thema

Standaards kunnen een markt opnieuw definiëren; denk maar eens aan de impact van SQL op de relationele databasemarkt. Standaards kunnen ook nieuwe markten creëren - zonder HTML en HTTP bijvoorbeeld zou er geen World Wide Web zijn. Mijn stelling in dit artikel is dat webservices en Java 2 Enterprise Edition (J2EE) een vergelijkbare dramatische impact hebben op applicatie integratie – zij zullen de industrie vooruit helpen van stap-voor-stap integratie oplossingen die pas achteraf ontwikkeld worden (integratie "in het klein") naar standaard applicatie containers die a priori kunnen integreren (integratie "in het groot").

Java en de lijm voor integratie

De toegevoegde waarde van J2EE

Onder applicatie integratie versta ik niet alleen Enterprise Application Integration (EAI), die vooral binnen een intranet achter de firewall wordt toegepast. Ook business-to-business integratie (B2B), waar de applicaties van een bedrijf direct contact maken met de applicaties van een business partner over het Internet (of virtual private network).

In feite zijn EAI en B2B al aan het samensmelten. Individuele business units beheren in toenemende mate hun eigen IT infrastructuur en applicaties. Dat heeft erin geresulteerd, dat webtechnologieën nu breed op het intranet gebruikt worden om het personeel van de ene unit toegang te geven tot de gegevens en diensten van de andere unit.

Zoals het web intranet en internet verenigt, zullen XML en webservices onontkoombaar worden voor zowel EAI als B2B.

1 Webservices zijn het meest eenvoudig te definiëren indien vergeleken met Internet: dat gaat om het verbinden van desktop browsers met documenten en applicaties waar dan ook op een groot netwerk. Webservices breiden dat platform (HTTP, SSL enzovoorts) verder uit voor communicatie van applicaties onderling. Anders gezegd: nieuwe applicaties zijn in toenemende mate al out of the box "web ready", ontworpen om aan de front-end een webbrowser te ondersteunen. Toekomstige applicaties zullen standaard *web integration ready* zijn, gereed om in te pluggen in de "service koppelnet" dat de basis zal worden voor integratie in het groot.

BUSINESS CASE Alleen technologie is nooit voldoende om een industrie te doen transformeren; er moet ook een overtuigende business case zijn. Vandaag de dag zijn grote bedrijven afhankelijk van tienduizenden applicaties. De meeste van deze applicaties opereren in silo's, en kunnen alleen contact leggen indien er een nauwe verwantschap is. Tegelijkertijd dwingt de scherpe concurrentie bedrijven om zich te specialiseren in datgene waarin ze beter zijn dan anderen. Maar terwijl bedrijven diversificeren en activiteiten gaan outsourcen, worden ze gedwongen om nog sterker te integreren met hun supply chains en distributiekanalen.

Integratie achteraf is een dermate groot pijnpunt dat sommige leveranciers nu beweren dat het enige tegengif nog is om iedere applicatie van een enkele aanbieder af te nemen zodat de applicaties al op voorhand geïntegreerd zijn. Beschouw dit echter maar als de slechtst mogelijke oplossing. In de eerste plaats toont de ervaring totnogtoe dat de belofte van integratie vooraf meer marketing hype dan realiteit is. Ten tweede, en nog veel belangrijker: de gedachte dat welke IT-intensieve business dan ook zou kunnen rekenen op een enkele applicatie provider is absurd. Want hoe zit het dan met verticalisering? Geen enkele leverancier heeft expertise in alle sectoren van de industrie. Hoe zit het met legacy? Hoe zit het met competitieve differentiatie met bijvoorbeeld software-uit-eigen-huis? Een IT-intensieve business zal nog minder snel al zijn applicaties van één leve-

rancier afnemen dan dat Boeing of Airbus alle componenten die ze nodig hebben voor een vliegtuig (zoals banden, stoelen, computers of boordapparatuur) bij één fabrikant zouden bestellen.

COMPLEXITEIT VAN INTEGRATIE Het beheersen van de complexiteit en de kosten van integratie zou wel eens de grootste behoefte van de hedendaagse IT organisatie kunnen zijn, en de trends wijzen erop dat het alleen maar erger wordt. Helaas is er geen panacee. Integratie zal inherent een moeilijk probleem blijven. In mijn ervaring is het ontbreken van een applicatie architectuur op basis van services de meest typerende en problematische barrière voor applicatie integratie. In veel applicaties is de business logica vermengd met de presentatielogica. Het integreren van dergelijke non-modulaire applicaties vereist over het algemeen een dure reorganisatie van de code om de herbruikbare business logica services (zoals webservices) te schiften van de presentatie logica.

Vanwege het kostbaar herschrijven van de code, blijft integratie van non-modulaire applicaties vaak beperkt tot data sharing (het opnieuw implementeren van de SQL calls in plaats van het hergebruik van de business service) of screen scraping (hergebruik van de presentatie logica als een business service). Over het algemeen refereert data sharing aan het hergebruik van soortgelijke schema's en tabellen over meerdere applicaties - een goede en veel toegepaste praktijk. Toch betekent data sharing voor applicatie integratie in de praktijk dat het datamodel van bijvoorbeeld een Enterprise Resource Planning (ERP) applicatie wordt blootgelegd voor een Customer Relationship Management (CRM) systeem. Dit schendt de bestaande praktijk van modulair programmeren, verhoogt de complexiteit van de applicatie (aangezien men opnieuw SQL queries en updates moet verkrijgen over ongelijksoortige applicaties) en kwetsbaarheid (aangezien schema-veranderingen binnen het ERP systeem de CRM applicatie kunnen afbreken). Screen scraping, aan de andere kant, doet aan hergebruik van de business logica, maar is tevens kwetsbaar omdat veranderingen in de presentatie de remote services kunnen afbreken. Screen scraping vereist vaak langdradig en saai programmeren. Zelfs voor fraai ontworpen applicaties blijft integratie moeilijk.

2 Terwijl de term "services-based architecture" relatief nieuw is, zijn de concepten al ruim twee decennia onderdeel van de praktijk van het modulair programmeren. Toch ontberen veel nieuwe webapplicaties op gelijke wijze een services-gebaseerde architectuur: denk maar aan de vele PERL of ColdFusion scripts die samengaan met een HTML paginaconstructie of SQL processing. J2EE stimuleert grotere modulariteit met SP, JavaBeans, en EJB's voor respectievelijk abstracte pagina constructie, presentatie logica en business logica.

SEMANTIEK Zo wordt met het woord 'klant' in de ene applicatie een business partner bedoeld, terwijl met ditzelfde woord in de andere applicatie een eindgebruiker wordt aangeduid. Zelfs als de klant van de ene applicatie ook semantisch gelijk is aan de klant van een andere applicatie, is er toch vaak een impedance mismatch tussen de respectievelijke representaties - object versus relationeel, Java versus COBOL, ASCII versus EPS-

> Beschouw integratie vooraf maar als de slechtst mogelijke oplossing

IDIC, tweeregelige versus drieregelige adressen enzovoort. Impedance mismatches kunnen ook voorkomen in de applicatiestructuur; synchrone systemen zoals moderne webapplicaties gedragen zich verschillend van asynchrone (message-based) systemen, en beiden verschillen weer van batchsystemen. Een ander voorbeeld van impedance mismatch is het verschil in de tijdigheid van data tussen een OLTP-applicatie en een beslissingsondersteunende applicatie.

Ondanks de marketing hype kunnen ook XML en webservices de impedance mismatch tussen representaties niet oplossen. XML definieert uitsluitend een gangbaar 'alfabet' voor het construeren van woorden en documenten. Voor het overige is het nog steeds aan de respectieve bedrijven en industrieën om de vocabulaires te definiëren (het mappen van de XML syntax naar business semantiek) waardoor 'conversaties' kunnen plaatsvinden. Sommige van deze vocabulaires zullen "top down" gedefinieerd worden door industriestandaard-organisaties en consortia. Andere vocabulaires zullen, net als 'echte' spreektalen, "bottom up" groeien, wanneer individuele bedrijven of kleine groepen bedrijven de webservices interfaces zelf specificeren.

XML-gebaseerde EAI staat voor dezelfde uitdagingen, hoewel binnen de grenzen van de organisatie. Terwijl tools die van de ene naar de andere vocabulaire mappen de last kunnen verlichten, het definiëren en uitvoeren van vocabulaires is de grootste barrière voor het bereiken van integratie in het groot. De infrastructuur hiervoor zoals webservices en J2EE adapters is relatief gezien eenvoudig.

INTEGRATIE STANDAARDS Applicatie architectuur en impedance mismatch zijn inherente uitdagingen voor integratie. Deze uitdagingen zijn onafhankelijk van de integratietechnologieën. Maar ook het gebrek aan integratie-standaards frustreert de IT-organisatie. Zonder passende standaards zal integratie in het klein

duur blijven, en integratie in het groot onmogelijk. Momenteel zien we, in plaats van samenbindende standaards, verschillende kleine vendors die proprietary integratie platforms aanbieden, hetgeen leidt tot een versnipperde markt die zijn eigen investeringen niet kan beschermen. De industrie zou deze proprietary technologieën collectief behoren te vervangen door standaards.

PROPRIETARY PROTOCOLLEN De lakmoesproef voor een proprietary protocol is wanneer dezelfde platform software aan beide kanten van het netwerk kan draaien. De analogie met het web is treffend - zonder gestandaardiseerde HTML en HTTP, zou de volledige interconnectie van heterogene webbrowsers en heterogene webservers nooit hebben kunnen plaatsvinden. Proprietary protocollen zullen eenvoudigweg niet werken voor de mate van vereiste integratie op het web. Sommige van de vroege B2B toepassingen konden slechts gedeeltelijk worden opgezet door hun afhankelijkheid van proprietary protocollen.

> Ook XML en webservices de impedance mismatch tussen representaties niet oplossen

De proliferatie van webintegratiestandaards zal ook de kostenbarrière om met B2B toepassingen te beginnen verlagen. Eerdere, proprietary B2B protocollen zoals Electronic Data Interchange (EDI) zijn eenvoudigweg te duur om nog voor algemene toepassing in aanmerking te komen. XML mag dan inderdaad een webstandaard zijn, de conventies voor het doorgeven van een XML document zijn soms nog steeds zeer proprietary. Om die reden is de opkomende reeks XML- en webservices standaards zoals SOAP, WSDL, UDDI en het Electronic Business XML Initiative (ebXML) zo essentieel. Zonder zulke standaards zullen gebruikers nooit met integratietoepassingen kunnen 'mixen en matchen' zoals ze dat met eerdere webtechnologieën deden.

PROPRIETARY ADAPTERS Adapters lossen het "eindsprint"-probleem van integratie op: zij voorzien in het mappen tussen een nieuwere technologie (zoals webservices en Java/J2EE) en een "legacy" technologie (zoals COBOL/CICS).

3 Aan het gebruik van de term "legacy" hoeft niemand aanstoot te nemen. Het is het doel van alle nieuwe software om onderdeel te worden van de legacy, en in de kern van de zaak zijn alle productie applicaties deel van de legacy.

Zelfs met de opkomst van XML en webservices blijven adapters essentieel, aangezien van de huidige legacy maar een klein deel zal worden uitgebreid om direct webservices te gaan ondersteunen. Bovendien, adapters zijn beter dan webservices in staat tot nauwere integratie met de legacy, bijvoorbeeld door het mogelijk te maken dat een enkele transactie of security context gedeeld wordt door de nieuwe en oude applicaties.

Zonder een standaardmodel voor adapters, is het bijna onmogelijk om kritieke massa te krijgen. Denk eens aan de impact die de standaardisering van een gewone database adapter - Java Database Connection (JDBC) - heeft gehad op het succes van Java. De J2EE Connector Architecture (CA) generaliseert JDBC om een universeel model voor Java adapters te definiëren. J2EE CA elimineert de n2 kosten van elke integratie-vendor die zijn eigen proprietary adapter moet '*one off*'*-en* voor ieder afzonderlijk legacy systeem. Nog belangrijker is het dat J2EE CA de programmeer-investeringen in integratie oplossingen op dezelfde wijze beschermt zoals JDBC de investeringen in database applicaties beschermt.

PROPRIETARY API'S EN INFRASTRUCTUUR De integratie server bevat de protocollen en adapters die samen de "eindpunten" van een integratie oplossing vormen. Integratie is uiteraard gecompliceerder dan eenvoudigweg mappen van het ene eindpunt naar het andere. Om die reden zijn integratieservers dan ook uitgerust met 'sophisticated' API's voor het ontwikkelen van de "lijm", die noodzakelijk is om deze eindpunten aan elkaar vast te knopen. Helaas zijn de meeste API's op de huidige integratieservers proprietary. Aangezien de meeste kosten van een integratie-oplossing gaan zitten in het ontwikkelen van deze 'glue programming', is het grootste risico voor integratieprojecten een langdurige afhankelijkheid van proprietary API's. Diezelfde proprietary API's verhinderen ISV's en systems integrators dat zij hun investering in integratie kunnen hergebruiken over verschillende integratieservers.

Het volgende is een opsomming van essentiële 'integratieserver lijm' die klaar is voor standaardisering:

- *Messaging: het meest fundamentele gedeelte van de integratieserver*

  De Java Messaging Service (JMS) standaardiseert asynchrone communicatie voor Java applicaties met een API die ondersteunt het spectrum aan messaging-behoeften: bijvoorbeeld store/forward versus publish/subscribe; hub-and-spoke versus peer-to-peer; unicast versus multicast. De ontwikkelen van Java messaging applicaties is nu al aan het convergeren naar JMS zoals al eerder database programmeren naar JDBC.

- *Messaging en webservice componenten*
  Server-side component models reduceren de complexiteit van het ontwerpen enorm, door beslissingen over de infrastructuur te verschuiven van de business logica naar deployment descriptors. Op deze manier kan de business logica, die definieert wat gedaan moet worden voor een bepaalde JMS message of webservice request, fors gesimplificeerd worden met Enterprise JavaBeans (EJB's). J2EE message-driven EJB's maken het programmeren van de 'trechters' voor JMS messages eenvoudig.

- *XML verwerking en transformatie*
  XML en webservices zijn zo nauw verbonden geraakt met J2EE services, dat zeer efficiënte XML-standaard parsing en transformatietechnologie ingebed moet worden op het platform. Zowel tools die het mappen van het ene XML schema naar het andere vereenvoudigen, evenals voorgedefinieerde XML transformations (zoals mappings naar EDI), behoren op gelijke wijze op de integratieserver worden gezet.

- *Transacties*
  Transacties garanderen levering via een two-phase commit protocol. Zonder levering met transactiegarantie, blijft er altijd een kwetsbare plek bestaan waarbinnen fouten een message kunnen dequeuen wordt, zonder dat de corresponderende databaseacties of andere noodzakelijke applicatieverwerkingen voltooid zijn. De J2EE standaard hiervoor is de Java Transaction API (JTA), een randvoorwaarde voor J2EE CA.

- *Beveiliging*
  Integratieservers behoren alle netwerkcommunicatie te beschermen (privacy, voorkomen van vertraging) evenals het op betrouwbare wijze identificeren van participerende applicaties (authentisering) en verzekeren dat elke participant alleen toegang kan verkrijgen tot de juiste diensten (autorisatie). In J2EE regelt de Java Authentication and Authorization Service (JAAS) de beveiliging van het platform en de ondersteunende infrastructuur zoals SSL.

- *Naamgeving*
  'Glue programming' hangt af van naamgeving en directory services. De Java Naming en Directory Interface (JNDI) is de Java API voor het manipuleren van locale namen (in de adresruimte) en voor toegang tot de inhoud van een LDAP-repository. Deze infrastructuur strekt zich nu ook uit tot webservices. In bepaalde producten wordt automatisch een dunne Java client consumer gegenereerd en gepubliceerd, beiden op een web URL en naar een UDDI directory, om eenvoudige toegang tot webservices te accommoderen.

- *Business Process Management (BPM)*
  BPM refereert aan hogere scriptingtalen en tools voor het samenstellen van workflows. In sommige producten zoals WebLogic Integration van BEA kunnen workflows grafisch geprogrammeerd worden door *drag and drop* van de standaard J2EE bouwstenen - EJB's, JMS messages, webservices enzovoort.

- *Performance, schaalbaarheid, en betrouwbaarheid*
  Integratieservers zijn net zo essentieel voor de organisatie als de applicaties die ze onderling verbinden. Het uitleveren van de clustertechnologieën die transparant zorgdragen voor quality of service - automatische replicatie, load balancing, failover, caching, sessiebeveiliging, bepalen van prioriteiten en routing (inclusief data- en contentafhankelijke routing) - is zowel erg moeilijk als erg duur. Zo duur dat de kosten van het ontwikkelen van business-critical platform software (e.g., besturingssystemen, databases en applicatieservers) over zeer lange termijn worden afgeschreven, terwijl de markt zich consolideert rond een kleine groep winnaars.

- *Monitoring en management*
  Management consoles behoren een duidelijk, operationeel overzicht te geven van de interacties op de integratieserver en de verbindingen met de verschillende applicaties. Bovendien behoort de integratieserver uitgebreide instrumentatie te bevatten zodat het verifiëren van een regel business view (orders, dollars et cetera) zo direct mogelijk kan plaatsvinden.

ESSENTIEEL Zonder twijfel is de hiervoor beschreven lijst onvolledig. Zo heb ik JDBC, dat nodig is voor data sharing, onbesproken gelaten. Maar de lijst illustreert wel treffend, dat de toekomst van Java-gebaseerde integratieoplossingen gebouwd zal worden boven op het J2EE platform! Totnogtoe worden J2EE-standaard applicatieservers vooral gebruikt om nieuwe applicaties te huisvesten (die vervolgens toegang hebben tot de legacy systemen), terwijl de integratie van bestaande applicaties wordt overgelaten aan meer proprietary integratieservers. Maar bedenk wel, dat J2EE adapters direct afhankelijk zijn van JTA, JAAS, JNDI, en clustering; en indirect van JMS (voor het asynchroon aanroepen), de EJB container (voor message driven EJB's), XML services enzovoort. Op dezelfde wijze zijn webservices direct afhankelijk van HTTP servlets, XML verwerking en clus-

4 Non-repudiation, de mogelijkheid de zender van een request achteraf vast te stellen teneinde bijvoorbeeld de authenticiteit van een order te bepalen, is een terrein waarvoor de standaards nog in ontwikkeling zijn.

tering, en indirect van EJB en JMS. Tel dit alles bij elkaar op, en de conclusie moet zijn dat vrijwel de gehele Java 2 Enterprise Edition essentieel is voor integratie, gegeven dat men gelooft in de noodzaak van standaardisering van adapters en webservices.

Toegevoegde waarde Het voorgaande mag geen verrassing zijn. Immers, waarom zou integratie programmeren fundamenteel verschillen van applicatie programmeren? Waarom zou je dure infrastructuursoftware twee keer bouwen of twee keer kopen? Uiteraard heeft de J2EE gemeenschap hard gewerkt aan uitbreiding van J2EE zodat die nog beter geschikt wordt voor integratie. Het resultaat is, dat de industrie in hoog tempo samentrekt rondom het Java/J2EE platform en het .Net alternatief van Microsoft. Beide platforms dragen een aansprekende en breed gedeelde visie over webservices uit, een visie die reeds bewezen wordt in directe interoperabiliteit tests. Op een aantal punten biedt Java echter toegevoegde waarde:

> Er is helaas is geen panacee; integratie zal inherent een moeilijk probleem blijven

- Java/J2EE maakt development en deployment mogelijk op vrijwel alle mainstream computing platforms (deze hoge 'dekkingsgraad' is uiteraard essentieel voor integratie).
- Java/J2EE API's zijn standaards en zijn een weerslag van de bijdragen van honderden bedrijven en organisaties binnen de Java gemeenschap.
- Webservices *bindings* kunnen transparant gegenereerd worden voor bestaande Java/J2EE applicaties (programmeurs gebruiken wat ze al weten), hetgeen de sleutel is voor integratie in het groot.
- De J2EE CA is thans commercieel levensvatbaar.

Software leveranciers zoals PeopleSoft, Siebel, SAP en BEA werken steeds nauwer samen om standaard J2EE adapters te leveren voor hun enterprise applicaties. In het algemeen is er samenwerking tussen alle grote Java systems-leveranciers als BEA, Sun, IBM, Bull, HP, Compaq, NEC, Nokia, Oracle en Unisys naar de verfijning van J2EE als platform voor integratie. Dit alles geeft aanleiding te veronderstellen dat J2EE-gebaseerde integratie de benodigde 'kritieke massa' heeft.

Voorbereiden Natuurlijk heeft dit nieuwe standaard integratieplatform tijd nodig om volwassen te worden. In het bijzonder webservices standaards zijn volop in ontwikkeling: non-repudiation, gegarandeerde aanlevering en compenserende acties zijn de drie kerngebieden voor standaardisering. J2EE-compliant adapters worden pas sinds kort gebouwd. Bovendien moesten leveranciers de J2EE CA nog uitbreiden met essentiële mogelijkheden als tweerichtingsverkeer, ondersteuning voor asynchrone verwerking (via de Java Message Service) en metadata repository's.

Hoe kunnen organisaties zich nu het beste voorbereiden op gestandaardiseerde integratie? Door de integratie-uitdaging zowel tactisch als strategisch tegemoet te treden: tactisch, door een mix samen te stellen van de best passende standaard en proprietary technologieën; strategisch, door in te zetten op opkomende standaards. In het bijzonder raad ik aan langdurige of veelomvattende commitments aan proprietary integratieframeworks te vermijden, in ieder geval totdat de transformerende impact van standaards de eventuele verliezers zal hebben geschift van de winnaars.

Conclusie Tenslotte nog een paar opmerkingen voor wie nog steeds twijfels heeft over de toepasbaarheid van J2EE. Recent zijn we getuige geweest van een soortgelijke transformatie op het terrein van de applicatieserver platforms. Vier jaar geleden waren er letterlijk tientallen applicatieservers op de markt, die allerlei proprietary programmeermodellen promootten. We zeiden toen dat J2EE een snelle consolidatie in deze markt zou bespoedigen, en dat degenen die de impact van J2EE negeerden hun tijd en geld zouden verspillen. De meerderheid van die proprietary platforms is nu verdwenen. De overblijvers hebben hun producten over het algemeen opnieuw ontworpen rond J2EE of het Windows-centrische .Net alternatief van Microsoft, dat sterk in opkomst is. Voor webgebaseerde applicatie-integratie is de geschiedenis zich nu aan het herhalen. De dwingende behoefte aan standaardisering in webservices en Java API's zal deze markt voortstuwen. Zoals met alle technologische transformaties, zijn er nu interessante mogelijkheden om een voorsprong op de concurrentie te behalen, in het bijzonder voor ISV's en systems integrators die zich richten op integratie oplossingen.

Scott Dietzen

Scott Dietzen is als chief technology executive verantwoordelijk voor BEA's E-Commerce Server productlijn en woordvoerder voor BEA's development programma rond draadloze Java-toepassingen.