We beschreven de ESB-markt als het codefree koppelen van webservices, met transformaties en procesflow. De logische vervolgvraag is dan: als we in de organisatie tóch al een applicatieserver met IDE hebben, wat zijn dan redenen om gewoon hiermee de hele service-logica te bouwen en niet met een aparte ESB? We kijken naar JBoss als representant van de coded markt, en bespreken aan de hand hiervan de (soms relevante) zelfbouwredenen. JBoss biedt tevens een toolset die net zo'n hybride karakter kan hebben als we bij veel tweesporen-leveranciers zagen...

JBoss jBPM/Server

ESB-tegenpartij of instap-ESB?

In plaats van eerst de tools langs te lopen en daarna de basisvragen te beantwoorden, geven we hier (na basisuitleg) een meer algemeen antwoord. Daarna pas komen de tools aan bod, en dan zien we voor JBoss zowel een hybride karakter als soms een afwijking van de algemene regel.

Maar eerst wat uitspraken over de theorie, en over enige vergelijkingspunten die zowel voor JBoss als voor de ESB's gelden; die punten vinden we dan ook in de tabel in het ESB-verhaal terug.

* Webservices is simpelweg een protocol waarmee middleware, gelijksoortig of disparaat, kan babbelen. De combinatie van HTTP en een XML-berichtformaat is relevant, wat er 'aan de binnenkant' van een provider of consumer zit, speelt niet.
* Zo'n provider of consumer kan worden gebouwd in C# of Java, maar ook worden 'gegenereerd' door parameterisering van een standaardpakket. In het eerste geval noemen we dat coded, en in het tweede geval een codefree ESB.
* De SOAP-standaard is echter nog bepaald niet zo uitgerijpt als zijn voorgangers in de COM+, CORBA en MQ-werelden - speciaal binnen J2EE. Om de ontwikkeling nog enigszins productief te houden bestaan er allerlei Java add-ons, zoals Apache AXIS. Die add-ons, én de nieuwe .NET-extensies, zijn aan het convergeren naar de WS-I standaardenset: Web Services Interoperability. Die dekt onder meer berichten-security en transactionaliteit. Vandaar dat we voor de ESB-tools support van WS-I als vergelijkingspunt meegenomen hebben.
* BPEL, de standaardorchestratie van processen, staat waitstates toe - geparkeerde processen. Dit betekent dat BAM (Business Activity Monitoring) altijd mogelijk is zodra een tool BPEL ondersteunt. De exacte implementatie van de waitstates, en de audit trails van de procesactiviteiten, is vrij; vandaar de grote variëteit aan BAM-implementaties.

Dan nu de gestelde vraag: wanneer kunnen we beter de al aanwezige IDE en applicatieserver gebruiken voor onze service-bus, in plaats van een aparte ESB? Potentiële redenen om dat te doen zijn:

1) Als coded qua bouw weinig duurder is dan codefree. Het lijkt een rare zin, maar de meeste IDE's hebben keurige efficiënte codegenerators voor een SOAP-provider en consumer. Straks een voorbeeld bij JBoss, dat soms niet veel meer arbeid kost dan de parameterisering van een ESB-tool.
2) Als het hosten qua CPU-kracht en licenties goedkoper is dan bij een ESB. In zijn algemeenheid zal dit níet zo zijn, de krachtiger applicatieservers zijn per CPU vaak duurder dan de beperkter ESB's. Maar er zijn zeker uitzonderingen, zoals JBoss zelf met zijn speciale licentiemodel - en ook de .NET-wereld, met de 'gratis' service-runtime versus het prijzige BizTalk.
3) Als de in de IDE aanwezige basisfuncties voor BPEL en transformaties voldoende zijn.

Vanwege punt 2), maar ook wel de generatie-aspecten van 1), is JBoss een aardig prototype van een ESB-tegenpartij.

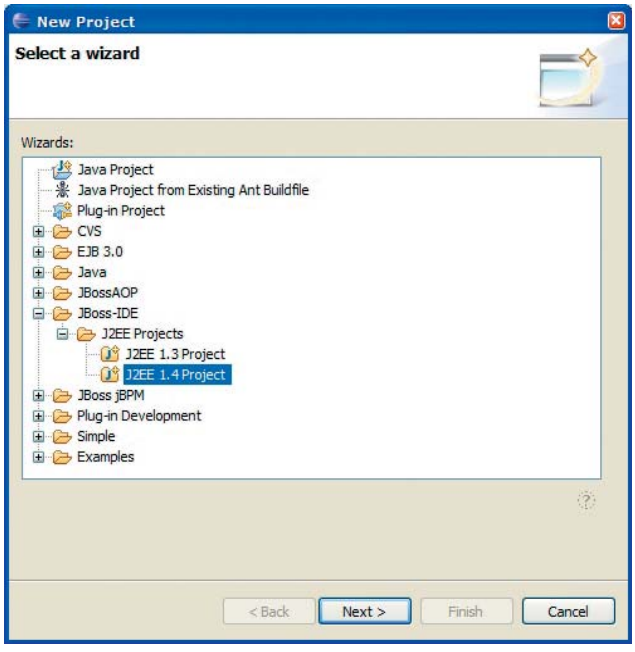
POSITIONERING JBoss, sinds dit voorjaar een zelfstandige divisie van Red Hat, kent een bijzonder Open Source Software-model. De hele productlijn wordt ont-

wikkeld door de eigen staf tezamen met de OSS-wereld, en is ook vrijelijk te downloaden en te gebruiken - ongeacht aantallen CPU's en dergelijke. Alleen voor support, veilige builds, automatische (ASP-) beheerdiensten en gegarandeerde codefixes moet betaald worden; tevens kunnen diensten worden afgenomen in onder meer training en migraties.

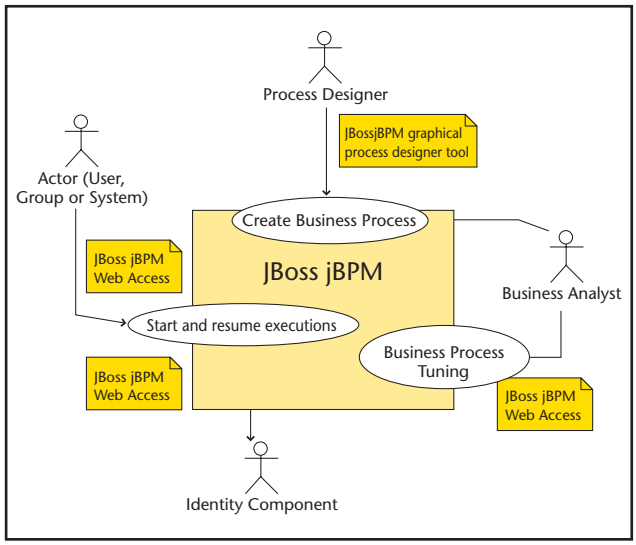
Voor de verplichte basisfuncties van een Enterprise Service Bus biedt JBoss een prima alternatief

Qua producten biedt men een complete J2EE-omgeving, inclusief modulen zoals een portal, transactionaliteit en een MQ-concurrent. De verzamelnaam is JEMS: JBoss Enterprise Middleware System. In dit verhaal bespreken we lang niet de hele JEMS-suite; we kijken primair naar de Web Service Bus-toolset binnen het JBoss platform. We negeren daarbij maar even dat JBoss op de roadmap ook een eigen 'JBoss ESB' heeft die veel doet denken aan pakweg Aqualogic. Recentelijk heeft men daarvoor ook Rosetta, een eigen ontwikkeling van een grote JBoss-klant (een verzekeringsbedrijf in Canada), gekocht.

JBoss webservices Die toolset begint, logischerwijs, met de eigen J2EE-applicatieserver. De huidige 4.0 versie kent volwassen handboeken in PDF-formaat, en een nette download-build op de website. Patches en overige aanvullingen worden alleen in sourcecode op de jboss.org website geplaatst, en voor veilige virusvrije



Figuur 1. JBoss Eclipse wizard.



Figuur 2. Architectuurschema van jBPM.

builds met gegarandeerde support is er zoals al genoemd het betaalde deel van de jboss.com site.

De applicatieserver is een samenstelling van hele brokken populair OSS-materiaal, en alleen waar nodig specifieke JBoss-extensies - onder meer in het Enterprise Java Beans stuk. We vinden onder andere de Apache webserver, de Tomcat web container en een eigen SOAP-stack terug in JBoss. Voor ontwikkeling wordt de al even populaire Eclipse IDE geboden, met waar nodig plugins die kennis hebben van de applicatieserver-extensies. Eentje is er voor de webservices; in de eigen SOAP-tooling kan op basis van een Java service (EJB- of standaard-object) in een paar muisklikken een SOAP provider gemaakt worden. Voorheen (tot en met een patch op application server 4.0) gebruikte men de AXIS 1.3 stack, maar men liep enigszins vast op diens performance, bugvrijheid en nog beperkte compatibiliteit met de WS-I strategie.

Beheer vindt plaats via de eigen vrij kale webconsole, maar er is ook een complete JMX API geïmplementeerd zodat tools als Tivoli en OpenView alle details kunnen besturen.

Het tweede deel van de ESB-tegenpartij binnen JBoss is jBPM. jBPM is in de eigen woorden een 'platform for workflow, BPM, web application pageflow and BPEL'. Dat zit ook echt in die volgorde; de jBPM-versie voor propietary workflow is al zeker twee jaar uit maar de BPEL-variant bestaat nog slechts in alpha-versie.

Het propietary stuk is echt bedoeld voor Java-programmeurs. Binnen een set schermen (JSP) kan workflow worden toegevoegd, op basis van standaardzaken zoals werklijsten. Ook kunnen Java services gemakkelijk in een orchestratie aan elkaar geknoopt worden, waarbij de data langs routing/omzetfilters zoals XSLT en XPATH geleid kan worden. Het gaat deels met point-and-click

in een Eclipse-designer, maar kennis van wat er onder de Java-motorkap gebeurt is essentieel; bovendien praten we in deze aanpak niet over webservices.

Het nieuwe BPEL-stuk werkt juist vanuit die andere, XML-gerichte, optiek. De ontwikkelaar dient niet in Java te denken maar in XML-stromen en in webservices. JBoss geeft overigens wel toe dat ook de finale jBPM-BPEL versie nog niet zo rijp is als bijvoorbeeld een WebLogic Integration. Voor BAM-functionaliteit bijvoorbeeld zijn alle basisdata, dus waitstate-tabellen en logs, aanwezig maar de rapportage zouden we helemaal zelf moeten gaan opzetten.

TEGENPARTIJ OF INSTAPVERSIE? Voor de verplichte basisfuncties van een Enterprise Service Bus, het SOAP-stuk, biedt JBoss een prima alternatief. Qua programmering kost het meer tijd (doch zelden erg veel), qua runtime-licentie is JBoss vrijwel altijd goedkoper. Voor de optionele maar zeker niet onbelangrijke ESB-extra's is het beeld meer wisselend. Enerzijds doordat JBoss jBPM minder diep gaat dan de routering, processflow en BAM van zwaargewicht ESB's (en zwaargewicht coded tools). Anderzijds omdat ook jBPM, met de BPEL-module, ook een stuk codefree werken biedt of althans werken-in-een-XML-wereld. JBoss werkt met jBPM ook al deels toe naar het toekomstige JBoss ESB-product.

Dit beeld zien we ook terug bij de JBoss-beoordelingen onder in de vergelijkingstabel van het voorafgaande ESB-artikel. Functioneel komt het een eind in de buurt van de codefree tools, en de coding-aanpak is natuurlijk een belangrijk onderscheid maar eist voor ESB-achtige zaken ook weer niet zóveel handwerk.

Waarmee we aan de hand van JBoss toch een duidelijke casus hadden voor de, terecht gestelde, tegenvraag als een ESB overwegen wordt: als we in de organisatie toch al een applicatieserver met IDE hebben, dan kunnen er goede redenen zijn om gewoon hiermee de hele service-logica te bouwen en niet met een aparte ESB. De keuze, met alle voors en tegens, dient individueel gemaakt te worden.

Erik de Ruijter RI

PATCHES Patches PATCHES Patches PATCHES Patches PATCHES

Artikelen over onderwerpen als software-ontwikkeling, Java, UML, eXteme Programming en nog veel meer vindt u in het Online Archief van Array Publications. Vaktijdschriften als Software Release, Java Magazine, Database Magazine en ons Oracle vakblad Optimize hebben hun artikelenarchief online gezet. Dankzij de heldere zoekstructuur vindt u snel wat u zoekt op www.release.nl.

Cordys introduceert nieuw Composite Application Framework

Cordys, leverancier van SOA-gebaseerde software, lanceert vandaag de nieuwste versie van zijn Composite Application Framework (CAF). Dit enterprise-platform is volledig gebouwd op basis van SOA-principes. Onderdeel van het nieuwe Cordys-platform is Web Workplace. Deze browsergebaseerde interface vereenvoudigt de creatie van samengestelde applicaties die aan specifieke bedrijfseisen moeten voldoen. Daarnaast heeft Cordys de Business Activity Monitoring (BAM) binnen Web Workplace verbeterd. Dit zorgt voor een diepere en bredere informatievoorziening binnen het gehele bedrijf.

Bovendien omvat het platform een verbeterd Business Process Management System (BPMS) dat bedrijven tools biedt om bedrijfsprocessen grafisch te ontwerpen, te beheren, aan te passen en te executeren in een codevrije omgeving. Cordys introduceert het nieuwe platform als één product waarbij vijf architecturale niveaus, één centraal beheerpunt en een eenmalige installatie zijn gecombineerd. Het platform biedt een volledig geïntegreerd en open SOA-gebaseerde bedrijfsarchitectuur:

* *Niveau 1 - Gebruikerservaring* De browsergebaseerde cliënt bevat Web Workplace, de AJAX-toolkit en Cordys XForms. Deze ondersteunen de opzet van samengestelde applicaties voor gegevens, diensten, applicatiemodules en andere bronnen om aan specifieke bedrijfseisen te voldoen.
* *Niveau 2 - Business Process Management Suite (BPMS)* Een visuele en intuïtieve BPMS waarmee beheerders bedrijfsprocessen kunnen ontwerpen, beheren, uitvoeren en aanpassen. Bestaande diensten op de Enterprise Service Bus (ESB) worden ingezet bij de implementatie, zodat het niet nodig is om nieuwe software te ontwikkelen of applicaties handmatig te integreren.
* *Niveau 3 - Integratie* Een op Webservices gebaseerde ESB met krachtige integratie, messaging, content-gebaseerde routering en de mogelijkheid om gegevens te transformeren.
* *Niveau 4 - Business Logic* Een omgeving waarin het mogelijk is om snel samengestelde applicaties te ontwikkelen. De omgeving is aangedreven door WS-AppServer. Hierdoor kunnen systeemontwikkelaars snel en efficiënt hoogwaardige transactiesystemen vervaardigen.
* *Niveau 5 - Gegevenstoegang* Een verfijnd Master Data Management-raamwerk (MDM) dat efficiënt gegevens synchroniseert binnen verschillende applicaties en tussen databases.

Het Cordys-platform is gebouwd vanuit een homogene XML-architectuur die doordringt in elke platformlaag. Dit vereenvoudigt integratieprojecten en optimaliseert het gebruik van bestaande IT-investeringen. Cordys levert het technologieplatform als onderdeel van zijn complete SOA Box-oplossing. Dit omvat het Composite Application Framework en Cordys@Work, een totaal van richtlijnen, blauwdrukken, templates en best practices. Deze zijn belangrijk voor de succesvolle ontwikkeling en implementatie van SOA-gebaseerde Cordys-oplossingen.