaties samen laten werken over een sterk gedistribueerd netwerk, misschien met sommige delen van het netwerk in de cloud, samenwerkend volgens hetzelfde paradigma: schrijven en lezen van die cache. Zie je, dat is een zeer schaalbaar concept. Je kunt het toepassen in je laptop, door data in de cache te delen en je kunt honderden computers over de gehele wereld laten samenwerken, alweer door dit gemeenschappelijke mechanisme om data uit te wisselen. Ik geloof dat dit veel fundamenteler is dan we denken. Ik kan me voorstellen dat deze caching-technologie databases kan vervangen - zij het niet helemaal. Ik kan me zelfs voor-

Pezzini: "Het probleem is: hoe zorg je ervoor dat die twee registry repositories informatie kunnen uitwisselen?"

stellen dat het ESB's kan vervangen. Applicaties die van de cache lezen en schrijven, ziet dat er niet uit als een publish&subscribe-mechanisme? Je hebt applicaties die over en weer berichten uitwisselen via deze infrastructuur, wat in feite het doel is van een ESB. We zullen zien, misschien droom ik, maar er zijn nu al producten die dit nu al kunnen doen. Het is al een gegeven, iets wat nu al gebeurt, geen sciencefiction. Wat het zo fraai maakt is dat het concept neerkomt op een in memory database."

Maar dat is iets wat Oracle al heeft.
"Ja, maar TimesTen is een relationele database, terwijl het hier gaat om objecten. Sommige plannen van Oracle zijn echter veelbelovend. Zo wil het de BPEL-process manager, ESB en BAM allemaal via Coherence laten lopen. Kijk naar de BPEL Process Manager. Ieder proces heeft een state. Zo'n state is een document, waarin staat welke stappen uitgevoerd zijn. Wat er gebeurt, is dat het proces van tijd tot tijd wordt opgeslagen op schijf. Dat gebeurt bijvoorbeeld wanneer een proces gestopt is en ergens op wacht. Wanneer het event waarop het proces wacht plaatsvindt, dan wordt het proces 'rehydrated', wat betekent dat de state van disk gelezen wordt, en dat is tijdsintensief. Maar wat gebeurt er wanneer je die state niet in een document opslaat, maar in Coherence? Dan is het enorm veel sneller. Oracle is van plan om de BPEL Process Manager met Coherence te combineren om hem sneller en schaalbaarder te maken. Dat levert weer de mogelijkheid op om veel meer parallelle processen te ondersteunen. Zie je, het is zeer fundamentele technologie, die je met veel producten kunt combineren om productiviteit en performance te vergroten. IBM is hetzelfde van plan met hun caching-technologie, Websphere Extreme Scale.'"

Nu we praten over Oracle en SOA: Oracle is een voorbeeld van een bedrijf dat een enorme uitdaging heeft op dat gebied. Ik bedoel daarmee dat ze bezig zijn allerlei producten te integreren binnen hun eigen platform die ook via webservices met elkaar communiceren.
"Op het gebied van SOA heeft Oracle twee belangrijke uitdagingen. Eén is de integratie van BEA, wat een hoop implicaties heeft vanuit een economisch perspectief. De strategie van Oracle is om het beste van twee werelden de kiezen, dus het beste van BEA, het beste van de orginele Fusion middleware en ze te combineren. Dit is geen triviale opgave want er zijn afhankelijkheden tussen de verschillende producten. De Oracle applicatieserver vormt het fundament voor Oracle ESB, voor de Oracle Process Manager, BAM et cetera. Ze moeten dus ook daar overal Oracle AS vervangen door Weblogic, het is geen onmogelijke opgave maar moet wel gebeuren. In sommige gevallen is het de strategie van Oracle om producten samen te voegen, zodat capaciteiten en eigenschappen van het ene product in het andere geschoven worden. Technologisch gezien is dat een echte opgave en dat geldt ook voor het omgaan van de installed base, die trouwens ook erg groot is. Dat is een set van uitdagingen. De andere is die aan de kant van de applicaties. Oracle heeft een grote set van applicatiepakketten verzameld, PeopleSoft, Siebel, JDEdwards, Jlog, Ritech, Iflex, Petasoft, Hyperion, er komt geen einde aan. Het uitbrengen van geïntegreerde architectuur vanuit een applicatieperspectief is niet eenvoudig. Ik geloof zelfs niet eens dat ze dat proberen. In mijn begrip is de strategie van Oracle voor applicaties tweevoudig. De eerste strategie komt neer op het leveren van een set van geïntegreerde processen gebouwd bovenop Oracle Fusion Middelware om die applicaties met elkaar te verbinden over industriespecifieke processen. Dat is één stap, het voorzien van klanten met voorverpakte integratieoplossingen en processen, door diverse applicaties van Oracle en zelfs van anderen met elkaar te verbinden. De tweede strategie bestaat uit de nieuwe generatie van wat Oracle Fusion Applications noemt, de volgende vernieuwing van voorverpakte oplossingen die geacht worden het beste van Peoplesoft, Siebel en dergelijken in een geheel nieuwe applicatieomgeving die SOA-enabled te zijn vanaf het begin. In plaats van het aanpassen van alle applicaties om ze SOA-compliant te maken, zorgen ze aan de één kant voor een set tools om die applicaties met elkaar te verbinden en zorgen ze voor een nieuwe set van applicaties die ontworpen zijn volgens het SOA-principe. Natuurlijk gaat dit jaren duren. Het bouwen van een geïntegreerde oplossing met ERP, CRM, SRM, HR en wat dan ook zal ook veel tijd kosten. Maar dit zijn de uitdagingen van Orace. Uiteraard zijn de twee dingen met elkaar verbonden. Oracle Fusion applicaties zullen gebaseerd zijn op Oracle Fusion middelware. Het is een behoorlijk gecompliceerde opgave voor Oracle. Ze hebben echter ook de resources, de technologie en de mensen ervoor, dus, nu ja, we zullen zien. Naar mijn mening zal het langer gaan duren dan de anderhalf jaar die ze beloofd hebben. In juli 2008 kondigden ze de BEA-Oracle integratie roadmap aan, en toen zeiden ze dat hij binnen de komende twaalf tot achttien maanden geïmplementeerd zou worden. Ik geloof dat het eerder twee, drie jaar zal worden. Maar de roadmap is redelijk, vrij degelijk, gebaseerd op redelijke beslissingen bezien vanuit een technologisch perspectief, dus we moeten afwachten. Het probleem ligt in de uitvoering. Ze weten wat ze moeten doen, ze hebben vrij gedetailleerde plannen, maar nu moeten ze die uitvoeren. De Amerikanen zeggen: 'the devil is in the details', dus we zullen zien hoeveel duiveltjes er in de roadmap zullen opduiken."

Dré de Man Tekst en foto's