De CMDB-hoeksteen van de SOA

SERVICE REGISTRIES

Een SOA-omgeving is niet zomaar neergezet. Er zijn relatief technisch ingestelde problemen op te lossen, zoals de services-connectivity. Maar ook veel meer functioneel gerichte uitdagingen zoals ontwerp en eigenaarschap van services. Wat regelmatig vergeten wordt, is de brug tussen deze werelden: een 'registry' die een totaaloverzicht van ons SOA-platform biedt. En daar komt beduidend meer bij kijken dan een simpele via UDDI (Universal Description, Discovery and Integration) ontsloten lijst.

Door Erik de Ruijter

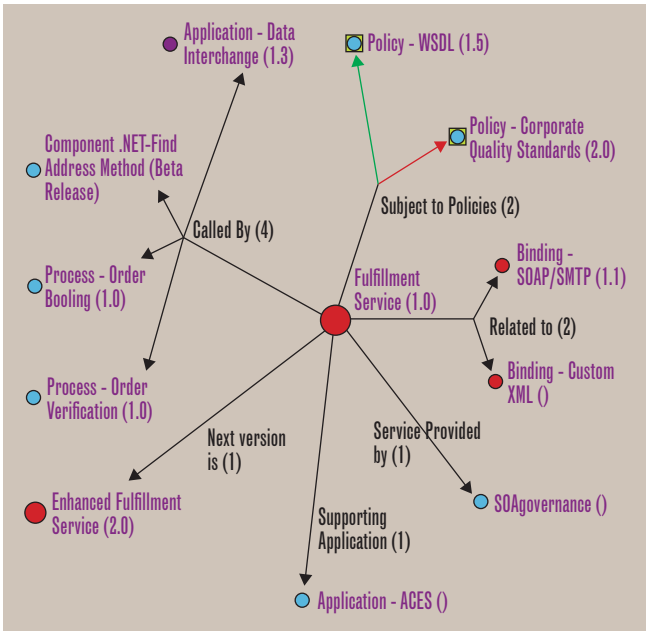
Om de problemen in de toch vrij jonge SOA-wereld goed te begrijpen, maken we een uitstapje naar een meer uitgerijpte IT-wereld, namelijk de beheerexperts met hun ITIL-framework. In het dagelijks beheer, maar ook bij de rapportage over 'business services' en 'assets' kwam men daar al tien jaar terug achter dat het spreken van eenzelfde taal essentieel is. En dat we dat spreken moeten afdwingen door alle diverse ITIL-modulen, van softwaredistributie tot performancemetingen, van één gedeelde waarheid te laten uitgaan. En dat heet de Configuration Management Data Base (CMDB).

Ook in de SOA-wereld is zo'n gedeelde waarheid essentieel, en dat heet hier een service registry of soms ook wel repository. (Of 'repository' is de totaalopslag en 'registry' is de UDDI-view er op.) De basis voor een registry zijn de servicebeschrijvingen oftewel de WSDL-files. En die kunnen worden gepubliceerd via de UDDI- interface. UDDI beschrijft de per service te bewaren data, inclusief zoekvelden - en daarbij ook een specifieke SOAP-interface om de dataopslag te benaderen. Plus de service-hiërarchie oftewel taxonomie, maar die is altijd incompleet, omdat van 'top level' services niet de gebruikende objecten geregistreerd worden.

Oorspronkelijk was UDDI een handige hulp om runtime te zoeken naar bijvoorbeeld 'de dichtstbijzijnde afstandsberekenings-service', maar publieke webservices en bijbehorende registries op internet lijden een zieltogend bestaan. Wél populair is UDDI binnen ontwikkelaarsteams: vanuit hun IDE-tool kunnen ze de juiste service opzoeken en daarna diens WSDL in hun applicatie includen. De oorspronkelijk voorziene 'runtime-mapping' kan veel beter via een mechanisme als de DNS-namen van servers plaatsvinden - en dient juist binnen organisaties nooit naar een variabele service toe te wijzen.

Maar willen we de service registry op CMDB-niveau krijgen, dan is veel meer nodig dan alleen WSDL en zoekvelden. Er is dan echt aanvullende 'metadata' nodig en rondom de registry ook tooling voor de service-levenscyclus. Want als er sprake is van meerdere versies van een service dan moeten die op de juiste momenten doorzoekbaar en zichtbaar worden. En dat vereist workflow, logging en dat soort functionaliteit.

Nog een stap verder is inzet van de registry voor het totale SOA-proces, inclusief minimaal de service-afnemers. Om dit waar te maken moeten ook metadata van alle andere applicatie-objecten bewaard worden: denk aan Java JSP's, Dotnet ASP's, BPEL scripts die allemaal aanroepers van SOAP-services kunnen zijn. En geven we per aanroeper (ook als die zelf webservice is) aan welke services gebruikt worden. Via handmatig onderhoud, maar liever nog via automatische scan van de code of, nóg mooier, door auditing binnen de applicatieser-



Figuur 1: OER service levenscyclus.

ver. Bij dit laatste niveau, met een echte 'ist' meting, kunnen we ook optioneel het volume van de aanroepen laten loggen, zodat SLA's en doorbelasting vanuit de registry bewaakt kunnen worden. En in alle varianten kunnen we minimaal vragen beantwoorden zoals 'wie gebruiken er allemaal services F en G, zodat we ze moeten consulteren, voordat een wijziging gepland wordt?'.

De registry-praktijk

Tot zover wat wensen voor een ideale invulling. Maar de praktijk kan weerbarstig zijn, omdat niet alleen het eigenaarschap van (gedeelde) services functionele hoofdpijn oplevert, maar ook de verantwoordelijkheid voor de registry. Oracle heeft een mooi referentievoorbeeld bij een klant, een middelgrote Nederlandse financiële instelling. Die had reeds de hele Oracle J2EE- en ESB-stack en ook een licentie op de OER/OSR repositories. Het management vond echter dat het project om de repositories te gaan gebruiken ten koste moest gaan van de bouwprojecten. Die weigerden vanwege de perceptie van kostenpost en dus blok-aan-het-been, een klassiek managementprobleem. De IT-afdeling besloot toen de repository in de pauzes en avonduren te vullen.Er bleken tot tien verschillende versies van eenzelfde business service en officieel 'end-of-life' zijnde services te zijn die nog volop werden aangeroepen en nog meer aanzienlijke besparingskansen. En toen die aangegrepen werden, was de business case bewezen en mocht de registry voortaan in de tijd van de baas en op van de projecten afgesnoept budget verder uitgebouwd worden.

De Oracle stack en SOA Governance

Oracle heeft drie hoofdproductlijnen en een bijwagen. De hoofdlijnen zijn de database, de applicaties en de 'Fusion Middleware'. En de bijwagen is het beheertool Oracle Enterprise Manager (OEM), dat voor alle productlijnen ingezet wordt. De basisversie wordt gratis met het betreffende Oracle-product meegeleverd en is dus een soort 'box manager'. Maar OEM kent ook betaalde management packs en hiermee uitgebreid is het een soort lichtgewicht concurrent op de overall ITIL-beheermarkt.

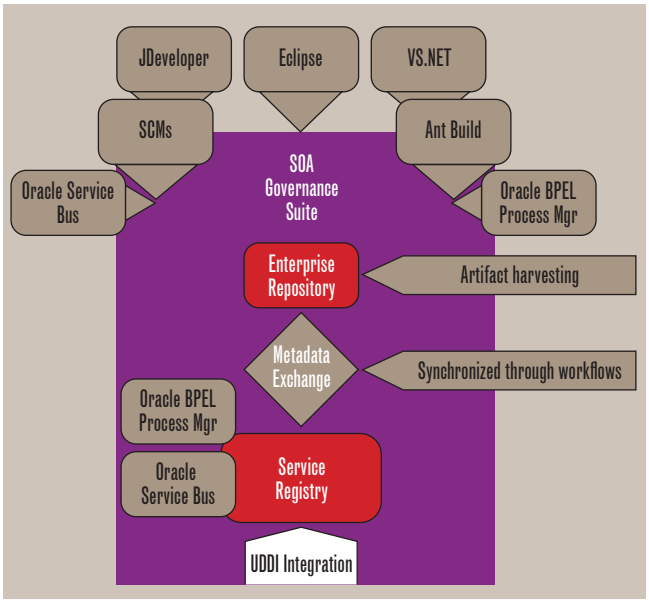
De Fusion Middleware bestaat uit de basis Java EE-laag, Oracle WebLogic, plus allerlei extra tools erbovenop; zoals de SOA Suite met workflow en business process management, de bijbehorende JDeveloper IDE waar we grafisch programmeren (en daarmee Java genereren) en eventhandling.

We zoomen nu in op de tools die Oracle groepeert als 'SOA-governance'. En dat zijn zelf weer drie aanzienlijk afwijkende soorten bouwstenen:

- Service registry', dus alle SOA-metadata. En die wordt gevormd door Oracle Enterprise Repository – OER – en Service Registry dus OSR.
- Web Services Manager. Dit is een 'SOA Policy Manager' vergelijkbaar met onder andere de HP SOA Policy Enforcer. Dit soort tools kan nuttig zijn voor onder andere afdwingen van de WS-Security standaard buiten de applicatieserver om en voor realtime afdwingen van servicecontracten en meten van servicegebruik, maar het blijft toch een kleine marktniche; zolang alles in huis draait is alle functionaliteit ook via andere routes (onder andere in de applicatieserver en in batchrapportage vanuit OER) in te vullen. In heterogene omgevingen kan het beeld anders zijn, bijvoorbeeld voor webservices binnen een standaardpakket (zonder WS-Security) of oudere Dotnet versies of van een Internet-ASP.
- Het ITIL-beheer (fout en performance) specifiek op de SOA-omgeving. Momenteel biedt Oracle alleen het 'Management Pack for SOA' bovenop OEM, en dat toont onder andere de hele serviceketen en de responstijden. En recent is er het 'Composite Application Performance Management' pack bijgekomen, bestaande uit de vorig jaar opgekochte ClearApp technologie. Dit voegt nog niveau's van diepgang toe in dit service-beheer, zoals de database-dimensie.

Oracle invulling – OSR

Oracle werkt met twee gescheiden, maar nauw gekoppelde, producten. OSR is een OEM-versie van 'HP SOA Systinet', een pioniersproduct op de UDDI-markt. En daarmee primair bedoeld voor de basisvelden die opgeslagen worden van webservices (de WSDL plus zoekvelden); ook kunnen XML, XSD en XSLT bestanden opgeslagen worden. De doorzoekbaarheid is uitstekend, maar de tool ontbeert een workflow, crossreference en versiebeheer. Dat kan ook allemaal prima, want Oracle zelf positioneert hiervoor OER (met OSR als een soort slave-opslag). OSR kent wel een beperkte workflow via twee fysieke instances, de 'discovery registry' en 'publication registry' - maar in een serieuze Oracle-omgeving wordt dit eigenlijk niet gebruikt, omdat OER dan alle versies bijhoudt en alleen de dán voor developers toegankelijke versies aanwezig mogen zijn in OSR.



Figuur 2: relatieschema in de OER console.

SAG CentraSite: breed inzetbare pionier

Dit artikel heeft weliswaar de Oracle-toolset als hoofdonderwerp, maar om een completer en breder beeld van de markt te geven, kijken we ook beknopt naar de repositories van twee andere leveranciers.
CentraSite, ontwikkeld door Software AG samen met Fujitsu, heeft sindsdien niet ingezet op specifiek diepere webMethods-integratie maar juist op een brede plek in de markt. Om die te bereiken is er sinds vorig jaar een gratis 'community edition'; de betaalde enterprise edition voegt onder andere workflow toe.
CentraSite is zowel bedoeld voor de UDDI-data als voor metadata van andere artefacten - bijvoorbeeld schermen. Voor de WSDL-artefacten kan zelfs direct in CentraSite geviewed en ge-edit worden, wat zeker bij gebruik van WSDL-attributen voor het definiëren van een SLA en servicecontract een duidelijk pluspunt is. CentraSite kent een behoorlijke set koppelingen met repositories, maar ook met bouwtools. Zo kan er met Business Objects gebabbeld worden, en is de repository zelf weer zichtbaar te maken in zwaardere rapportagetools zoals IBM Cognos. Software AG zet hem echt neer als het 'SOA governance' product in de keten, inclusief het meten van servicegebruik op een vergelijkbare manier als OER dat aanbiedt. Voor SOA policy enforcement and mediation positioneert SAG de webMethods X-Broker, het voormalige Infravio produkt. Dit opereert meer in het marktsegment van Oracle Web Services Manager, met actief bewaken van service security en aansluitkontrakten.
Samengevat heeft Software AG een grofweg vergelijkbare functionaliteit als Oracle. Doch met betere interne integratie (UDDI gewoon aan boord), soms iets meer kracht (WSDL edit) en met een gratis instapversie die een vrij brede marktacceptatie begint te krijgen.

Integraties zijn essentieel voor een registry, en OSR zal er in een totale omgeving minimaal twee hebben:

1. Met OER, waarbij OSR als een soort 'slave' alleen de vereiste objecten opgeslagen heeft;
2. Met de IDE. Ondersteund worden Oracle JDeveloper maar ook een generieke Eclipse-plugin (voor o.a. IBM en Software AG/webMethods) en Microsoft Visual Studio.

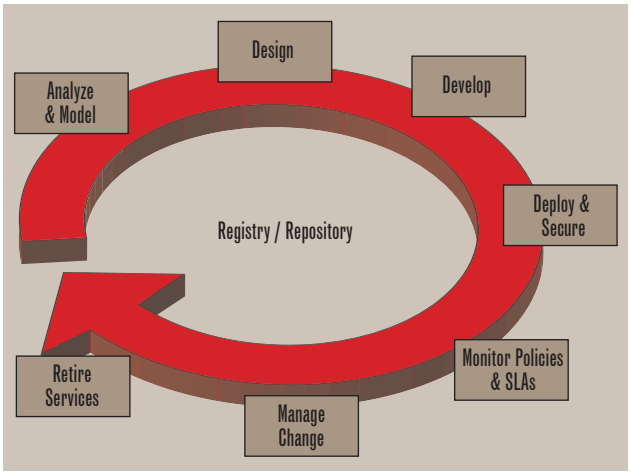
Op het eerste gezicht is opvallend dat OSR dus niet zelf babbelt met code-repositories zoals IBM ClearCase, Borland StarTeam of Serena PVCS. Die integratie gebeurt namelijk in de IDE.

Oracle OER: functionaliteit en integraties

Oracle Enterprise Repository begon zijn leven als product van het pioniersbedrijf Flashline, dat in 2006 onder BEA-vleugels belandde en daarmee dus in 2008 onder de Oracle-vlag kwam.
OER levert 'alle eerder in dit verhaal gegeven functionaliteitswensen met uitzondering van de UDDI-interface'. In zekere zin een wat gekunstelde constructie, maar Oracle geeft aan dat ze vooralsnog tevreden is met de OER-OSR tandem en dus in tegenstellign tot de meeste concurrenten niet naar één fysiek tool streeft. OER schopt onder de motorkap OSR aan en uiteindelijk hebben we volgens Oracle dus een geïntegreerd 'virtueel tool'.

Die wensen en functionaliteiten komen, samengevat, neer op het volgende:

1. Opslag van de metadata van álle artefacten in onze SOA-omgeving.
2. Levenscyclus-beheer op deze metadata, liefst gesynchroniseerd met de cyclus van de fysieke code. Vandaar dat het mogelijk is om promotie- en buitengebruikstellingsworkflow in OER zelf te configureren, maar evengoed om de onderliggende acties aan te schoppen vanuit een IDE of vanuit een code repository. Er worden ook voorbeeldstructuren en workflows meegeleverd, bijvoorbeeld voor architectuurgoedkeuring, op basis van best practices bij klanten. En aanschoppen kan zelfs vanuit het bouwscript, zoals ANT voor Java, dat vanuit de IDE zelfstandig werkend de code-voorbereiding doet. Ook is een koppeling mogelijk met softwaredistributietools zoals Microsoft SCCM of Tivoli of zelfs Oracle's eigen OEM, indien die gebruikt worden als leidende workflow in SOA-distributie. En kunnen we, net als in OSR trouwens, notification instellen: groepen mensen (bijvoorbeeld de eigenaren van een gebruikende applicatie) kunnen een bericht krijgen zodra service X nieuwe versies of andere changes krijgt.
3. Bijhouden van de onderliggende relaties tussen de artefacten. Soms kan deze alleen maar handmatig worden onderhouden, maar de voorkeur is juist vanwege de forse aantallen een volautomatisch proces. Voor sommige artefacten kan OER bij het ophalen van de metadata een 'introspection' doen - bijvoorbeeld een BPEL script kan worden geparsed en vervolgens worden naast de metadata ook alle aanroep-calls als relaties in OER opgeslagen. We praten hier nog steeds over de 'soll' dimensie, hoe code zich volgens de regels zou moeten gedragen - inclusief zaken die nog in een testfase zitten. Wat OER daarnaast kan is volautomatisch de volledige 'ist' dimensie van relaties binnenhalen, natuurlijk alleen voor code in produktie. Dit gebeurt vanuit de J2EE- of Dotnet server. In geval van Oracle WebLogic kan deze een audit trail van alle SOA-calls bijhouden en dat via de gratis versie van OEM laten exporteren naar OER; voor andere servers zijn vergelijkbare bruggen mogelijk. En, als toefje slagroom op de functionaliteit,

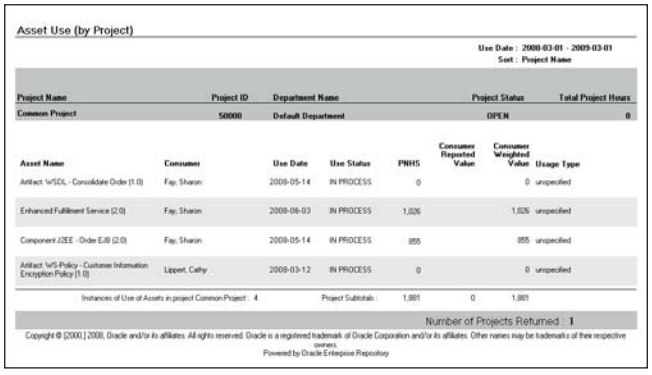


Figuur 3: Synchronisatie-schema rondom OER-OSR.

zijn desgewenst ook de volumetrics van de onderlinge aanroepen te loggen en door te geven aan OER. Waardoor rapportage mogelijk is zoals 'service P-1.01 wordt gebruikt door JSP's uit applicaties X, Y en Z en wel met een volume van respectievelijk 100, 10.000 en 5 calls per uur'. De integraties die OER hiervoor nodig heeft zijn ongeveer dezelfde als OSR: vanuit de IDE of elk ander tool dat de master-workflow kent en onderwater met OSR. En daarnaast, voor de crossreference van de ist-situatie, met de applicatieserver - soms via een tussenstation zoals OEM. Er is ook een interface 'naar de IDE toe'. Dus de ontwikkelaar kan via UDDI services zoeken in OSR maar ook via een andere interface in de OER-opslag alle daar aanwezige artefacten-metadata.

De balans

Een service registry vormt de 'gedeelde waarheid' tussen ons IDE ontwikkeltool, de code-opslag in repositories of filesystemen en de applicatieserver. Er worden dan ook hoge eisen aan gesteld qua integratie-opties, maar ook qua workflow en rapportagemogelijkheden.
De Oracle-tools voldoen aan die eisen, waarbij opgemerkt moet worden dat de integraties logischerwijs het diepste uit-


Asset Use (by Project)

Use Date : 2008-03-01 - 2009-03-01
Sort : Project Name

Project Name	Project ID	Department Name	Project Status	Total Project Hours
Common Project	50000	Default Department	OPEN	0

Asset Name	Consumer	Use Date	Use Status	PNHS	Consumer Reported Value	Consumer Weighted Value	Usage Type
Artifact: WSDL - Consolidate Order (1.0)	Fay, Sharon	2008-05-14	IN PROCESS	0		0	unspecified
Enhanced Fulfillment Service (2.0)	Fay, Sharon	2008-06-03	IN PROCESS	1,026		1,026	unspecified
Component J2EE - Order EJB (2.0)	Fay, Sharon	2008-05-14	IN PROCESS	855		855	unspecified
Artifact: WS-Policy - Customer Information Encryption Policy (1.0)	Lippert, Cathy	2008-03-12	IN PROCESS	0		0	unspecified
Instances of Use of Assets in project Common Project : 4			Project Subtotals :	1,881	0	1,881	

Number of Projects Returned : 1

Powered by Oracle Enterprise Repository

Figuur 4: Asset Usage rapport in de OER-console.

IBM in vergelijkbare situatie

IBM zit enigszins in een vergelijkbare tweespalt als Oracle. In de 'WSSR Advanced Lifecycle Edition', waarbij de afkorting staat voor WebSphere Service Registry and Repository, zijn twee behoorlijk apart ontstane tools samengevoegd: WSSR zelf en Rational Asset Manager (RAM).
Net als Oracle integratiebruggen aanbiedt met softwaredistributie kan de IBM repository, speciaal het RAM-stuk, koppelen met de IBM Tivoli distributietools. En qua opslag van feitelijke code is er vanzelfsprekend de meest hechte integratie aanwezig met de Rational ClearCase versiebeheer-repository.
Als beide tools in tandem gebruikt worden, dus in de Advanced Lifecycle Edition, dan raadt IBM zelfs aan om vanuit de eigen IDE (WebSphere Studio of één van de Rational add-ons erbovenop) niet via UDDI te gaan zoeken in WSRR maar via de direkte integratie in RAM. Daarin is immers meer uitgebreide informatie over elke service beschikbaar.
Interessant neveneffect hiervan is dat de hele 'open' UDDI-dimensie hierdoor bij IBM min of meer afgeraden wordt ten faveure van de propietary integratie met RAM, al wordt UDDI op zichzelf ondersteund voor wie dat in de ontwikkeling wil gebruiken.

Ook bij IBM zien we de kernfuncties rondom de service-CMDB die Oracle (en SAG CentraSite) aanbieden. Met natuurlijk nét wat andere accenten en sterkten/zwakten. Het onderstreept hoe essentieel een service registry, inclusief brede integraties, is voor een serieuze SOA-omgeving!

gewerkt zijn met applicatieservers en andere delen van de Oracle middleware-stack. Maar ook koppelen met bijvoorbeeld Visual Studio is goed mogelijk, die crossplatform-openheid is een mooie dimensie van de web service-markt. Het werken met twee aparte systemen, OSR naast OER, is 'suboptimaal' maar vormt na de nodige configuratie beslist geen hindernis voor functionaliteit en gebruiksgemak.
Ongeveer gelijke kwaliteiten zien we bij de tools die we minder diepgaand bekeken, van Software AG en IBM. Ook zij hebben openheid (mede naar de Oracle applicatieserver en IDE toe!), en het inzetten van een service registry in een multiplatform SOA-omgeving is zoals we hiermee zien écht een haalbare kaart. Maar zelfs met deze zeer wel oplosbare technische hobbels is de strijd nog niet gestreden; functionele zaken zoals eigenaarschap van services (en van de registry zelf) vormen de volgende te nemen 'hobbel'.

Erik de Ruijter is ICT architect bij ABN Amro Nederland.