Ook in het webtijdperk blijven er applicaties die niet voor browsers bedoeld zijn. We praten dan vooral over B2B, het communiceren van computer naar computer. Een vakgebied zo oud als de ICT, dat tóch behoorlijk overhoop gegooid wordt, juist dankzij Internet-technieken. We zien hoe met platforms als J2EE en .NET de tools voor moderne B2B communicatie gewoon worden bijgeleverd, en ook waar we bij de programmering allemaal op moeten letten. Big B2B business ligt dichter binnen handbereik dan men soms denkt...

Big business

B2B applicaties ontwerpen

In het werk van ons slag mensen, ontwikkelarchitecten, zijn zuivere definities cruciaal. Vandaar dat we hiermee beginnen, al moet vanwege de ruimte soms een compacte formulering gebruikt worden. B2B, ons focusgebied, definiëren we als 'praten met andere onbemenste computers'. De eindapplicatie kan misschien best een vorm van user interface hebben, denk aan een PIN betaalautomaat, maar de voor de B2B-bouwers relevante scope heeft dat niet; het is computer A (kassasysteem) tegen computer B (banksystemen). Een zuiverder naam zou misschien host-to-host zijn, maar 'B2C' (consumer) en 'B2E' (employee) worden als termen nu eenmaal met dialogen geassocieerd en B2B veelal niet.

Definities In deze indeling zouden we bedrijfsapplicaties grofweg in drie groepen kunnen opsplitsen: dialoog (B2C/B2E), B2B en batchverwerking. Waarbij de laatste geen dialoog kent, maar binnen de bedrijfsmuren plaatsvindt, bijvoorbeeld in het geval van maandrapportage of datawarehouse-transformaties. En we zullen ons vooral richten op B2B met moderne platforms zoals J2EE en .NET. Klassieke standaards zoals EDI (Electronic Data Interchange) en VAN (Value Added Network) bestonden natuurlijk al in de Cobol- en Visual Basic-tijdperken; in moderne B2B-technieken zoals ebXML en Internet VPN's zien we ze terugkomen.

Een volgende groep definities vinden we binnen het B2B gebeuren. De communicatie van host naar host kent namelijk zelf weer twee hoofdtypen, te weten request/response en one-way.

One-way, ook wel Event-Driven Architecture (EDA) of 'asynchroon verkeer' geheten, gaat uit van zelfstandige berichten. Host X heeft iets te vertellen aan host Y, en wil alleen enige garantie dát Y deze informatie gaat oppakken - geen directe synchrone verwerking. Voorbeelden zijn e-mail en inkooporders. Uiteindelijk zal host Y wel weer een, geheel andere, message terugzenden met de laatste status. Dat kán binnen een seconde gebeuren maar kan ook minuten duren of zelfs tot de volgende dag, bij one-way transport moet dat allemaal kunnen.

Request/response heet dan vanzelfsprekend 'synchroon verkeer' en vormt de basis van de welbekende SOA (Service Oriented Architecture). En die techniek kennen we, net als alle tot op heden genoemde B2B-concepten, ook volop tussen interne applicaties; er zijn de nodige parallellen tussen interne integratie oftewel EAI (Enterprise Application Integration) en externe B2B-integratie.

Onder andere vanwege de opkomst van webservices en Internet-communicatie neigt B2B verkeer meer en maar request/response verkeer, en daarom kijken we ook nog even naar de programmering daarvan. Hoewel men dat niet direct zou verwachten is het MVC-model (Model View Controller) dat een webapplicatie, in .NET of J2EE, domineert, hier gewoon toepasbaar. De Modelkant is zelfs geheel dezelfde, namelijk de datastructuren. De View wordt dan de afhandelaar van uitgaande request-berichten, en de Controller handelt de inkomende verzoeken af. Responses zijn gewoon inkomende en uitgaande berichten: dus bijvoorbeeld een inkomende SOAP call, die weer leidt tot een tegenbericht via de View.

Protocolstack Voordat we dieper de eigen B2B-applicaties induiken zullen we nog meer moeten beschrijven over de externe communicatie. Termen als one-way en request/response zijn immers loze kreten als er geen harde protocol-afspraken aan gekoppeld worden. En bij protocollen hoort soms specifieke mid-

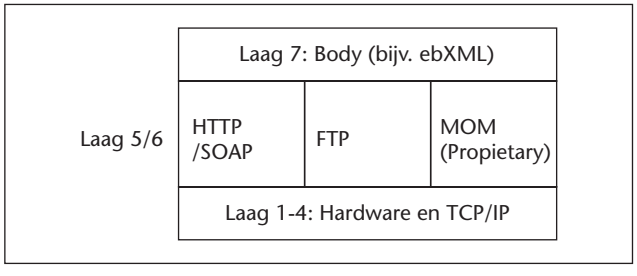
dleware, het is niet anders - en om de OSI stack komen we al helemaal niet heen. We beperken ons echter tot de bovenste lagen van de stack, de applicatieve layers. Figuur 1 geeft hiervan een overzicht.

Als we via open marktstandaard protocollen naar andere organisaties toe willen communiceren is de keuze gelukkig beperkt. Ten eerste is er SOAP over HTTP voor het request/response verkeer. Door Oasis tot standaard verheven, zelfs J2EE en .NET zijn compatibel, dus wie klaagde er over 'onvoldoende verbreid'? Natuurlijk, er blijft nog veel te wensen over. Zo vereist de beveiliging bijvoorbeeld dat aanvullende protocollen gebruikt moeten worden, zoals WS_Security. Dit terrein is wat implementaties betreft nog zodanig incompleet, dat we veelal gewoon TCP/IP encryptie (VPN of HTTPS) moeten gebruiken.

B2B communicatie, veelal over Internet, vereist wel én geen speciale vaardigheden in onze programmatuur

We kunnen kiezen voor eenrichtingsverkeer. In feite is ook SOAP hiervoor inzetbaar, mits we dan 'stoppen na de ACK voor ontvangst'. Zodra een verzonden bericht door de andere host ontvangen is stopt de actie, en wat de andere host dan later zal terugmelden is een apart (asynchroon) retourbericht. Gaat het om grotere bergen informatie - en bij one-way kan dat voorkomen - dan is ook FTP goed inzetbaar. Natuurlijk, ook dat kent forse beperkingen qua security en foutgevoeligheid. Maar die zijn oplosbaar, wederom via IP encryptie of door een geavanceerd FTP-sprekend transfertool af te spreken.

GRENZEN Al die 'extraatjes om écht kwaliteit te bereiken' geven al aan dat de open standaards grenzen kennen. Er zijn dan ook legio gesloten standaards die out-of-the-box hoogwaardige kwaliteit bieden - maar tegen de prijs van dure tooling en lock-in. Voor one-way verkeer kennen we de populairste categorie: MOM oftewel Message Oriented Middleware. IBM leidt deze markt met WebSphere M(essage)Q(ueue), en ver daarachter komen we kleinere spelers tegen zoals Microsoft MQ, Oracle MQ en SonicMQ. Er is allemaal niets mis mee, maar overweeg ze alleen serieus als in een bepaalde B2B-situatie 'standaards onderling vast te leggen zijn' en dus propietary mogen zijn en blijven. Voor request/response verkeer, om volledig te zijn, is er niet écht een populaire B2B standaard. Synchroon B2B verkeer is eigenlijk pas in de mode gekomen dankzij Internet, en werd dan ook meteen goed neergezet.Tot zover de protocollen voor laag 5 en 6 samen, oftewel sessielaag en presentatielaag. In laag 7 van de stack, op de applicatielaag, vinden we dan afspraken over de berichtinhoud. SOAP en FTP (en de MOM-tools) zeggen namelijk alleen wat over de 'envelop', en misschien dat er XML gebruikt zal worden, maar meer ook niet. Die afspraken kunnen ook gewoon van geval tot geval gemaakt worden, maar de markt kent ook pogingen om dieper te standaardiseren. Denk maar eens aan het ebXML committee van Oasis, bemand door oude rotten uit de EDI-wereld. Het is allemaal een potentiële goede aanvulling op de eerder gekozen fundamenten, waaronder SOAP. Dat is ook wel de randvoorwaarde; er is ook een stroming binnen ebXML die de SOAP envelopes wil modificeren maar dat lijkt ons echt een in de markt gepasseerd station, en niet het navolgen waard voor een B2B-implementatie die 'open' wil blijven.

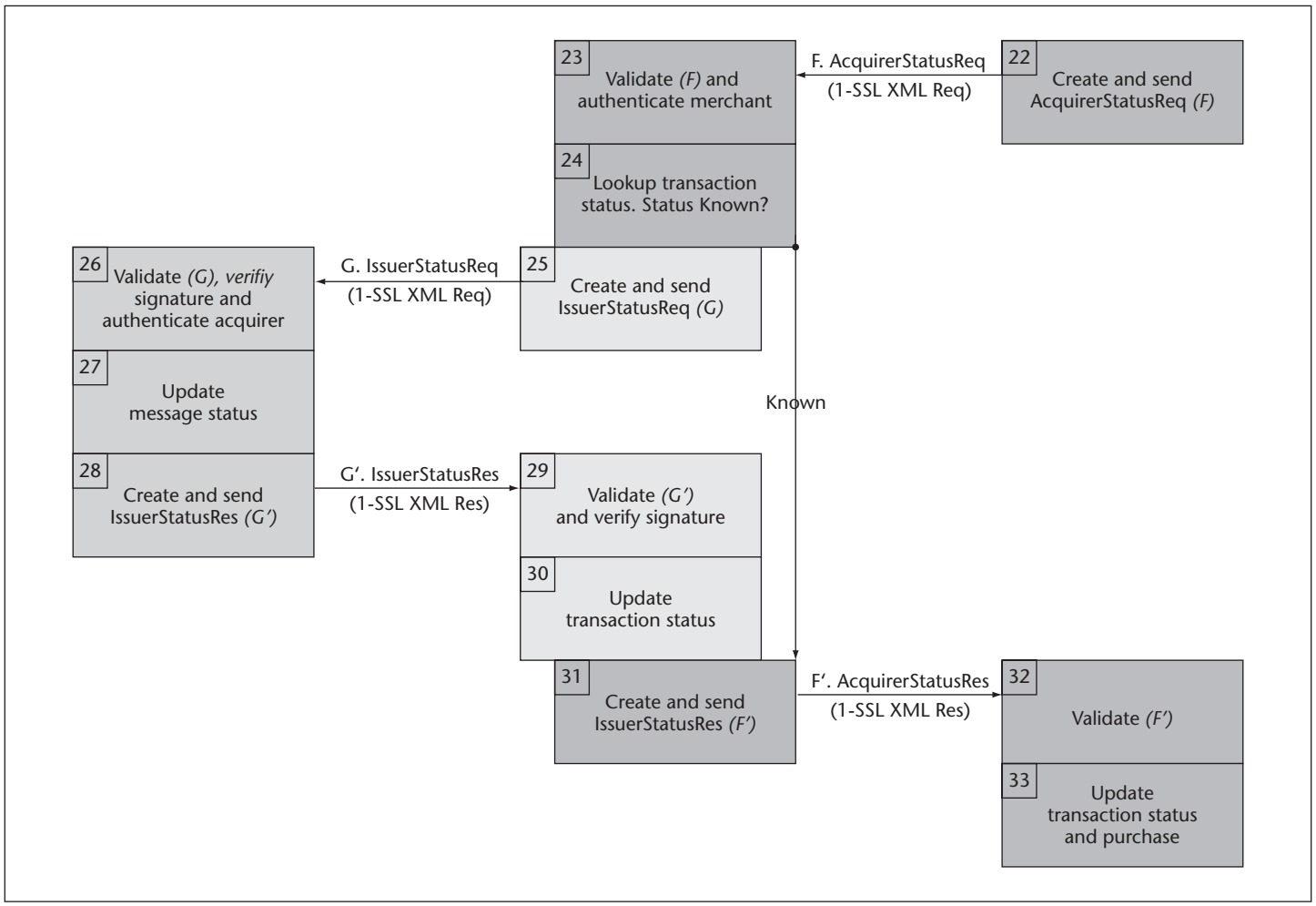


FIGUUR 1. Protocollen gebruikt voor B2B

B2B GATEWAYS Qua communicatietypen en protocollen zijn er de nodige parallellen tussen EAI en B2B, maar we komen nu op een terrein dat vrij uniek is voor B2B applicaties: gateways. Gartner definieert hem als een 'central managing point for external communications and interactions'. En het is niet zozeer een enkele centrale technische instance maar een set functies die we 'in place' moeten hebben voordat we onze systemen zomaar met vreemde hosts laten babbelen. Soms zal het nuttig zijn om de functies ook op één plaats (en één hoofd-URL) binnen het bedrijf te concentreren en soms kan het rustig versnipperd per applicatie of zelfs handelspartner - als de basisarchitectuur maar gevolgd wordt.

Routing, ook wel met als voorvoegsel 'content-based' of 'intelligent', is de verkeersagent-rol. Hier gaat het om het verdelen van messages zodat ze de juiste bestemming bereiken: inkomende berichten, vanaf een centraal (beveiligd) punt naar de achterliggende business applicaties, en uitgaande berichten, vanaf applicaties naar de juiste handelspartner, ook als diens URL gewijzigd is.

Transformatie spreekt bijna voor zichzelf: omzetting van het ene naar het andere dataformaat. Als we tussen



FIGUUR 2. Voorbeeld diagram van B2B ontwerp (web-betalingen)

de bedrijven dataformaten zoals ebXML of EDI-voorlopers inzetten, dan is transformatie onvermijdelijk. Dit soort zwaar gestructureerde data willen we onze interne applicaties en business services veelal niet aandoen, zeker als die services ook nog eens voor een veelvoud aan interne doelen hergebruikt worden. Dataformaatomzetting heet ook wel 'semantische transformatie', en kan onder andere betekenen dat een code 0-1 in applicatie P wordt omgezet in een M-V waarde voor B2B communicatie Q. Het kan ook complexer, bijvoorbeeld bij het afleiden van de regiocode uit de postcode. We moeten echter uitkijken dat de transformatie niet uitgroeit tot echte businesslogica-regels, die horen thuis in goed beheerde business services en niet in de B2B-middleware. Naast deze semantische transformatie is er technische transformatie, bijvoorbeeld tussen karaktersets of de Unix versus Windows varianten van ASCII. Deze laatste kan beter puur aan middleware gedelegeerd worden, om ons niet teveel om te hoeven te bekommeren in het B2B-ontwerp.

Audit trailing is een applicatief concept dat vaak vergeten wordt, maar nogal essentieel is. Als we onze eigen applicaties aan die van partners koppelen, dan zijn verdwenen of verminkte transacties nooit geheel uit te sluiten. Om priemende vingers te vermijden is het belangrijk om gedetailleerd te loggen wat er ons bedrijf binnenkomt en verlaat. Als de 'tegenpartij' dit nu ook doet dan kan duidelijk worden bepaald waar zaken fout gaan: bij de zender, de ontvanger of in de tunnel tussen beide gateways in. Zeker als een bestaand applicatief complex door Business Process Outsourcing 'over meerdere rechtspersonen verdeeld wordt' is het bewaken van de knip erg belangrijk, voor het bewaken van de lieve vrede én van alle SLA's.

De laatste gateway-rol is meer 'een inkopper', omdat hij voor de meeste serieuze integratiecomponenten geldt: *exception handling*. Niet alleen moet de gateway de normale zelf-bewaking hebben, maar omdat hij als poortwachter van onze organisatie fungeert, moet hij ook de boel opvangen wanneer achterliggende services er uit liggen. Net zoals een HTTP-server bij B2C-verkeer dan nog minimaal een nette foutmelding kan tonen, zo zal de gateway binnenkomende calls netjes bouncen (bij request/response) of simpelweg bufferen (bij bepaalde one-way varianten) als ze niet afleverbaar zijn bij de juiste business applicatie. Andersom geldt, dat wanneer

een uitgaande call faalt, de gateway de taak heeft om de returncode netjes terug te geven naar onze eigen applicatie, en zonodig ook zelf te interpreteren.

ONTWIKKELTOOLING Om het big business-beeld compleet te maken kijken we kort naar de tooling. De trend binnen zowel .NET als J2EE is om meer en meer extra's in het basisplatform te stoppen, juist om de bedreiging vanuit de open source applicatieservers zoals Mono en JBoss te limiteren.

> In het werk van ons slag mensen, ontwikkelarchitecten, zijn zuivere definities cruciaal

Voor communicatieprotocollen hebben we qua basis voldoende aan onze toolset. Zowel SOAP-HTTP als FTP zit immers in standaard omgevingen al aan boord. Voor gateway-programmering zijn soms nog extra's nodig, maar die worden door onder meer Microsoft en IBM ook gaandeweg naar de basis verschoven.

Allereerst de (content-based) routing: hiervoor komen we vaak een heel eind door de definities in XML Path language, oftewel XPATH, te plaatsen en dan vanuit Visual Studio of onze Java IDE diens XPATH processor aan te roepen. Wat verder kán helpen is BPEL: Business Process Execution Language for Web Services. BPEL staat grotendeels los van EAI en B2B omdat het gewoon een nuttig tool is voor snelle programmering in een SOA, middels zijn 'orchestration' van onze aparte services. Maar binnen routering kan BPEL nuttig zijn voor het overzichtelijk orchestreren van complexere routes, de 'forks' en 'joins' in een message-pad.

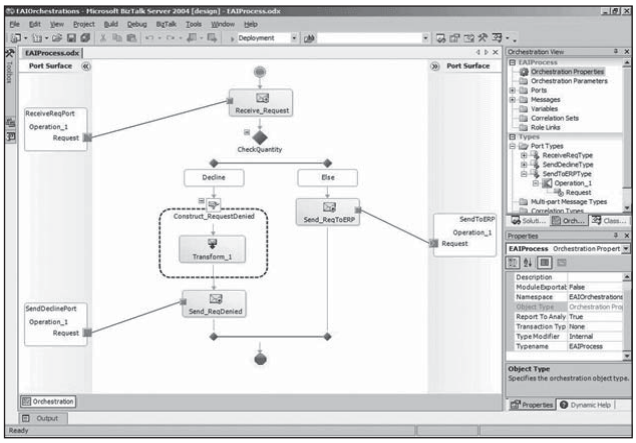
Het tweede protocol betreft de transformatie. Hier is de tooling glashelder: XSLT. Zet alle omzettingen, tussen bijvoorbeeld ebXML en de interne DTD, in een XSL sheet dat op runtime geprocessed moet worden door ons platform. Hier is een goed onderhoudstool met drop-and-drag beheer van het XSL sheet, erg handig.

Voor audit trailing en exception handling tenslotte, bestaan er geen echt algemene standaards. Maar ook hier mogen we van ons basisplatform verwachten dat ze er standaard patterns en functies voor hebben, bijvoorbeeld in IBM's met WebSphere meegeleverde 'Web Services Gateway'.

Maar er zullen ook situaties zijn waar we, vanwege B2B of EAI, extra functies bovenop de basis nodig hebben. Denk bijvoorbeeld aan complexe XSLT editors inclusief business service-aanroep, meerlaagse orchestratie bovenop BPEL, koppeling met workflow en BAM (Business Activity Monitoring).

Dan moet worden gekeken naar *process managers* en *integration brokers*. Bekende namen in de markt zijn onder andere MS Biztalk, IBM WebSphere Business Integration, BEA Weblogic Integration, WebMethods en SeeBeyond. In ons zusterblad LAN Magazine staat winter 2003-2004 een serie besprekingen van deze tools, en die is nog redelijk actueel, ondanks de updates sindsdien.

CONCLUSIES Al met al kunnen we constateren dat B2B communicatie, veelal over Internet, wel én geen speciale vaardigheden in onze programmatuur eist. Wél, in die zin dat we meer dan anders moeten letten op communicatiestijlen en dat we in de applicatiearchitectuur een specifieke 'gatewayfunctie' moeten tussenvoegen. Tegelijk zijn géén specialiteiten vereist, omdat de vereiste programmeertools in hoge mate reeds aanwezig zullen zijn in onze .NET of Java IDE. Wanneer we de verschillen tussen de normale browser-applicaties en B2B stijlen goed in de gaten houden, zijn we gewoon 'in business'!



FIGUUR 3. Ontwerptool van MS BizTalk 2004

Erik de Ruijter RI is werkzaam als ICT-architect bij ABN Amro Bank N.V., Unit Consumer & Commercial Clients Nederland.