Op weg naar een business process enabling-infrastructuur

Van EAI naar BPA (1)

Device-drivers, databasemanagementsystemen, job control language, transactie processing-monitoren: allemaal is het ooit middleware genoemd. Middleware had en heeft tot taak om techniek te verbergen voor de programmeur, zodat die zich kan concentreren op de functionaliteit. In dit tweeluik behandelen de auteurs de huidige generatie middleware: software voor 'Enterprise Application Integration'. Daarbij besteden ze vooral aandacht aan de laatste ontwikkeling daarbinnen, het zogenaamde 'Business Process Automation'. Maar eerst, om door de bomen vooral het bos te kunnen zien, een overzicht van de verschillende categorieën EAI-middleware van dit moment.

De combinatie van Business Process Automation (BPA) binnen de huidige Enterprise Application Integration (EAI) kan méér opleveren dan alleen onafhankelijkheid van de onderliggende techniek. Mits goed toegepast, kan deze EAI-middleware een extra dimensie geven aan het begrip 'onafhankelijkheid'. Middleware zorgt niet alleen voor een scheiding tussen techniek en functionaliteit, het zorgt ook voor een scheiding tussen functionaliteit en bedrijfsprocessen. Met andere woorden: EAI-middleware wordt een middel om bedrijfsprocessen en onderliggende applicaties zich los van elkaar te laten ontwikkelen. Daardoor wordt EAI-middleware een faciliterende (enabling) factor bij het ondersteunen van nieuwe bedrijfsdoelstellingen. Het groei uit tot een zogenoemde Business Process Enabler (BPE).

In dit eerste artikel maken we een verdeling in categorieën van EAI-middleware en behandelen we een aantal aspecten die nauw samenhangen met de inzet van deze middleware. In het volgende deel bespreken we de toepassingsstadia van EAI, waarbij met name het stadium waarin procesondersteunende functionaliteit wordt toegepast van belang is.

Dataneutrale middleware

We onderkennen ruwweg vier categorieën middleware: de dataneutrale, de datamanipulerende, de procesgeoriënteerde en de B2B-middleware. De eerste categorie, dataneutrale middleware, houdt zich niet bezig met wat er gecommuniceerd wordt tussen applicaties, alleen met hoe. Zoals bekend is het ook zonder middleware mogelijk om toepassingen met elkaar te laten communiceren. IP-sockets en SNA LU 6.2 zijn voorbeelden van protocollen die rechtstreeks vanuit programmatuur zijn te gebruiken. De eerste, zuiver op techniek gerichte, categorie middleware implementeert daar een schil overheen met de volgende drie eigenschappen:

1. Platformonafhankelijkheid. De meeste EAI-middleware is te implementeren op een flink aantal hardware- en softwareplatforms. Applicaties in die verschillende soorten omgevingen kunnen met elkaar communiceren zonder dat ze last hebben van de technische implementatieverschillen, zoals karaktersets (EBCDIC, ASCII) en taalafhankelijkheid. Ooit was ook onafhankelijkheid van netwerkprotocol (ethernet, token-ring, IPX, IP, SNA) een punt. Nu vrijwel alle operating systems IP ondersteunen wordt dat steeds minder belangrijk.

2. Gegarandeerde aflevering. Veel middleware-implementaties kunnen ervoor zorgen dat een aangeboden bericht ook gegarandeerd aankomt bij de beoogde ontvanger. De middleware is bestand tegen het tijdelijk niet beschikbaar zijn van applicaties, systemen en netwerken. Zodra een applicatie weer in de lucht komt, zorgt de middleware ervoor dat het bericht alsnog wordt afgeleverd. Natuurlijk moeten ook de samenwerkende applicaties ertegen bestand zijn dat de communicatie niet synchroon is. De verzendende applicatie moet onafhankelijk van de ontvangende applicatie kunnen doordraaien.

3. Locatietransparantie. Veel middleware-producten zorgen ervoor dat de applicaties niet van elkaar hoeven weten wáár ze geïmplementeerd zijn. Applicaties kunnen daardoor bijvoor-

beeld van plaats veranderen, zonder dat andere applicaties daar last van hebben.

Deze technische middleware valt uiteen in twee groepen, synchroon en asynchroon.

Synchrone middleware. Dit is een categorie middleware waarvoor het nodig is dat communicerende applicaties gelijktijdig beschikbaar zijn. Dit type wordt vooral toegepast binnen een organisatie. Die ene organisatie is dan verantwoordelijk voor het gelijktijdig beschikbaar zijn van de betrokken applicaties. Immers: als een aangeroepen applicatie even niet beschikbaar is, heeft de aanroepende applicatie een probleem. Zulke problemen zijn simpel op te lossen (of beter: te voorkomen) als beide applicaties een gemeenschappelijke eigenaar hebben. Voorbeelden van synchrone middleware zijn RPC, CORBA en COM. RPC, Remote Procedure Call, is gedefinieerd binnen de Distributed Computing Environment (DCE) van de Open Software Foundation. DCE is tamelijk complex en wordt steeds minder toegepast. CORBA, Common Object Request Broker Architecture, is een objectgeoriënteerde standaard van de Object Management Group. COM is Microsofts (Distributed) Common Object Model, de manier waarop Windows applicaties met elkaar samenwerken.

Asynchrone middleware. Voor dit type middleware is het niet nodig dat applicaties tegelijkertijd actief zijn (bijvoorbeeld: de ene applicatie stuurt een bericht naar de andere). Deze categorie gebruikt men vaak 'over organisaties heen'. De middleware, in dit geval ook wel MOM (Message Oriented Middleware) genoemd, garandeert dat berichten aankomen. Er zijn grofweg twee asynchrone interactiemodellen, publish & subscribe en message queing. Applicaties kunnen zich inschrijven ('subscribe') voor gegevens met een bepaald 'onderwerp' (in de praktijk 'topic' of 'subject' genoemd). Andere applicaties maken gegevens beschikbaar op zo'n onderwerp. Tibco's RendezVous is een voorbeeld van middleware die dit interactiemodel ondersteunt. Bij message queuing schrijft de ene applicatie gegevens naar een lokale queue (een door middleware beheerd 'bestand'). De middleware (zoals IBMs MQSeries, recentelijk omgedoopt tot WebsphereMQ, of Microsofts MSMQ) zorgt voor distributie van de inhoud van de queue over de IT-infrastructuur. Andere applicaties kunnen vervolgens de gegevens lezen uit lokale queues (lokaal ten opzichte van de lezende applicaties).

Datamanipulerende middleware

In aanvulling op de eerste categorie houdt de categorie datamanipulerende middleware zich ook bezig met wat er gecommuniceerd wordt tussen toepassingen. We denken dan met name aan message- of integration brokers, maar ook aan adapters.

Message- of integration brokers (zoals Mercator, Biztalk of Websphere MQ Integrator) zijn in staat om berichtstromen samen te voegen en te splitsen en om conversies uit te voeren op de inhoud van berichten. Zo kunnen ze een order die in XML-formaat door een webserver wordt aangeleverd opsplitsen en doorsturen. Orderregels voor de ene fabriek gaan dan bijvoorbeeld als IDoc (Sap's eigen Intermediate documentformaat) naar een Sap-systeem op Windows NT, orderregels voor de andere fabriek gaan in Edifact-formaat naar een legacy-systeem op het mainframe. Door hun centrale implementatie (in de zogenoemde Hub-and-Spoke-topologie) gedragen de meeste messagebrokers zich als 'single-point-of-failure' of anders wel als bottleneck. Inmiddels kunnen ze allemaal wel high available worden gemaakt door middel van hardwareclustering, en ondersteunt een grote groep ook load balancing. Een paar messagebrokers (zoals eGate van Seebeyond) echter zijn volledig decentraal. Ze worden weliswaar centraal geconfigureerd, maar ze worden gedistribueerd, 'dicht' bij de betrokken applicaties geïmplementeerd. Men spreekt dan wel over de Bus-topologie.
Ook adapters zet men vaak in voor het uitvoeren van berichtconversies en deze kunnen in dat geval tot de datamanipulerende middleware worden gerekend. Over adapters later meer.

Procesgeoriënteerde middleware

Hulpmiddelen voor Business Process Automation zijn de laatste ontwikkelingen in het domein van EAI, alhoewel de eerste implementaties zoals Crossworlds of Vitria al weer een jaar of vier bestaan. Een BPA-tool staat het toe om een bedrijfsproces te beschrijven (vaak: grafisch te modelleren) als een set van applicaties die gegevens met elkaar uitwisselen. Te denken valt aan het openen van een nieuwe betaalrekening bij een bank. De persoonsgegevens worden eerst gecontroleerd op juistheid in systeem A, vervolgens wordt de kredietwaardigheid geverifieerd in systeem B, de rekening aangemaakt in systeem C en een contract geprint in systeem D. Dit hele proces (inclusief alle mogelijke excepties en de afhandeling daarvan) kan een organisatie modelleren in het BPA-tool. In operatie leveren de BPA-tools 'execution engines' die de gemodelleerde processen uitvoeren door het (synchroon of asynchroon) aansturen van de betrokken applicaties.

Dit soort tools is met name handig in omgevingen waar regelmatig snel nieuwe bedrijfsprocessen ondersteund moeten worden over bestaande applicaties. In de praktijk echter zal de snelheid van invoering dan in

grote mate afhankelijk gaan worden van de volgende zaken:

- De menselijke functies die moeten participeren in het nieuwe bedrijfsproces. Die vergen immers de nodige acceptatie en inleertijd;
- Testen en in productie nemen van nieuwe applicatiekoppelingen, hetgeen zeker niet makkelijk is.

De afgelopen jaren is een duidelijke trend waarneembaar van EAI-leveranciers die BPA- of BPM-toepassingen in hun productportfolio opnemen, niet zelden door complete overnames van leveranciers van dit soort tools. Deze toepassingen waren reeds langere tijd op de markt beschikbaar, maar echte grootschalige implementaties lieten op zich wachten. Waarschijnlijk wegen in omgevingen met velerlei bestaande applicaties de sterke punten van BPA of BPM (flexibiliteit in procesmodellering en -aansturing) niet op tegen de zwakke punten (het niet op efficiënte en eenduidige wijze kunnen aansluiten op meerdere soorten bestaande applicaties). Het laatstgenoemde is nu juist het gebied waar EAI-middleware zijn meerwaarde al lange tijd heeft bewezen.

Tegelijkertijd is de opbouw van applicaties zelf ook niet veranderd gebleven: in vergelijking met een aantal jaren geleden worden nieuwe applicaties steeds meer opgesplitst in onafhankelijke functies (meer 'granulair') en sluiten dus steeds beter aan op zowel EAI- als BPA-tools.

B2B-middleware

B2B-middleware vormt een parallelle categorie van EAI-tools. B2B is het verschijnsel dat applicaties binnen het ene bedrijf communiceren met applicaties in het andere bedrijf.

B2B lijkt erg op EAI. Er is een aantal verschillen: security-eisen zijn scherper, er wordt gebruikgemaakt van open communicatieprotocollen, zoals HTTP(S), in plaats van de proprietary-protocollen die vaak door middleware-producten worden geïmplementeerd en berichtformaten voldoen vaker aan standaarden (vergelijk traditioneel EDI). We zien dat de B2B-middleware (zoals Webmethods) sterk is in het op elkaar afbeelden van berichtenstandaarden en dat EAI-middleware meer faciliteiten biedt zoals applicatieadapters en messagebrokering. We zien dat beide productgroepen naar elkaar toe groeien, al dan niet door overnames in de middleware-branche.

Additionele middleware-aspecten

Er is nog een aantal aspecten dat een rol speelt bij de inzet van middleware, XML, integratie op data(base)-niveau, transactionele aspecten en adapters. Deze aspecten beschrijven we in het nu volgende stuk.

In een verhaal als dit mag XML (eXtensible Markup Language) natuurlijk niet ontbreken. In EAI wordt XML gebruikt voor het vastleggen van (al dan niet complexe) datastructuren die tot op zekere hoogte 'self-describing zijn (aan de hand van de XML-tags kan immers ongeveer worden afgeleid wat een bericht betekent). XML is om twee redenen belangrijk. Ten eerste zijn er dankzij de hype enorme hoeveelheden goede tools beschikbaar om XML-berichten te manipuleren. Ten tweede richten B2B-uitwisselingsmechanismen (zoals ebXML) en berichtstandaarden (zoals RosettaNET) zich helemaal op XML. Niet kiezen voor XML betekent dat van deze ontwikkelingen niet of moeilijk gebruik kan worden gemaakt. Een nadeel van XML is het geëxplodeerde 'wireformaat': niet alleen de gegevens zelf maar ook de tags die de structuur en de attributen beschrijven moeten over het netwerk worden gecommuniceerd.

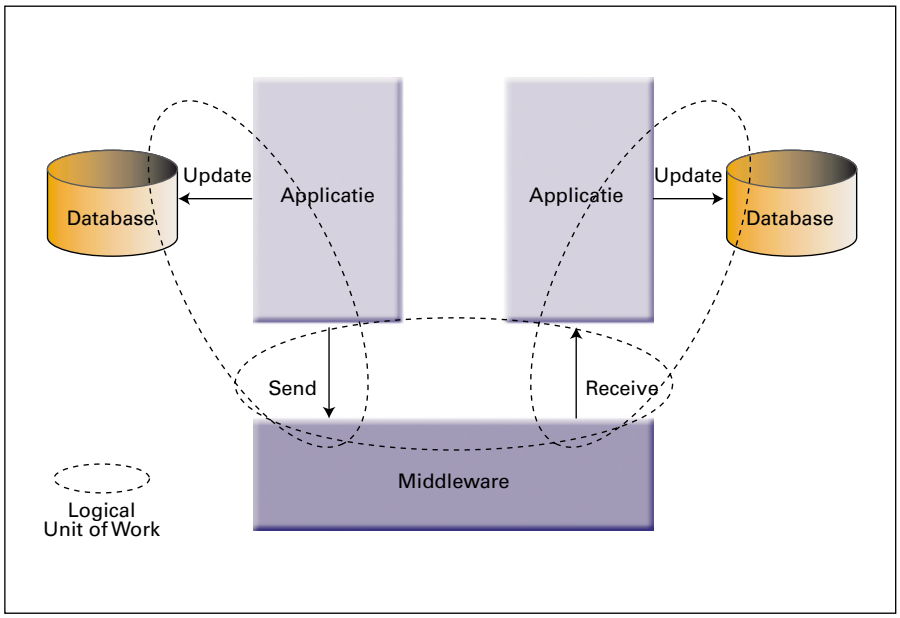
Integratie op data(base)niveau

Dit is een manier van integreren, waarbij de ene applicatie direct de database van de andere benadert. Probleem met deze interactievorm is dat een databasemodel geen stabiele interface tussen beide applicaties kan vormen. Zeker als applicaties zich ontwikkelen, moeten de databasemodellen kunnen meeveranderen. Als één databaseconfiguratie wijzigt moeten alle applicaties die daar direct gebruik van maken ook worden gewijzigd. Dit gaat nog voorbij aan het meer conceptuele feit dat een database data bevat die alleen in de context van de applicatie betekenis heeft. Door andere applicaties dezelfde data in hun eigen context te laten interpreteren ontstaan ongewenste verwarringen. Zou lezen uit een vreemde database onder omstandigheden toegestaan kunnen worden, schrijven is helemaal uit den boze. Daarbij wordt immers voorbijgegaan aan allerlei mogelijke validaties binnen de functionaliteit van de ontvangende applicatie. Soms gebruikt een organisatie wel een tussenliggende database ('operational datastore' genoemd), waarin de ene applicatie gegevens wegschrijft en de andere applicatie gegevens leest. Tussen de betrokken applicaties wordt dus een gemeenschappelijk datamodel afgesproken, dat door beide applicaties stabiel kan worden gehouden, en geen hindernis hoeft te vormen bij hun individuele evolutie. Tenzij dat datamodel erg complex is, wordt het gebruik van message-queuing (feitelijk een gedistribueerde datastore) en XML-berichtdefinities aanbevolen.

Transactionele aspecten

Bij middleware-implementaties getroost men zich grote moeite om gegarandeerde aflevering van berichten te waarborgen. Een bericht dat eenmaal ter verzending is aangeboden zal gegarandeerd ook bij de beoogde ontvangers aankomen (of er wordt een foutmelding gegenereerd, waarna een beheerder de situatie kan herstellen). Gegarandeerde aflevering is niet een

Advertentie



Drie gekoppelde Logical Units of Work.

zaak van de middleware alleen: aan de verzendende kant moet het versturen van een bericht onderdeel zijn van dezelfde 'Logical Unit of Work' als waarin updates naar de locale database worden geschreven. Het is van tweeën één: of het bericht wordt verstuurd èn de database wordt aangepast, of geen van beiden. Als het één wel en het ander niet gebeurt, raakt het bericht of verloren, of de applicatie is 'vergeten' (de applicatie kan het niet uit de database afleiden) dat het bericht is verstuurd. In dat laatste geval zal de applicatie het bericht nogmaals versturen en is het bericht feitelijk verdubbeld. Aan de ontvangende kant gelden soortgelijke overwegingen: als er iets misgaat tijdens het ontvangen van het bericht en het verwerken van dat feit in de lokale database, moet het bericht als niet ontvangen kunnen worden beschouwd.

We zien dat MOM-gebaseerde asynchrone interacties tussen applicaties zijn opgebouwd uit kettingen van transacties (zie afbeelding): een applicatie past de locale database aan èn verstuurt een bericht, de middleware ontvangt een bericht èn levert het af bij de ontvanger (guaranteed delivery) en de ontvangende applicatie haalt een bericht binnen èn past de locale database aan.

Adapters

Veel applicaties (met name legacy-applicaties of pakketten) kunnen we niet rechtstreeks op de middleware aansluiten. Er is dan een adapter nodig: software die aan de ene kant met de betrokken applicatie kan communiceren en aan de andere kant met de gekozen middleware. Veel middlewareleveranciers hebben adapters in hun assortiment waarmee hun middlewareproduct kan worden aangesloten op de gangbare applicatiepakketten. Zo hebben vrijwel alle middlewareleveranciers een Sap-adapter. Het open-source project 'OpenAdaptor' is een goed voorbeeld van een generiek inzetbare adapter.

Een andere reden om adapters te implementeren is het verbergen van de middleware-API voor de applicaties die er gebruik van maken. Het belangrijkste argument is dat er vooralsnog geen standaard is voor de middleware-APIs. De OAG (Open Applications Group) heeft een poging gedaan met AMI, Application Messaging Interface. Die is echter niet op brede schaal geadopteerd. JMS (Java Messaging Service) wordt wèl breed opgepikt, maar is alleen bruikbaar voor Java-applicaties. Hetzelfde geldt voor JCA (Java Connector Architecture), de generieke manier om vanuit Java op applicaties aan te sluiten. Zolang er geen duidelijke standaard-API beschikbaar is, schrikken veel bedrijven ervoor terug om product-specifieke APIs in te bouwen in hun applicaties. Door een adapterlaag tussen de middleware-API en de applicaties te zetten worden de consequenties van API-wijzigingen beperkt tot die adapterlaag. Dit soort adaptergebruik wordt ook wel 'Super-API' genoemd.

Adapters worden ook vaak gebruikt om gewenste functionaliteit toe te voegen die in het middleware-product ontbreekt. Denk daarbij aan foutdetectie en afhandeling, mogelijkheden voor het volgen en opsporen van berichten (tracking & tracing) en berichtformaatconversies.

Infrastructuur

In dit artikel zijn de verschillende categorieën van EAI-middleware besproken. In het volgende deel bespreken we de toepassingsstadia van EAI, waarbij met name het stadium waarin procesondersteunende functionaliteit wordt toegepast van belang is. Mits goed toegepast zien we dat een EAI-implementatie in combinatie met BPA-tools kan uitgroeien tot een business process enabling-infrastructuur.

Hans Martens en Bart Wolters

Hans Martens (hans.martens@atosorigin.com) is als senior middleware-architect werkzaam bij de Adaptive Infrastructure Solutions groep van AtosOrigin. Bart Wolters (bart.wolters@atosorigin.com) is senior system-architect en werkzaam bij dezelfde groep binnen AtosOrigin.