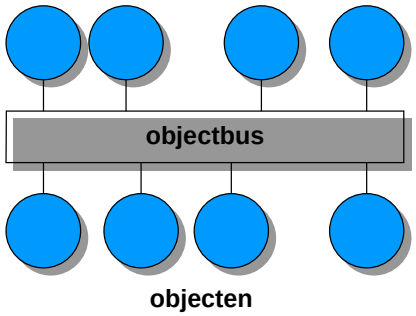
De vuist van EBM in de strijd om Internet

De strijd om de toekomst van het internet wordt momenteel hevig gevoerd rond de zogenaamde derde generatie internettoepassingen. Op het strijdtoneel bundelen partijen als IBM, Oracle, Sun Microsystems en Netscape de krachten als EBM (Everything But Microsoft) om een vuist te kunnen maken tegen Microsoft. In het strijdgewoel worden alle mogelijke wapens ingezet, variërend van het strooien met vooroordelen tot het overnemen van elkaars terminologie. Voor de (toekomstige) consument van deze derde generatie internettechnologie is het zo langzamerhand onmogelijk om het strijdtoneel te overzien, laat staan om strategische keuzen te maken met betrekking tot het inzetten van deze technologie. Dit artikel geeft enerzijds de (toekomstige) consument een globaal overzicht van het strijdtoneel en verschaft anderzijds een dieper inzicht in de technologie achter de gedachte van EBM.

Waar de strijdende partijen het over eens zijn, is dat de basis van derde generatie internettoepassingen gevormd wordt door objecten die onderling via het internet communiceren middels een objectbus.

Objecten en een objectbus

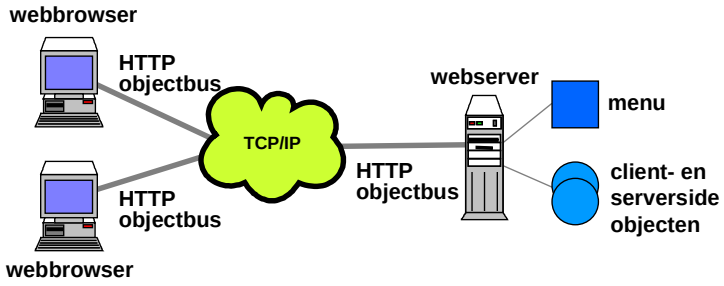
Een objectbus biedt de infrastructurele voorziening zodat objecten mogelijk onafhankelijk van hardware, besturingssysteem, programmeertaal en netwerkprotocol met elkaar kunnen communiceren. Een objectbus biedt dus een transparante vorm van communicatie tussen objecten (zie figuur 1).



Figuur 1 Objecten communiceren middels een objectbus

De objectbus treedt op als een soort makelaar. Hij redigeert vragen van objecten naar andere objecten en verzorgt de terugkoppeling van het antwoord. De objecten bieden elkaar diensten aan.

Ter illustratie: Verzekeringsbedrijf X heeft een derde generatie internettoepassing gebouwd ter ondersteuning van het offertetraject voor tussenpersonen.



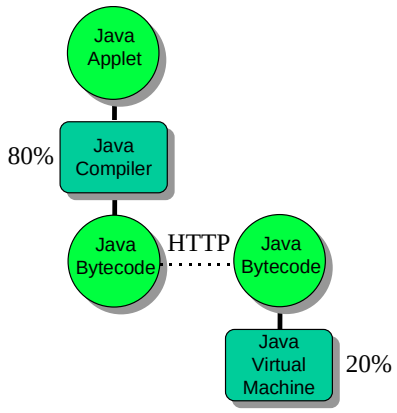
Figuur 2 Derde generatie internettoepassing van een verzekeringsbedrijf

De toepassing kent een centraal menu met het logo van het bedrijf en de mogelijke services die aan de tussenpersonen geboden worden. Wat opvalt is dat naast de elementen van een objectbus en objecten ook de al bekende elementen uit de eerste twee generaties gebruikt worden: de webbrowser, de webserver, html-pagina's en het HTTP-protocol bovenop TCP/IP (zie figuur 2). Het menu is gerealiseerd middels een html-pagina dat zich aan de kant van de webserver bevindt. Op het moment dat de tussenpersoon de betreffende html-pagina middels het selecteren van de betreffende URL (*Unified Resource Locator*) in zijn webbrowser oproept, vindt de distributie van het menu plaats van de webserver naar de webbrowser met behulp van het HTTP-protocol. De aangeboden servies zijn gerealiseerd middels een aantal objecten. Door in het menu een service te kiezen, vindt het transport

van de benodigde client-side objecten van de webserver naar de webbrowser plaats, eveneens middels het HTTP-protocol. Als de benodigde client-side objecten, betreffende het grafische gebruikersinterface (GUI) van de aangeboden service, volledig bij de webbrowser zijn aangekomen, worden deze geëxecuteerd. Vervolgens wordt afhankelijk van de invoer van de tussenpersoon, bijvoorbeeld naw-gegevens en de leeftijd van een klant bij de aanvraag van een pensioenverzekering, de ingevoerde gegevens naar een server-side object gecommuniceerd middels de objectbus voor verdere verwerking. Eventueel vindt er communicatie terug plaats, bijvoorbeeld de te betalen premie voor de pensioenverzekering, via de objectbus.

Juist om de invulling van bovenstaand concept loeit de strijd op en staan EBM en Microsoft lijnrecht tegenover elkaar. In de visie van Microsoft worden de objecten en de objectbus gerealiseerd middels de microsoftcentrische technologie: ActiveX en Distributed Com. Terwijl in de visie van EBM de invulling respectievelijk plaatsvindt middels open standaarden: Java en Corba.

Java
Java is een krachtig, platformonafhankelijk en object-georiënteerde programmeertaal voor derde generatie internettoepassingen. Met Java worden objecten gecreëerd, zogenaamde Java Applets. Om platformonafhankelijkheid te creëren, vindt compilatie van een Java Applet in twee stappen plaats.



Figuur 3 Java Compiler en JVM

Allereerst wordt de Java Applet gecompileerd tot platformonafhankelijk Java Bytecode. Deze Java Bytecode bevindt zich op de webserver. Nadat een webbrowser een verzoek doet voor een Java Applet, verstuurt de webserver de Java Bytecode met behulp van het http-protocol. Vervolgens wordt in de webbrowser de Java Bytecode geïnterpreteerd door de zogenaamde Java Virtual Machine. De platformonafhankelijkheid wordt gecreëerd doordat er diverse implementaties beschikbaar zijn van de Java Virtual Machine.

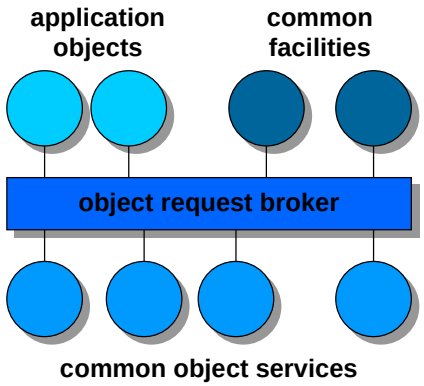
In de markt wordt deze platform onafhankelijkheid aangeduid met het “Write Once Run Anywhere”-principe (WORA) waarmee gedoeld wordt op het feit dat een applicatie slechts éénmaal gebouwd hoeft te worden en vervolgens op verschillende platformen geëxecuteerd kan worden.

Om met behulp van hergebruik van Java Applets applicaties te bouwen, het zogenaamde *Component Based Application Development*, is een standaard ontwikkeld: *Java Beans*. Deze standaard geeft aan hoe herbruikbare componenten in Java gebouwd moeten worden, hoe de eigenschappen van een component aan een ontwikkelomgeving beschikbaar gesteld moeten worden en hoe de communicatie tussen componenten onderling plaatsvindt. Een Java Applet dat aan de eisen van de Java Beans standaard voldoet wordt een Java Bean genoemd. Bedrijven als JScape (o.a. Java Beans voor de bouw van grafische gebruikersinterfaces), Corel, EnterpriseSoft, Gemstone, IBM (o.a. CICS Gateway), K&A Software, KL Group, Lotus Development, Novell, ProtoView Development, Rogue Wave, en Stingray Software ontwikkelen standaard Java Beans.

De Java programmeertaal is een platformonafhankelijk en object-georiënteerde programmeertaal. Echter, doordat Java alleen de communicatie tussen Java Applets onderling ondersteunt, is een infra-

structuur die alleen gebaseerd is op Java niet open. Door het gebruik van Corba als objectbus voor de communicatie tussen Java Applets en andere objecten wordt een open infrastructuur gecreëerd.

Corba
Corba staat voor *Common Object Request Broker Architecture* (zie figuur 4). Deze objectbus middleware standaard is ontwikkeld (en wordt nog verder ontwikkeld) door de *Object Management Group* (OMG). De OMG is in mei 1989 opgericht door acht bedrijven: 3Com, American Airlines, Canon, Data General, Hewlett-Packard, Philips Telecommunications, Sun Microsystems en Unisys. Sinds oktober 1989 opereert de OMG als onafhankelijke non-profit instelling. Medio 1998 zijn meer dan 700 bedrijven en instellingen uit de softwarebranche lid van de OMG.



Figuur 4 Corba

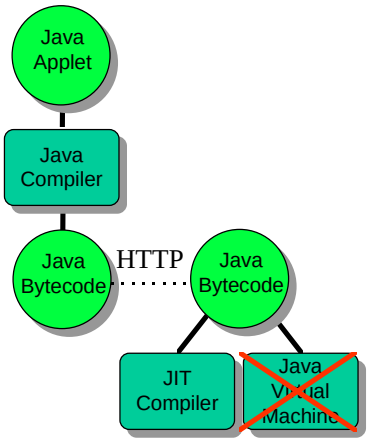
De *Object Request Broker* (ORB) is de objectbus van Corba. De ORB verzorgt de communicatie tussen objecten. Objecten kunnen tijdens uitvoer aan de ORB vragen welke diensten beschikbaar zijn en er vervolgens gebruik van maken. De *Common Object Services* zijn een verzameling van elementaire diensten verpakt als objecten. Ze zijn een uitbreiding op de diensten van een ORB. Er zijn services gedefinieerd voor bijvoorbeeld security, transacties, concurrency en persistentie. De *Application Objects* zijn de objecten die de organisatie modelleren en de gebruikers bij hun werkzaamheden ondersteunen. Deze objecten worden ook wel aangeduid met de term *Business Objects*. De application objects maken gebruik van object services. Om de ontwikkeling van toepassingen te vereenvoudigen kent Corba de zogenaamde *Common Facilities*. Dit zijn standaardraamwerken (*frameworks*) van samenwerkende objecten die kant-en-klaar ingezet kunnen worden in specifieke situaties. Er zijn twee soorten Common Facilities: horizontale en verticale. De horizontale Common Facilities bieden domeinonafhankelijke ondersteuning, zoals bijvoorbeeld werkstroombesturing. De verticale Common Facilities bevatten objecten voor de ondersteuning van specifieke bedrijfstakken, bijvoorbeeld de sociale zekerheid.

Op dit moment zijn er er verschillende commerciële Corba implementaties beschikbaar. De twee meest bekende zijn: Orbix van Iona Technologies Ltd en VisiBroker van Visigenic Software Inc. Een veelbelovende Corba implementatie betreft CBConnector van IBM (de officiële release vindt medio 1998 plaats).

Vooroordeel
Een veel gehoord vooroordeel over het gebruik van Java en Corba voor derde generatie internettoepassingen betreft de slechte performance van Java door het platformonafhankelijke karakter. In principe gaat platform-onafhankelijkheid altijd ten koste van performance, maar bij de ontwikkeling van Java, heeft men deze trade-off meegenomen door het introduceren van een compilatieslag van platform-onafhankelijke Java sourcecode naar platform-onafhankelijke Java Bytecode. Bij deze compilatieslag wordt ongeveer tachtig procent van het werk vooraf al uitgevoerd (zie figuur 3). Slechts de overige twintig procent bestaat uit de feitelijke interpretatie van deze Java Bytecode waardoor het performanceverlies beperkt wordt en de platform-onafhankelijkheid behouden blijft.

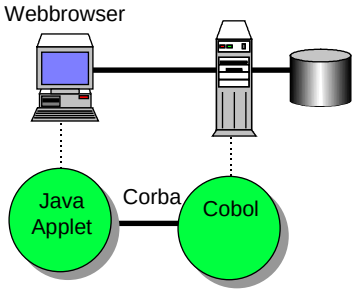
Om de performance van deze Java toepassing nog verder te verbeteren, vindt er een andere ontwikkeling plaats die onlangs is ingezet: de zogenaamde JIT-compiler. Nadat de Java Bytecode middels het HTTP-protocol naar het betreffende platform getransporteerd is, vindt er geen interpretatie meer plaats

door de JVM maar wordt de Java Bytecode naar native uitvoerbare code gecompileerd. De JIT-compiler ("Just in Time" compiler) zorgt hiervoor en neemt derhalve de feitelijke taak van de JVM over (zie figuur 5). In de meeste situaties is op deze manier zelfs een vergelijkbare performance met gecompileerde code te realiseren.



Figuur 5 JIT-Compiler

Een andere mogelijkheid betreft de combinatie van Java en Corba met talen zoals bijvoorbeeld Cobol of C++ voor de realisatie van een goede performance aan de server-side. Doordat Corba de communicatie tussen objecten mogelijk maakt onafhankelijk van hardware, besturingssysteem, netwerk protocollen en programmeertaal wordt deze mogelijkheid geboden.



Figuur 6 De combinatie van Java en Cobol

Op deze manier is het bijvoorbeeld mogelijk om de grafische gebruikersinterface (GUI) van de offertetoepassing middels Java te realiseren. Op deze manier wordt het voordeel van het "WORA" principe van Java aan de client-side volledig benut en wordt een goede performance gerealiseerd aan de server-side door de applicatielogica en de koppeling met bijvoorbeeld een database middels Cobol te realiseren.

Auteur

Drs. P.J. Koning is werkzaam bij de Technology Consulting Group van Cap Gemini Nederland b.v.

Literatuur

- The Essential Client/Server Survival Guide - Second Edition, *R. Orfali, D. Harkey en J. Edwards*, John Wiley & Sons, 1996.
- The Essential Distributed Objects Survival Guide - Second Edition, *R. Orfali, D. Harkey en J. Edwards*, John Wiley & Sons, 1996.
- Gedistribueerde Objecten: Corba, OLE en OpenDoc, *G.M.A. van der Harst*, Personal Computer Gids, 1996.
- Client/Server Programming with Java and Corba, *R. Orfali en D. Harkey*, John Wiley & Sons 1997.