Bruikbaar platform of data morgana?

SOA-STANDAARDS EN OASIS

Dat we in de SOA-wereld veelal op webservices uitkomen moge duidelijk zijn. Maar waar sommige IT'ers zweren bij snelle invoering van deze standaards, en er de hele Service Bus versneld op willen baseren, tonen anderen de nodige scepsis.

Door Erik de Ruijter

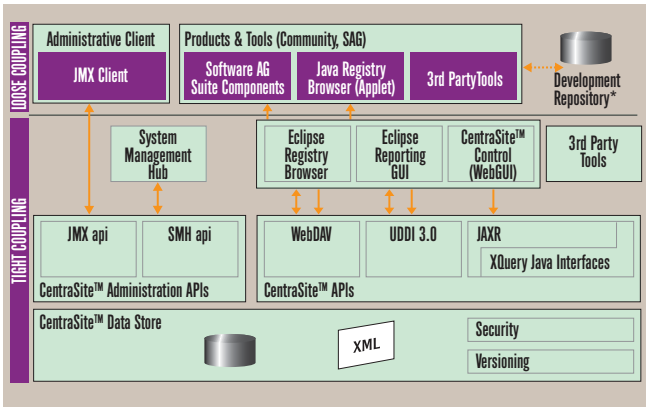
Er bestaan naast succesverhalen ook de nodige minder geslaagde of in ieder geval fors vertraagde invoeringen. We bekijken beide zijden van de medaille: de OASIS/WS-I standaards en de sceptische argumenten. Afhankelijk van de situatie kunnen zowel de fans als de critici gelijk hebben, hetgeen de vooruitgang echter niet teveel zou mogen tegenhouden.

Theorie en platforms

De clubs die zich buigen over webservice-standaardisatie zijn even veelvormig als de hele ISO/IEEE-wereld waaraan ze verwant zijn. In hun missie tonen ze een stuk van de eigen zwakte; met name WS-I (Web Services Interoperability Organisation) levert 'profiles' op om implementaties van de onoverzichtelijke set meervoudig interpreteerbare SOAP-standaarden van OASIS wél met elkaar te laten samenwerken! De 'oerclub' voor de SOAP-wereld is niet IEEE of ISO, want die komen pas in beeld bij technologie-overschrijdende standaards, maar W3C, oftewel het WWW consortium dat ook zaken zoals HTML standaardiseert. W3C kent een aantal basisdefinities zoals XML-over-HTTP en WSDL voor de servicebeschrijvingen, maar het besluitvormingsproces gaat langzaam en de voor SOAP relevante W3C-leden zitten veelal ook in OASIS. En deze club, voluit Organisation for the Advancement of Structured Information Standards, heeft dus het leeuwendeel van de SOAP-standaards op zijn naam; van relatief stabiele en populaire stukken als UDDI (service directory) en SAML (authenticatie) tot meer exotische stukken rondom provisioning en zelfs alerting, en ook 'uitdagende' stukken zoals WS-Security en WS-Transactions. De laatste club, Web Services Interoperability Organisation, heeft zoals al gesteld de ondankbare taak om de OASIS-standaards hapklaar en werkbaar te maken.

De standaards rondom SOAP beloven 'de zevende hemel' en pas als we het ideaalbeeld helder hebben kunnen we ook de bezwaren van de tegenstanders in perspectief zien. Een paar kernpunten:

- SOAP over HTTP is de 'echte' standaard. In alle OASIS-definities is het ook best mogelijk om SOAP-XML op andere layer 7-protocollen te plaatsen. Met name SOAP over messaging middleware (IBM WebSphere MQ, Microsoft MQ en de kleinere spelers Oracle, Sun en Sonic/Progress) komt wel eens voor, maar zelfs de voorgangers COM+ en RMI/CORBA zouden er bruikbaar voor zijn. Al die andere protocollen zijn mogelijk veiliger en stabieler dan HTTP, *maar* ze zijn niet over platformgrenzen heen gestandaardiseerd. Dus willen we pakweg Java en .NET, of Windows en RISC-Unix laten communiceren zonder installatie van proprietary agents dan is het SOAP-HTTP en niets anders waarop we uitkomen.
- Functioneel is het OASIS-portfolio inmiddels in staat om 100 procent van de functies van de 'concurrenten' te bieden. Beveiliging via WS-Security, transactionaliteit via WS-Transactions en MQ-achtige gegarandeerde aflevering op HTTP via WS-(Reliable)Messaging. Dus COM+, RMI en dergelijke zouden allemaal niet meer nodig hoeven te zijn.
- WS-BPEL (Business Process Execution Language) is een



Afbeelding 1: Architectuur van CentraSite, een UDDI service directory.

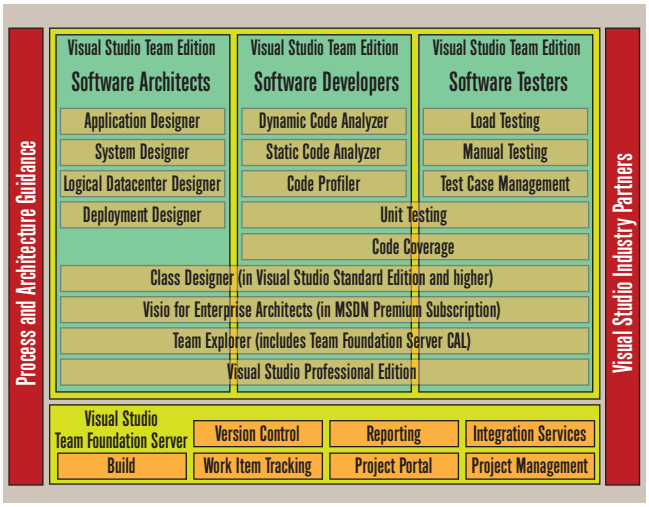
uitstapje in de BPM-wereld dat aan populariteit wint.

- SOAP is een protocol en geen implementatie. Deelnemers aan W3C verplichten zich om voorstel-standaards ook indien er patenten in zitten licentievrij aan te bieden, terwijl OASIS iets meer risico's op auteursrechtproblemen kent. Maar in de praktijk zijn er nog geen specifieke ruzies op dit vlak geweest, elke SOAP-standaard kent zowel closed als open source implementaties.

Service bus sceptici

Voor alle duidelijkheid: de tegengeluiden, vooral uit de praktijk van grotere organisaties, komen níet neer op dat de OASIS webservice-concepten fundamenteel niet goed zouden zijn. De kwaliteiten worden erkend, maar met de huidige implementaties hebben heel wat partijen aardig hun vingers gebrand. Enkele voorbeelden volgen.

- In de tijd voorafgaand aan complete WS-I profielen, en dat was dus tot begin 2007, konden vele vendors claimen 'OASIS versie x' compliant te zijn, maar ze waren tóch niet interoperabel. Dit ligt aan het eerder genoemde karakter: het is een vrij onoverzichtelijke set meervoudig interpreteerbare SOAP-standaarden. Voor elk van de honderden OASIS-participanten (vendor en grote gebruikers en EDI-gerichte branche-organisaties) zitten er wel wensen in verwerkt, maar de S van SOAP staat daardoor zelden nog voor 'Simple'.
- Stacks die WS-I profielen ondersteunen komen pas nu langzaamaan op de markt; onder andere de .NET runtime 3.0 en de nieuwste versies van J2EE vendors. Maar voordat alle gewenste 'domeinen' die een grote organisatie op de service bus wil aansluiten dan ook naar deze applicatie-server-versie gemigreerd zijn, kan dit zelf weer een aardige tijd duren.
- Vandaar dat in brancheverbanden die toch interoperabiliteit nodig hadden, zoals in de betaal- en verzekeringswereld, vaak met eigen stacks gewerkt wordt. Apache AXIS voor Java is een bekende, hij kan op zo ongeveer elke J2EE-server draaien. Van zo'n stack bestaan dan weer meerdere versies die niet altijd onderling compatibel zijn en niet door iedereen vertrouwd worden. JBoss (Red Hat) bijvoorbeeld dumpte vorig jaar AXIS wegens 'kwaliteits- en flexibiliteitsproblemen' en bouwde een eigen 'AXIS compatibele' stack.
- Functionaliteit die in OASIS beloofd wordt hangt soms samen met onrealistische wensen. WS-Transactions is er zo eentje; hij belooft een 'two-phase commit' voor SOAP-transacties.
  Intimi weten hoe moeilijk zo'n model reeds met databases is, en zien het zelfs daar niet altijd werken; invoeringen met de transactionaliteitsopties bij SOAP voorgangers CORBA en COM+ zijn regelmatig gestruikeld, omdat multi-tier transacties gewoon niet goed programmatisch bestendig te maken bleken tegen de complexe realiteit. Niet gehinderd door deze voorkennis herhalen sommige gebruikers (en designers/architecten) deze aanpak nu WS-Transactions wederom een walhalla belooft; ze komen vervolgens van een koude kermis thuis en OASIS krijgt de schuld. Niet geheel terecht in dit geval.
- WS-Security kent minstens twee valluiken. De eerste is dat hij slechts de 'credentials' van de service consumer-applicatie doorzet naar de provider-applicatie, zonodig met encryptie/signering. Zorgen dat zo'n applicatie zonder programmatisch ingrijpen ermee om kan gaan is een uitdaging. J2EE-servers zoals WebSphere en Sun hebben er twee tot drie jaar over gedaan voordat identiteitscheck en access control voor gebruikerssessies configuratief opgelost kan worden en zitten nog aan het begin van deze groeicurve voor WS-Security. En .NET heeft behalve het matige Microsoft Authorisation Manager nooit veel aan standaard tools voor identiteitscontrole gehad, dus eist al snel third-party plug-ins hiervoor. Het tweede valluik is dat alles staat of valt met een goede en schaalbare PKI/X.509-infrastructuur – en daarvan zijn er een veel minder live en probleemloos in het veld dan de security vendors ons willen doen geloven.



Afbeelding 2: Overzicht Microsoft (Dotnet) Visual Studio modulariteit.

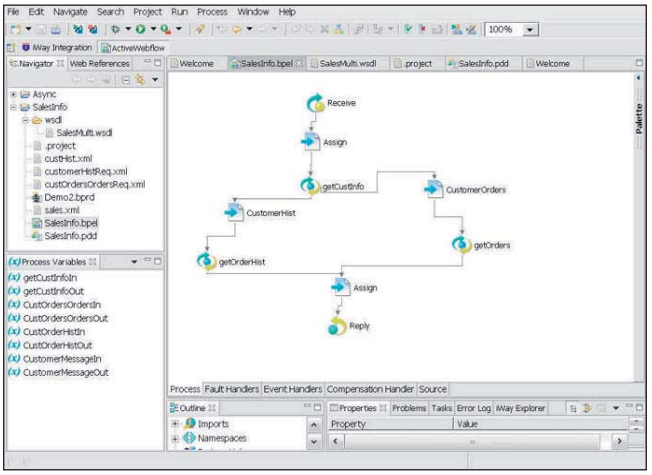
- Het algemene punt van overhead. Vergeleken met RMI of COM+ kost de 'XML (de)marshalling' van SOAP soms 100 tot 200 procent extra rekencapaciteit. En WS-Messaging voegt door allerlei garantie-checks nog een fors stuk extra toe, terwijl de legacy messaging-producten zoals Microsoft MSMQ eenzelfde functionaliteit bliksemsnel kunnen leveren.

Waar de OASIS-sceptici dus op uitkomen, en niet geheel zonder onderbouwing, zijn statements zoals: "Onze stove-pipe-oplossing kan pas op de service bus aansluiten en in overall applicaties integreren als alle middleware op de juiste versies zit. Tot dat moment gaarne budget om onze eigen omgeving separaat te kunnen onderhouden en verbeteren" en "Aangezien cross-platform werken nog faalt, dient iedereen die met ons systeem communiceert over te stappen op .NET versie xx.yy / J2EE-vendor P versie Q.1/MQ merk A optie B" (doorhalen wat niet verlangd wordt).
Want alleen zó menen ze concreet te kunnen werken zonder zich vast te pinnen op de multiplatform beloften van OASIS en WS-I. Die labelen ze, daarvoor zijn het ook sceptici, als een optie in een verre toekomst.

Oplossing en groeipad

Gelukkig hebben de sceptici in de regel 'legacy' systemen, die weliswaar draaien maar tegen ongewenst hoge kosten, en met bepaald suboptimale integraties. Dus hun geloofwaardigheid is niet per se beter dan die van de service bus 'evangelisten'. Maar ook de grootste webservice-scepticus moet na enig delibereren toegeven dat open systemen op de lange termijn beter zijn dan gesloten; en dat het mogelijk moet zijn om naar dat ideaalbeeld toe te groeien in overzienbare stappen. Dus zonder op onrijpe stukken techniek te springen, en zonder vooralsnog haast onhaalbare verwachtingen (zoals WS-Transactions) te wekken. Zo'n set realistische en haalbare afspraken zou bijvoorbeeld kunnen neerkomen op:

- Het kiezen voor de meest beproefde (en waarschijnlijk functioneel meest beperkte) versie van de standaard, als 'grootste gemene deler'. Simpelweg omdat op die wijze meer applicaties kunnen aansluiten. Dus bijvoorbeeld SOAP versie 1.1 in plaats van een WS-I profiel, als we daarmee ook een standaardpakket zoals SAP op de service bus-trein kunnen krijgen.
- Een beperking tot de basisfunctionaliteit: connectivity met voldoende beveiliging en voldoende betrouwbaarheid. WS-Messaging bijvoorbeeld kunnen we in 98 procent van de situaties vermijden, door zodanig te programmeren dat de applicaties zelf een hertransmissie plegen als ze een foutmelding terugkrijgen. WS-Transactions moeten we in veel gevallen zelfs qua gewenste functionaliteit niet willen. Het ontwerpen van single-tier transacties die atomair en terugrolbaar zijn is weliswaar soms tijdrovend, maar vooralsnog structureel veiliger dan de multi-tier beloften (dit zijn échte luchtspiegelingen volgens velen).
- Deze basisfunctionaliteitregel geldt in versterkte mate voor



Afbeelding 3: iWay, een bouwtool voor BPEL/webservices.

  security. Vaak is afscherming op het niveau van VLAN's of tweezijdige SSL-tunnels voldoende robuust en moeten we nog niet denken aan WS-Security met de bijbehorende complexe PKI. Ook kan het een oplossing zijn om met sterk versimpeld certificaatbeheer te werken.
- En last but not least, geen 'big bang' maar een groeipad qua invoering: beginnen met domeinen/afdelingen waar relatief makkelijk een business case voor SOAP te maken is omdat ze tóch aan onderhoud toe waren, en pas later beginnen aan legacy-bastions waar de terugverdientijd veel langer is. Door al op onderdelen proprietary middleware zoals MQSeries of COM+ te elimineren, gaan we bovendien de onderhoudskosten van die algemene infrastructuur proportioneel hoger maken voor de resterende legacy-gebruikers, die uiteindelijk de kostenvoordelen van het op de service bus aansluiten zullen inzien.

Als we dit soort regels aanhouden dan hebben we de tegenwerpingen der sceptici erkend maar ze als blokkade voor SOAP-invoering omzeild. Bijvoorbeeld: "Tot het moment dat alle middleware WS-I compliant is gaarne budget voor eigen omgevingen". Nee, juist omdat WS-I invoering vrij langzaam zal gaan, met name door de trage upgrade-paden, zullen we met minder genoegen moeten nemen. Een lagere grootste gemene deler-functionaliteit en misschien minder aansluiters dan waar de visionaire plaatjes naar toe willen.
Of: "Iedereen die met ons systeem communiceert moet werken met middleware-versie P.Q.R." Juist ja, en de enige taal die alle systemen spreken zonder proprietary agents heet S.O.A.P. en niet CORBA of RMI of COM+ of MQ! Systemen moeten integreren, en de visie die OASIS daarvoor heeft neergezet is met al zijn beperkingen wél correct.
En door de SOAP-visie in kleine en uitvoerbare stappen op te knippen bereiken we langzaamaan de service bus-wereld, zonder ons door luchtspiegelingen tot al te grote, of van de hoofdroute afwijkende, sprongen te laten verleiden.

Erik de Ruijter RI is ICT Architect bij ABN Amro Bank Nederland.