Component Broker

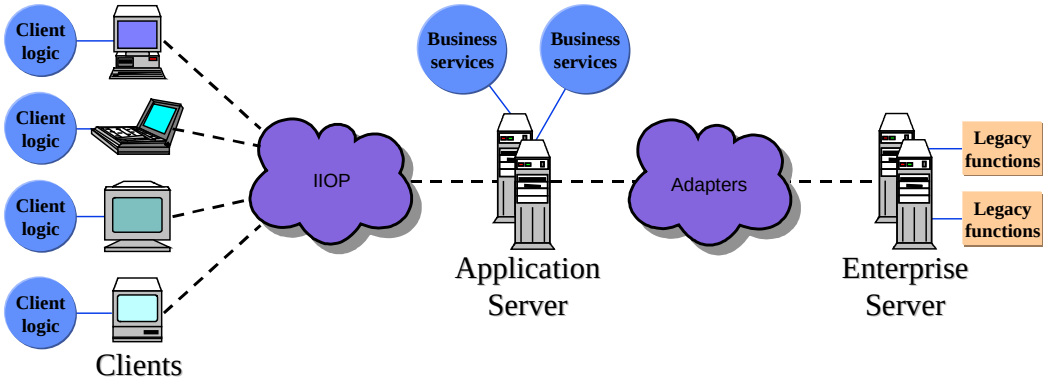
Modern en bedrijfskritisch hand in hand

Een aantal jaren geleden realiseerde IBM zich dat de toenmalige strategie voor applicatieontwikkeling van grote bedrijfskritische applicaties op termijn plaats zou moeten maken voor een meer open, object-bus-based, multi-tier client/server strategie en ontwikkelde het Network Computing Framework (NCF).

Deze technologische visie definieert onder andere dat toekomstige applicaties gerealiseerd zullen worden volgens het multi-tier client/server principe en derhalve over minimaal een drietal fysieke lagen verdeeld zullen worden (zie figuur 1):

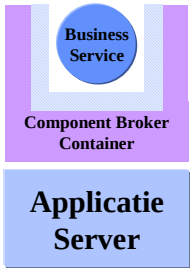
- *Clients*
- *Application Servers*
- *Enterprise Servers*

De Client-tier bevat de presentatie van de applicatie in de *client logic*. Deze presentatie kan gerealiseerd worden middels een traditionele programmeertaal en derhalve getoond worden in een traditionele client, of middels internettechnologie en derhalve getoond worden in een webbrowser, een Network Computer, etc.



Figuur 1 Network Computing Framework van IBM

De Application Server-tier bevat *Business Services* die de applicatielogica huisvesten. De Business Services maken gebruik van de standaardvoorzieningen die de Application Server levert, o.a. persistency voor de opslag van gegevens en adapters voor het hergebruiken van traditionele (legacy) functies. De Enterprise Server-tier bevat enerzijds de gegevens van de applicatie en eventueel de traditionele legacy functies die door de Business Services uit de Application Server-tier hergebruikt worden middels de adapters.

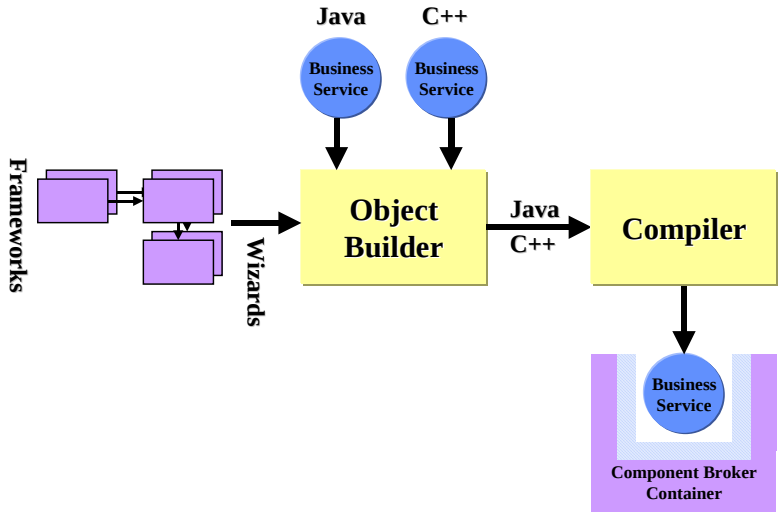


Figuur 2 Een Business Service draait in een container

Voor de realisatie van het Network Computing Framework levert IBM, na een *early adopter* programma van ongeveer 13 maanden met o.a. Swiss Bank en Bank of America, sinds kort drie nieuwe producten onder de vlag Component Broker: Object Builder, Component Broker Server en Component Broker System Manager.

Object Builder
Object Builder betreft IBM's omgeving voor de realisatie van Business Services. Deze Business Services draaien op de Application Server in zogenaamde Component Broker Containers (zie figuur 2). De Business Services kunnen binnen één Object Builder omgeving in Java of in C++ gespecificeerd worden (zie figuur 3). Vervolgens wordt, door gebruik te maken van twee aanwezige frameworks, MOFW en AAF, code gegenereerd die de Business Services en containers representeren. Deze gegenereerde code vormt vervolgens input voor Java en C++ compilers, die de uiteindelijke run-time code vervaardigen.

Het gebruik van de frameworks is vereenvoudigd binnen Object Builder doordat middels Wizards het framework ingesteld en op de juiste wijze aan de applicatielogica "geplakt" wordt.



Figuur 3 Generatie van Business Services en containers in Object Builder

Managed Object FrameWork
Het eerste framework dat binnen Object Builder gebruikt wordt betreft het *Managed Object FrameWork*. Het MOFW heeft tot doel Business Services beheersbaar te maken voor de Application Server. Voor het beheersbaar maken van de Business Services worden een aantal design patterns [3] gebruikt:

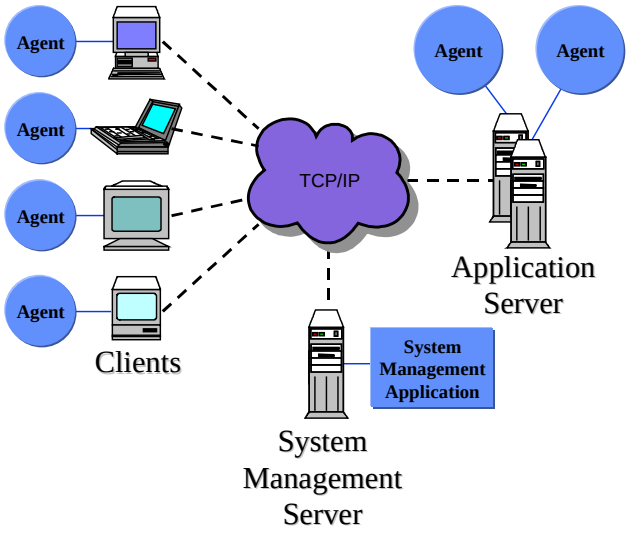
- *Business logic pattern* — voor het scheiden van de pure business logic van technische zaken zoals data access, management, caching en persistency. Dit design pattern maakt het mogelijk dat een business service los in een Component Broker Container kan draaien (zie figuur 2).
- *Proxy pattern* — voor het transparant kunnen aanroepen van applicatielogica in de Applicatie Server-tier door de client-tier via een proxy-object. Indien de proxy en de applicatielogica op verschillende platformen staan, wordt uiteindelijk de communicatie middels Corba gerealiseerd. Mochten de proxy en de applicatielogica op hetzelfde platform staan, wordt een directere vorm van communicatie gebruikt. Dit levert geoptimaliseerde performance op voor de communicatie tussen de client-tier en de Applicatie Server-tier.
- *Data access pattern* — levert een uniform interface voor het afhandelen van opslaan van gegevens.
- *Persistency pattern* — voert de opslag van informatie uit voor een specifiek medium middels een zogenaamde adapter. Momenteel levert IBM adapters voor DB2, Oracle, IMS, CICS, MQSeries en SAP/R3. Het gebruik van het persistency pattern in relatie tot het data access pattern levert een loskoppeling van de applicatielogica en het medium waarop gegevens opgeslagen worden.
- *Management pattern* — voor het scheiden van de applicatielogica van beheerfaciliteiten zoals beveiliging en transactie-mogelijkheden.
- *Caching pattern* — geeft mogelijkheden om de toegang tot een opslagmedium te optimaliseren middels het gebruik van een aantal soorten caching-mechanisme.

"Onder water" maakt het Managed Object FrameWork gebruik van de standaard (low-level) services die de Applicatie Server levert (zie Component Broker Server).

Application Adapter Framework
Het tweede framework betreft het Application Adapter Framework. Het AAF levert een plaats waarin Business Services kunnen leven door het afdekken van life cycle functionaliteit: het creëren, verwijderen, activeren en deactiveren van Business Services. Het AAF maakt enerzijds gebruik van het MOFW en anderzijds van de standaard (low-level) services die de Applicatie Server levert voor de invulling van zijn functionaliteit. De functionaliteit die het AAF levert, wordt voornamelijk door de client-logica gebruikt, bijvoorbeeld om een nieuwe Business Service te creëren.

Component Broker Server
De Component Broker Server is een belangrijk onderdeel van de infrastructuur, de Application Server (zie figuur 1), waarop de in Object Builder gebouwde Business Services en containers draaien. De huidige versie van de Component Broker Server bevat o.a. een Corba 2.0 compliant ORB voor de communicatie tussen de Clients en Business Services middels IIOP, een JVM voor de interpretatie van in de Object Builder gerealiseerde Business Services in Java, workload balancing voor het verdelen van de aangevraagde werklast over meerdere Business Services en een aantal standaard services:

- *Naming service*
  Component Broker Server implementeert de standaard Corba 2.0 compliant Naming service die het mogelijk maakt dat services aan elkaar kunnen refereren middels een naam.
- *Security service*
  De security service die Component Broker Server biedt maakt het mogelijk om enerzijds de communicatie tussen services te beveiligingen en anderzijds de authenticatie van clients te realiseren. De security service is Corba 2.0 compliant geïmplementeerd.
- *Event service*
  De life cycle service biedt de mogelijkheid om objecten in te lichten bij het voorkomen van een bepaalde gebeurtenis (event). De life cycle service is Corba 2.0 compliant geïmplementeerd.
- *Identity service*
  De Corba 2.0 compliant identity service maakt het mogelijk om te achterhalen of twee proxies naar één Business Service wijzen om vast te kunnen stellen of de twee proxies identiek zijn of niet.
- *Life cycle service*
  De Corba 2.0 compliant life cycle service die Component Broker Server implementeert, ondersteunt het creëren, kopiëren, verplaatsen en verwijderen van Business Services. Daarnaast heeft IBM de mogelijkheid geïmplementeerd om op basis van bepaalde gegevens Business Services te vinden.



Figuur 4 System Management

- *Externalization service*
  De Corba 2.0 compliant externalization service die IBM geïmplementeerd heeft, biedt de mogelijkheid om objecten *plat* op te slaan voor bijvoorbeeld transport naar een ander platform.
- *Object Transaction service*

De OTS biedt de mogelijkheid om transacties te definiëren over objecten heen middels een two-phase commit protocol. De OTS is Corba 2.0 compliant geïmplementeerd.

- *Concurrency service*
  De concurrency service (Corba 2.0 compliant) die door IBM geïmplementeerd is biedt de mogelijkheid om verschillende services te synchroniseren.
- *Query service*
  De query service is door IBM Corba 2.0 compliant geïmplementeerd en levert de mogelijkheid om objecten te selecteren middels een SQL-achtige vraagtaal op basis van bepaalde waarden van een attribuut. De door de *Object Database Management Group* (ODBMG) ontwikkelde standaard wordt volledig ondersteund: *Object Query Language* (OQL).

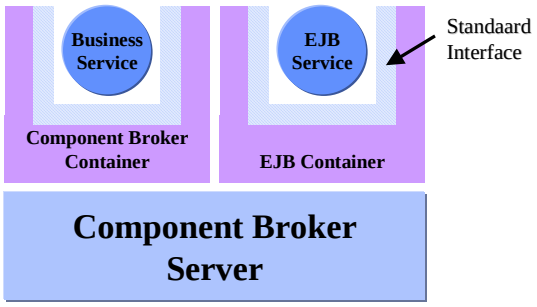
Component Broker System Manager
De Component Broker System Manager maakt het mogelijk om de gerealiseerde Business Applicaties te configureren, distribueren, beheren en controleren door het gebruik van Agent-technologie en een System Management Applicatie (zie figuur 4). Op elke Client en Component Broker Server zorgt een Agent voor de afhandeling van de system management taken. Vanaf de System Management Applicatie wordt het beheer centraal geregeld, doordat alle informatie van de Agents samenkomt en van hier uit de eventuele taken naar de Agents gedelegeerd worden. Door het onderkennen van Server Groups is het mogelijk om de workload te managen en eventuele fail-over te regelen.

Kort samengevat kan gesteld worden dat met Object Builder, Component Broker Server en Component Broker System Manager op basis van een open, objectbus-based, multi-tier client/server strategie (NCF) Business Services en containers gebouwd, uitgerold, geëxecuteerd en beheerd kunnen worden

Integratie
Voor de realisatie van de Clients die gebruik maken van de Business Services via containers levert IBM integratie met VisualAge for Java voor de realisatie van Java-clients. Voor de integratie met traditionele clients in C++ wordt integratie met VisualAge C++ en Microsoft Visual C++ geleverd.

Voor de analyse en het ontwerp van Business Applicaties wordt integratie met Rational Rose, middels XML/XMI, en Erwin geleverd. Vanuit Rational Rose is het mogelijk om de interface van de Business Services te genereren en op basis hiervan in Object Builder middels het gebruik van wizards uiteindelijk de containers te genereren.

Toekomst
Een belangrijke uitbreiding van Component Broker Server die voor het eind van dit kwartaal gepland staat, is de ondersteuning van de Enterprise Java Beans standaard.



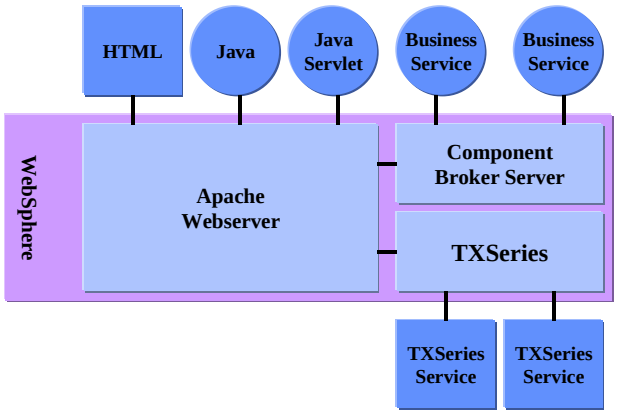
Figuur 5 Ondersteuning van EJB in Component Broker Server

Met de ondersteuning van de EJB standaard wordt het mogelijk om Enterprise Java Beans in een Enterprise Java Bean container op de Component Broker Server te executeren. Het voordeel van de EJB-standaard is de standaardisering van het interface tussen EJB en de EJB container. Hierdoor wordt een onafhankelijkheid van de Business Service ten opzichten van de container en de Applicatie Server gerealiseerd. Hierdoor is het enerzijds mogelijk om EJB's die in andere ontwikkelomgevingen gerealiseerd zijn binnen een Component Broker-omgeving te executeren en anderzijds EJB's van een Component Broker-omgeving naar een andere omgeving te verplaatsen. Andere partijen die de EJB-standaard ondersteunen betreffen o.a. Forté Software, Baan, BEA Systems, Inprise, Oracle en natuurlijk Sun Microsystems.

Momenteel is Component Broker Server beschikbaar voor Microsoft Windows NT en IBM/AIX. Als alles volgens planning verloopt, wordt aan het eind van dit kwartaal Component Broker Server voor IBM's OS/390 en in het derde kwartaal van 1999 Component Broker Server voor Sun's Solaris geleverd. Voor het eind van dit jaar staat Component Broker Server voor HP-UX gepland. Terwijl de implementaties voor Microsoft Windows NT, IBM/AIX, OS/390, Sun Solaris en HP-UX van elkaar afgeleid zijn, is de implementatie van Component Broker Server op OS/390 volledig nieuw ontwikkeld en, in verband met de performance winst, sterk verweven met de onderliggende hardware.

Momenteel ondersteunt Object Builder XML/XMI voor de uitwisseling van informatie tussen diverse omgevingen. Naar verwachting zal XML/XMI de standaard worden in de toekomst en derhalve kunnen andere tools, b.v. Select Enterprise, gebruikt worden in combinatie met Object Builder.

Met betrekking tot de integratie van diverse technologieën heeft IBM onlangs de weg ingezet om tot de consolidatie van Component Broker Server, TXSeries en Apache Webserver te komen in Websphere.



Figuur 6 WebSphere

Op deze wijze wordt de volledige integratie van een aantal technologieën gerealiseerd:

- *Internettechnologie* — middels de Apache Webserver voor de distributie van HTML en Java naar de webbrowser en het uitvoeren van Java Servlets op de webserver.
- *Corba-technologie* — middels Component Broker Server voor de onderlinge communicatie tussen C++ clients, Java clients en Business Services.
- *CICS, Encina en MQSeries technologie* — middels TXSeries voor de synchrone en asynchrone communicatie tussen traditionele applicaties.

Op deze wijze wordt het mogelijk om een internetapplicatie te bouwen voor de verkoop van producten waarbij de presentatie in HTML en Java gerealiseerd is, en de applicatielogica middels Business Services in Object Builder gerealiseerd is en op de Component Broker Server draait. Een andere mogelijkheid betreft het realiseren van de applicatielogica met meer traditionele technologie, bijvoorbeeld CICS, Encina of MQSeries, middels TXSeries Services.

Voordelen

Component Broker betreft een set van producten die binnen het NCF gepositioneerd kan worden voor het bouwen van grote bedrijfskritische systemen in een omgeving waarin (momenteel) IBM producten zoals MQSeries, CICS, DB2 en IMS dominant aanwezig zijn. Ten opzichte van een traditionele robuuste, schaalbare en bedrijfskritische IBM-omgeving, met bijvoorbeeld Cobol/CICS, kan middels Component Broker technologie een aantal voordelen bereikt worden:

- *Verhoging van de applicatiekwaliteit*
  Doordat alleen de Business Services gebouwd moeten worden, de containers worden gegenereerd op basis van het MOFW, het AAF en de wizards, kan de ontwikkelaar zich concentreren op het ontwikkelen van deze Business Services. Dit in tegenstelling tot een standaard Cobol/CICS-omgeving waarin o.a. alle middleware en database-afhandeling geprogrammeerd dient te worden. Het gebruik van Object Builder ontdoet de ontwikkelaar van deze technische complexiteit. Dit komt de uiteindelijke Business Service en derhalve de applicatie ten goede.

- *Productiviteitswinst*
  Daarnaast kan een aanzienlijke productiviteitswinst gerealiseerd eveneens doordat een hoop technische zaken niet meer zelf geprogrammeerd dienen te worden. Deze zaken worden door de wizards, beide frameworks en de Applicatie Server afgehandeld.
- *Meer flexibiliteit*
  Doordat met Object Builder applicaties ontwikkeld kunnen worden onafhankelijk van doelplatform (Windows NT, IBM/AIX, OS/390 en Sun Solaris), ontwikkeltaal (Java en C++), databases (DB2 en Oracle) en locatie kunnen applicaties op diverse wijze uitgerold worden. Zo kan een applicatie bijvoorbeeld vandaag in een 2-tier client/server opstelling uitgerold worden en morgen, in verband met performanceverbetering in een 3-tier client/server opstelling. Dit omdat de Business Services van elkaar niet weten op welke locatie ze executeren.
- *Behoud van investeringen*
  De investeringen die in het verleden gedaan zijn in (legacy) functionaliteit en gegevens, met name DB2, Oracle, IMS, CICS, MQSeries en SAP/R3, kunnen behouden worden door de het gebruik van de diverse adapters. Daarnaast is het mogelijk om eigen adapters te bouwen voor de ontsluiting van andere omgevingen. Op deze wijze wordt het mogelijk om ook niet IBM-omgevingen te ontsluiten.
- *Geleidelijk migratiepad*
  Door de duidelijke plaats van legacy systemen binnen de NCF-strategie is het mogelijk om op soepele wijze te migreren van de bestaande (traditionele) situatie naar een moderne, open, objectbus-based, multi-tier client/server omgeving middels Component Broker.

Uiteraard levert Component Broker, middels zaken zoals load balancing in de Component Broker Server en de Component Broker System Manager, de robuustheid en de schaalbaarheid van traditionele technologie zoals bijvoorbeeld Cobol/CICS, die voor grote bedrijfkritische applicaties van levensbelang zijn.

Auteur

Drs. P.J. Koning, Technology Consulting Cap Gemini.

Literatuur

1. *Red Book: IBM Component Broker Connector Overview*, Saniya Ben Hassen et al., International Business Machines Corporation, december 1997.
2. *IBM Network Computing Framework for e-business*, International Business Machines Corporation, www.software.ibm.com/ebusiness, 1998.
3. *Design Patterns – Elements of Reusable Object-Oriented Software*, E. Gamma et al., Addison-Wesley, 1995.