Zeitschrift: Comtec : Informations- und Telekommunikationstechnologie = information and telecommunication technology

Herausgeber: Swisscom

Band: 78 (2000)

Heft: 7-8

Artikel: Komponententechnologie mit WebSphere

Autor: Gisiger, Hans Peter

DOI: https://doi.org/10.5169/seals-876462

Komponententechnologie mit WebSphere

Im März 2000 hat das Management CIT entschieden, das Produkt WebSphere Enterprise Edition von IBM als zukünftige Entwicklungs- und Integrationsplattform (Middleware) einzusetzen. Damit wird die Grundlage für eine einheitliche Software- und Infrastrukturarchitektur geschaffen, die als Basis für zukünftige Projekte gilt.

HANS PETER GISIGER, BERN

Nach einer sorgfältigen Evaluation wurde letztes Jahr gemeinsam mit der Firma IBM ein «Engagement Project Ramp-up WebSphere» gestartet und im April dieses Jahres erfolgreich abgeschlossen. In einer ersten Phase wurden gemeinsam mit IBM die notwendigen Skills erworben. Und in einer zweiten Phase hat man dann mit diesem Produkt eine Pilotanwendung mittlerer Grösse und Komplexität realisiert. Diese Anwendung wurde anschliessend erfolgreich auf das IBM-Mainframe OS/390 übertragen und in Betrieb genommen. Obwohl es sich bei WebSphere Enterprise Edition auf dem Mainframe zurzeit noch um eine Betaversion handelt, konnte die Pilotanwendung ohne grösseren Aufwand erfolgreich von NT auf das Mainframe portiert werden.

«Engagement Project Ramp-up WebSphere»

Die heutigen Unternehmen sind permanent mit neuen Herausforderungen des Marktes und des Umfeldes konfrontiert, wie beispielsweise:

- neue Geschäftsmodelle – infolge der Transformation der Märkte –, speziell getrieben durch die Herausforderung des E-Business
- Erhöhung der Flexibilität und des Reaktionsvermögens der Kernprozesse
- Reduktion der Time-to-Market
- Kostentransparenz, Reduktion des IT-Budgets

Einerseits kann diesen Herausforderungen vielfach nicht entsprochen werden, da die existierende IT nicht flexibel genug ist. Andererseits bietet aber gerade der Einsatz der integrierten IT einen Konkurrenzvorteil. Einer der erfolgversprechendsten Ansätze, diese Herausforderungen zu meistern, ist der Aufbau einer verteilten Komponentenarchitektur. Diese Technologie kann die Komplexität existierender, agiler Inter-Enterprise-Systeme, bestehend aus grossen, verteilten Applikationen, stark reduzieren. Seit langer Zeit besteht der Wunsch nach Wiederverwendbarkeit. Objektorientierte Ansätze haben diese Möglichkeiten zwar geprägt, in der Praxis haben sie sich jedoch nicht wirkungsvoll durchgesetzt. Komponenten haben im Gegensatz zu den Klassen und Objekten eine gröbere Granularität und umfassen meist einen abgeschlossenen Business/Applikations-Domain. Mit dem Einsatz von Komponenten kann diese Forderung nach Wiederverwendbarkeit Erfolg viel versprechend neu aufgenommen werden.
Obwohl die Komponententechnologie mit JavaBeans, Enterprise JavaBeans, CORBA-Komponenten inzwischen den notwendigen Reifegrad erreicht hat, wird ohne kommunizierbaren und effektiven Nutzen kaum jemand diese neue Technologie einsetzen. Im Beitrag wird ein objektiver Nutzen der Komponententechnologie illustriert.
In der Zwischenzeit gibt es verschiedene Komponentenmodelle. Ihre Spezifikationen unterscheiden sich einerseits nach

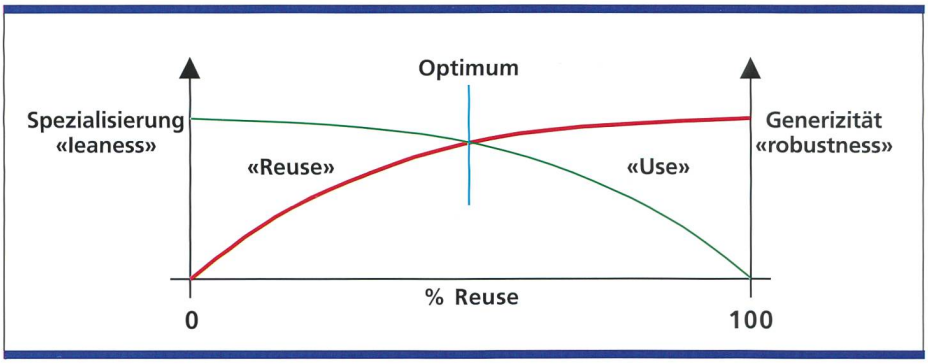

Bild 1. Wiederverwendbarkeit: Use & Reuse.

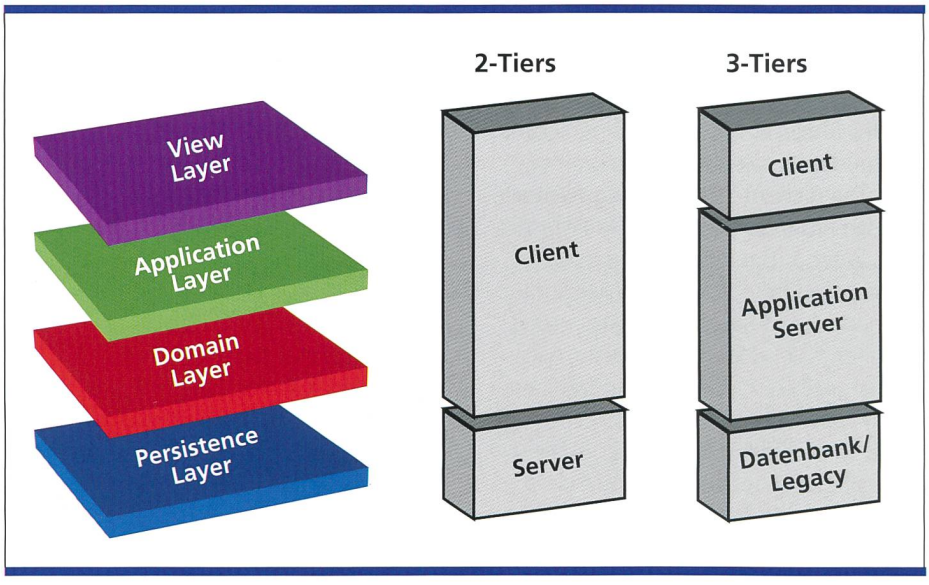

Bild 2. 2-Tiers kontra 3-Tiers.

Anwendungsmöglichkeiten und andererseits nach Lieferanten(-Gruppen) und Konsortien. Je nach Modell können Komponenten in entsprechende System- und Applikationsarchitekturen eingesetzt werden. Details zu den einzelnen Komponentenmodellen müssen der Literatur entnommen werden. Der Beitrag zeigt eine Applikationsarchitektur, in der Komponenten eingesetzt werden können.
Für den praktischen Einsatz wird eine Komponentenlösung mit Industriereife benötigt. Die Server-Laufzeitumgebung muss hochzuverlässig sein, muss eine gute Performance aufweisen, muss bei zunehmender Last (Workload) entsprechend skalieren und muss hochverfügbar sein. Beides, sowohl Integration wie Konnektivität zum Resource Manager, ist wichtig. In vielen Applikationsbereichen ist eine Transaktionskoordination verteilter Ressourcen unverzichtbar. Ebenso ist die Interoperabilität mit asynchronen, messageorientierter Middleware eine weitere wichtige Anforderung.
Die Zusammenarbeit von Komponenten mit der bestehenden Ressourcenverwaltung ist nicht trivial. Um diese Aufgabe richtig zu machen, ist eine sehr gute Werkzeugunterstützung wichtig. Ein entsprechender Werkzeugsatz muss vor allem die Abbildung der Komponenten auf die bestehende Umgebung unterstützen. Dadurch können die Komponentenentwickler weit gehend von der mühsamen und aufwändigen Plattformintegration in verteilten, objektorientierten Umgebungen (Plumbing Code) befreit werden. Ebenso sollten die Werkzeuge verschiedene Rollen im Softwareentwicklungsprozess unterstützen. Programmierer sollten sich auf die Entwicklung der Business-Logik, andere Teammitglieder können sich auf die Abbildung der Komponenten auf die bestehende Infrastruktur konzentrieren. Andere übernehmen Aufgaben wie Analyse & Design, Deployment, System Management; rollenbasierte Entwicklung unterstützt ein paralleles Vorgehen mit effizientem Transfer von Artefakten zwischen Spezialisten, sodass Kostenüberwachung, Time-to-Market und Qualität verbessert werden. Zur Administration und Verwaltung der Applikationen, der Server und der Ressourcen wird eine Managementumgebung benötigt. Die einzelnen Komponenten einer Lösung müssen definiert, konfiguriert, installiert, ausgeliefert, in Betrieb gesetzt und überwacht werden können.

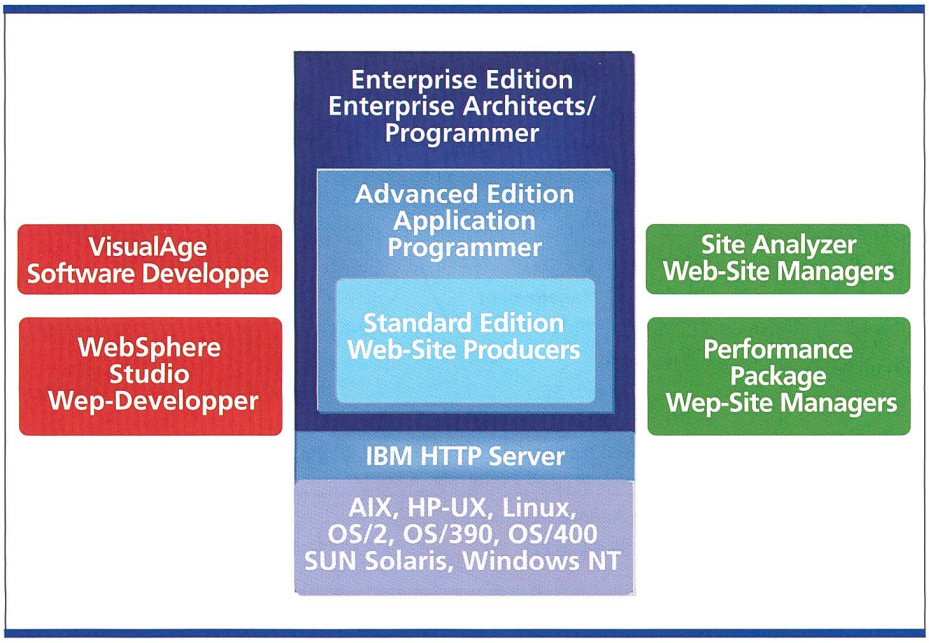

Bild 3. Die IBM-WebSphere-Familie.

All diese Anforderungen an eine reife Komponentenlösung haben Swisscom zu WebSphere geführt. Im Engagement Project «Ramp-up WebSphere» konnten verschiedene der erwähnten Aspekte mit gutem Erfolg an einer Pilotapplikation untersucht werden. Im Beitrag wird das Produkt WebSphere vorgestellt und die Evolution einer Anwendung von einer einfachen Website zu einer hochkomplexen E-Business-Webapplikation skizziert. Im Weitern wird zum Abschluss ein Überblick über die WebSphere Enterprise Edition, inbesondere über die Eigenschaften des Component Broker gegeben.

Wiederverwendbarkeit und Komponenten

Seit Jahren wird immer wieder Wiederverwendbarkeit von Softwareprodukten propagiert. Dabei hat Wiederverwendbarkeit verschiedene Facetten wie Quellen, Verfahren und Zeitpunkt im Lifecycle einer Applikation. Quellen reichen von individuellen Zeilen an Codes über Libraries, Frameworks bis zu Komponenten. Je nach Quelle sind dann auch die Verfahren der Wiederverwendung unterschiedlich; so kann bei der Verwendung von individuellen Zeilen an Codes typischerweise das Verfahren Copy & Paste, bei Frameworks die Vererbung, Polymorphismus und Instanzierung und bei Komponenten die Komposition von Shared Services angewendet werden. Ebenso wirken sich die Formen in verschiedene Phasen des Lebenszyklus einer Anwendung aus und haben somit auch unterschiedliche Bedeutung. Tabelle 1 gibt dazu einen Überblick.
Die vielversprechenste Form der Wiederverwendbarkeit geht wohl von Komponenten aus. Sie haben bei richtigem Design die geeignete Granularität, umfassen im Normalfall einen abgeschlossenen Business-Domain und können sowohl aus Sicht Analyse & Design als auch aus Sicht Deployment und Runtime eigenständige, logische Einheiten sein.
Komponenten aus Sicht Analyse & Design erlauben die Wiederverwendbarkeit in der Analyse-&-Design-Phase. Aus Sicht Deployment sollen neue Applikationen oder Komponenten auf existierenden Komponenten durch Servicedelegation aufgebaut werden können. Sie stellen Shared Services zur Verfügung und erlauben einen schrittweisen Aufbau eines ganzen Applikationsverbundes mit klar definierten Serviceschnittstellen (Bild 1).

Definition von Komponenten

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties. (Szyperski and Pfister, 1997)
Das Ziel einer Komponente kann nicht ausschliesslich die *Wiederverwendbarkeit* sein, da sie sonst beliebig aufwändig, schwerfällig und schwer einzusetzen ist. Mit steigender *Generizität* wird eine Komponente natürlich robuster. Ein Opti-

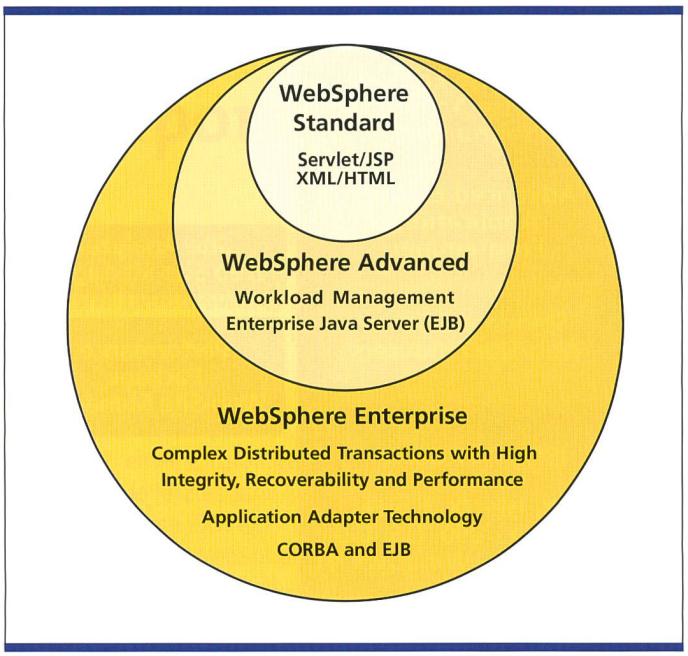

Bild 4. WebSphere Editions.

mun liegt jedoch irgendwo dazwischen, denn es gilt folgende Maxime (Bild 1): «Maximizing reuse minimizes use.»

Nutzen der Komponententechnologie

Die komponentenbasierten Konzepte und Technologien sind inzwischen reif und können erfolgreich eingesetzt werden. Typische Vertreter von Komponenten sind Modelle wie JavaBeans, Enterprise JavaBeans (EJBs) und CORBA-Komponenten (CCM). Ohne stichhaltige wirtschaftliche Argumente wird jedoch kaum jemand eine neue Technologie forcieren und einsetzen. Mit Hilfe der Komponententechnologie wird jedoch ein klar definierter, objektiver Nutzen angestrebt:

– Durchgängigkeit durch Identifikation der Business-Objekte
  Indem man die Business-Logik als Menge kooperierender Komponenten modelliert, kann eine Durchgängigkeit zwischen den Anforderungen des Kunden (Problem Domain), der *Implementation* des Softwaresystems und des *Deployment* (Installation und Inbetriebsetzung) der einzelnen Komponenten erreicht werden. Änderungen in Business-Anforderungen manifestieren unmittelbar Änderungen in den entsprechenden Komponenten. Indem man sofort erkennt, welche Komponenten zu modifizieren sind, kann *schneller* auf Änderungen der Business-Anforderungen *reagiert* werden.
– Flexibilität durch Kapselung der Business-Logik
  Komponenten *kapseln* die Business-Logik und werden in einer Umgebung ausgeführt, die von Implementationsdetails wie *Persistence, Konnektivität, Sicherheit* abstrahiert. Das Client-Programm, das über den Objektbus auf die Schnittstellen dieser Komponenten zugreift, muss diese Einzelheiten nicht beachten, das heisst, diese Einzelheiten können, ohne den Rest des Systems zu tangieren, jederzeit geändert werden. Die *Flexibilität* einer Applikation wird durch die Kapselung der Business-Logik stark unterstützt.
  1. Durch *Kapselung* der Komponenten werden Clients vor der *Komplexität* der *APIs* und der *Konnektivität*, sowie weiteren Aufgaben des Resource Managers, der im Hintergrund der Komponenten läuft, abgeschirmt. Clients greifen auf einfache Art und Weise auf Komponenten zu, wodurch eine hohe Entwicklungsproduktivität erreicht wird.
  2. Trotz unterschiedlicher Implementation und Integration in bestehende Systeme werden mehrere Systemteile mit Hilfe einer Komponentenschnittstelle (Facade), das heisst, eines einheitlichen externen Kontrakts, *gekapselt* (Wrapping) und für zugreifende Clients gleichwertig gemacht. Diese Eigenschaft unterstützt die *Konsistenz* des Programmiermodells.
  3. Unternehmen mit vielen Systemen, Applikationen und kundenbezogenen Daten (eventuell durch Merger und Acquisition entstanden) können mit Hilfe einer durch Komposition bestehender Einzelkomponenten synthetisierten neuen Komponente von einer einheitlichen Business-Sicht auf das Unternehmen profitieren. Kompositionen von Komponenten können auf das neue Business-Modell abgebildet werden. Eigenschaften von Komponenten können jedoch auch durch Vererbung (Inheritence) von anderen/neuen Komponenten übernommen und durch weitere Eigenschaften ergänzt werden. Komponenten können bei der Ausführung mehrerer, unterschiedlicher Business-Prozesse gleichzeitig aktiv sein und mit ihren Services einer breiten Vielfalt von Clienten dienen. Dies sind Beispiele der *Wiederverwendbarkeit*, die ebenfalls zu erhöhter *Produktivität* und besserer *Konsistenz* führen.
– Investitionsschutz bestehender IT-Assets und neuer Investitionen
  Nicht nur *bestehende* IT-Assets können geschützt und mit Hilfe neuer Interfaces ohne grosse Änderungen in bestehende Systeme integriert werden, sondern auch die *neuen Investitionen* in Komponenten müssen laufend gesichert, geschützt und können mit Hilfe neuer Business-Modelle verbessert werden. Es gibt dabei mindestens *drei Aspekte des Investitionsschutzes* und jede davon ist kritisch:
  1. Die Möglichkeit, Komponenten *zwischen verschiedenen Plattformen zu verschieben,* erlaubt auf *vertikale Skalierungsbedürfnisse* zu reagieren. Diese Form der Portierung sollte jedoch *ohne* Entwicklungs- oder Programmieraufwand erfolgen und bloss ein *neues Deployment* auslösen. Diese Form der Portabilität schützt die Investitionen massgebend.
  2. Investitionsschutz impliziert die *standortunabhängige Erreichbarkeit* von Komponenten durch Clients. Diese Forderung umfasst sowohl Java Servlets in einer Webserverumgebung als auch traditionelle Desktop-Clients. Diese breite Verfügbarkeit ist eine wichtige Form des Investitionsschutzes, da dadurch Applikationen beim Auftreten neuer Formen von Clients oder Servern nicht neu implementiert werden müssen.
  3. *Interoperabilität zwischen Produkten verschiedener Anbieter* ist ein wichtiger Investitionsschutz, weil man dadurch nicht auf eine einzige lieferantenspezifische Umgebung festgelegt wird. Die Möglichkeit, verschiedene Lösungen zu mischen, garantiert

Wahlfreiheit und ermöglicht eine Evolution, die auf die spezifischen Bedürfnisse des Business abgestimmt ist.

Applikationsarchitektur für Komponenten

Zehn Jahre nach ihrer Einführung sind Client/Server-Anwendungen die dominante Applikationsarchitektur geworden. Mit diesem revolutionären Ansatz gelang es, die Rechenlast einer monolithischen Mainframeanwendung zwischen Client und Server zu verteilen. Client/Server-Architekturen entwickeln sich von der ursprünglichen *2-Tiers-* immer mehr zur *3-Tiers-Architektur.* Die Auswirkungen dieser Änderung werden bedeutender sein als der ursprüngliche Trend weg von monolitischen Mainframes zu Client/Server-Applikationen. Diese Bewegung wurde ursprünglich durch die Nachfrage nach Enterprise-Applikationen belebt. Heute wird dieser 3-Tiers-Trend vor allem durch die verteilten Objekte und Komponenten stark getrieben.

In den 80er-Jahren versuchte man die Nachfrage nach Mid-Range-Rechner durch die Einführung des Begriffs 3-Tiers zu beleben. *3-Tiers* stand für die *physikalische Aufteilung* einer Applikation zwischen Terminal (Tier 1), dem Kleinrechner (Tier 2) und dem Mainframe (Tier 3). Heute werden *Tiers* zur Beschreibung der *logischen Unterteilung* einer Applikation in *Client* und *Server* verwendet. Die Verteilung der Rechenlast ist im Client/Server-Konzept zentral; hinzu kommt jedoch auch der Designentscheid, wem die entsprechende Rechenlast zugeteilt werden. Der Begriff *Tier* erlaubt, grundsätzliche Ansätze der Architekturen von Applikationen zu beschreiben (Bild 2):

- 2-Tiers-Architektur: Aufteilen der Rechenlast auf zwei Schultern: Die Mehrheit der Applikationslogik läuft auf dem Client; der Client schickt SQL-Anfragen an eine serverbasierte Datenbank. Da mit dieser Architektur grosse Teile der Applikation auf dem Client laufen, wird sie auch als *Fat-Client* bezeichnet.
- 3-Tiers-Architektur: Verteilen der Rechenlast zwischen dem *Client* auf dem die grafische Interface-Logik (GUI) läuft, dem *Application Server,* auf dem die *Business-Logik* läuft und der Datenbank und/oder der/den Legacy-Applikation(en). Da mit dieser Architektur die Applikationslogik auf dem Server angesiedelt ist, wird sie auch als *Fat Server* bzw. *Thin Client* bezeichnet.

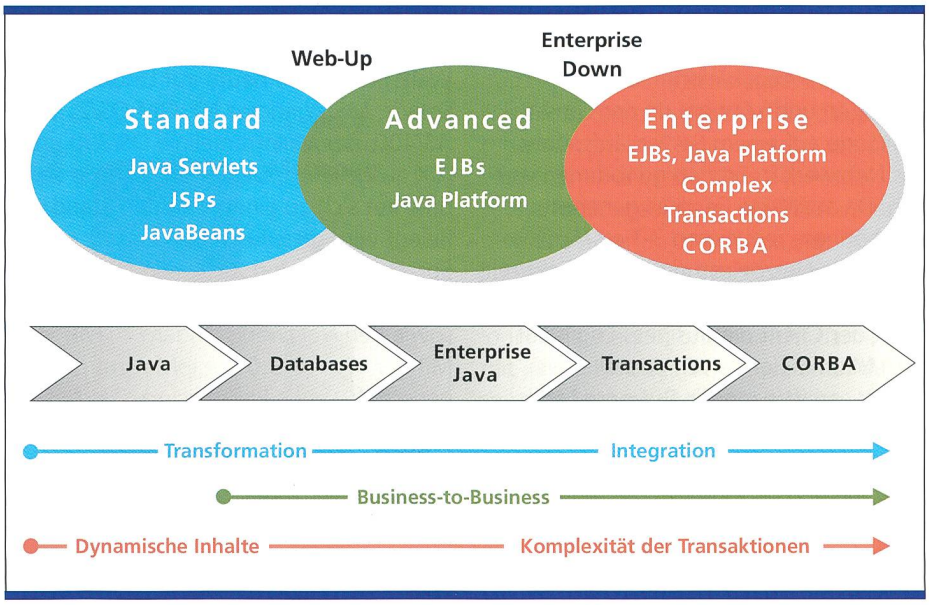

Bild 5. Positionierung der verschiedenen WebSphere-Runtime-Umgebungen.

In 2-Tiers-Client/Server-Applikationen ist die Applikationslogik entweder im User Interface, in der Datenbank oder auf beide Tiers verteilt. Die GUI läuft auf dem Client, sendet Systemaufrufe, SQL und HTTP-Befehle über ein Netzwerk an den Server, der Server führt diese Aufrufe aus und gibt ein entsprechendes Resultat zurück. Um auf Daten zuzugreifen, muss der Client wissen, wie die Daten im Server organisiert sind. Eine Variation des 2-Tiers-Ansatzes verwendet so genannte *Stored Procedures,* um einen Teil der Ausführungslogik auf die Datenbank auszulagern. Einfachheit ist der wichtigste Erfolgsfaktor des 2-Tiers-Ansatzes. Dieser Ansatz ermöglicht eine effiziente Entwicklung und eignet sich vor allem für kleine Anwendungen, so genannte Abteilungslösungen. Will man jedoch kritische, unternehmensweite Client/Server-Applikationen oder E-Commerce-Anwendungen realisieren, eignet sich dieser Ansatz nicht, da er nicht skaliert.

In 3-Tiers-Applikationen unterstützt der Client das *User Interface* und interagiert mit dem Server durch *Remote Services* oder Methodenaufrufe. Die Applikationslogik lebt im *Middle Tier* und läuft auf einem oder mehreren Servers. Da einzig mit 3-Tiers skaliert und damit die Anforderungen von grossen Internet-&-Intranet-Client/Server-Applikationen er-

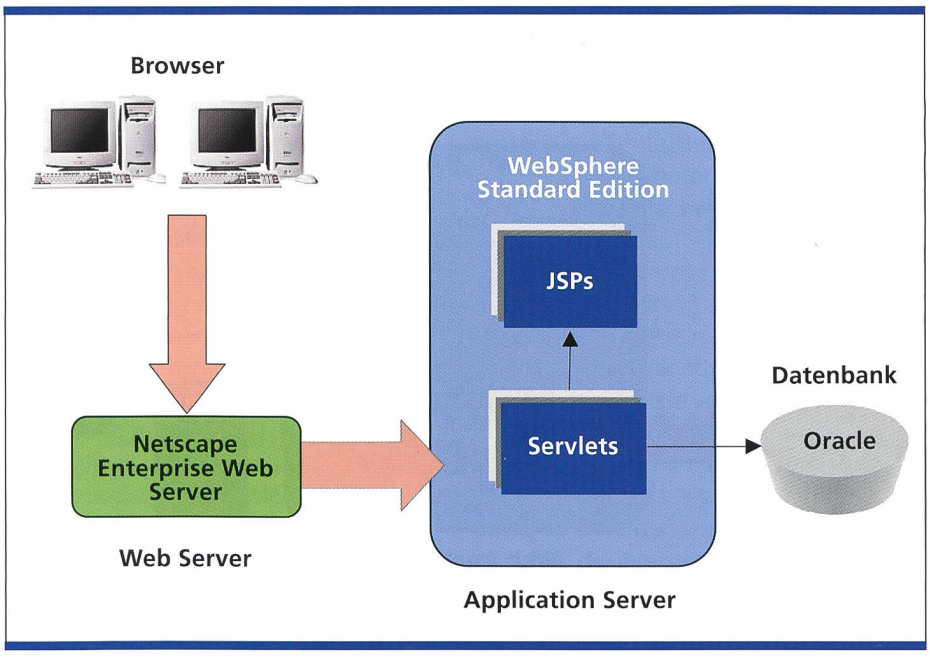

Bild 6. WebSphere Standard Edition mit Servlets, JSPs und Oracle DB.

füllt werden können, gewinnt diese Architekturvariante an Bedeutung. Zudem sind 3-Tiers-Applikationen einfacher zu verwalten und können, da der Grossteil des Code auf einem Server läuft, über das Netzwerk in Betrieb genommen werden. Da abstrakte Services generiert werden können, reduzieren 3-Tiers-Applikationen die Netzbelastung. Statt direkt mit der Datenbank zu kommunizieren, benutzt der Client die Business-Logik auf dem Middle-Tier-Server. Anstelle des Clients, greift dann die Business-Logik des Middle Tier auf die Daten der Datenbank zu. 3-Tiers ersetzt damit die vielen SQL Queries und Updates des 2-Tiers durch ein paar wenige abstrakte Serveraufrufe. Da dem Client keine Datenbankschemas bekannt sind und eine feinkörnige Authorisierung auf dem Server möglich ist, kann die Sicherheit besser und gezielter unterstützt werden. Aus Tabelle 2 wird schnell klar, dass die 3-Tiers-Architektur der 2-Tiers-Architektur bei Client/Server-Applikationen weit überlegen ist.

Das Produkt WebSphere von IBM

IBM positioniert seine WebSphere-Familie (Bild 3) als einfach zu implementierende, skalierbare Lösung für Kunden, die E-Business-Applikationen bauen, verwalten und in einer stabilen und sicheren Umgebung ausführen wollen (*develop – manage – run*). WebSphere ist ein wichtiger Bestandteil in IBMs Application Framework für E-Business. Das Produkt wurde zur Unterstützung von Kunden, Entwicklern und anderen Serviceprovidern mit transaktionellen, weborientierten Applikationen entworfen. WebSphere umfasst eine breite Palette von integrierten Softwareprodukten, die dem Anwender erlauben, einfache Websites zu publizieren, aber auch hochkomplexe E-Business Webapplikationen zu entwickeln. Nicht jede Organisation ist in der Lage, auf derselben Stufe ins E-Business einzusteigen. Aus diesem Grund hat IBM drei verschiedene *Editionen* des Produktes WebSphere auf den Markt gebracht. Diese Editionen sind so positioniert, dass deren Applikationserver nebeneinander koexistieren können. Verschiedene Kunden profitieren von der Kompatibilität dieser Server, indem sie zwei oder drei dieser Server für spezifische Situationen kombiniert einsetzen können.

Der WebSphere-Application Server und verwandte Produkte können auf verschiedenen Betriebssystemen und Hardwareplattformen ausgeführt werden. WebSphere wird heute auf Linux, Sun Solaris, MS Windows NT, IBM OS/2, IBM AS/400, der Mainframelinie OS/390 und AIX ausgeliefert. Die Produktfamilie verpflichtet sich zu einem offenen Standard. Belegt wird dies durch den Einsatz der Sprachen Java und XML, den CORBA-Standard der OMG für verteilte Systeme sowie den Netzwerkstandard TCP/IP.

Mit der Version 3.0 hat IBM nicht nur Eigenschaften wie Performance, Verfügbarkeit, Konnektivität und Managementmöglichkeiten bei den Applikationsservern stark verbessert, sondern hat seine Palette an Entwicklungswerkzeugen gut abgerundet. So wurde beispielsweise die Java-Entwicklung mit der Version 3 von *VisualAge for Java* stark verbessert. Das Produkt enthält jetzt eine einfachere Debuggingumgebung sowie Wizards zur Unterstützung bei der Erzeugung von Applets, Servlets und Enterprise JavaBeans (EJB). Für grössere Unternehmen sind die Konnektoren und vor allem die Adaptoren zu verschiedenen ERPs wie SAP oder transaktionelle Systeme wie IBMs CICS oder IMS von grosser Bedeutung.

Das Produkt *WebSphere* von IBM wird in drei verschiedenen Editionen angeboten, die WebSphere *Standard Edition,* die WebSphere *Advanced Edition* und die WebSphere *Enterprise Edition*.

Es ist hilfreich, die drei Editionen nach *Anwendersfokus* und nach *Funktionalität* zu unterscheiden.

- Standard Edition (WSE): Die *Standard Edition* richtet sich hauptsächlich an Websiteproduzenten, die dynamische Präsentationen mit *Logik* und *Inhalt* versorgen wollen, und unterstützt dabei nur die *grundlegenden* Webbedürfnisse. Diese Bedürfnisse umfassen IBMs hochleistungsfähigen HTTP-Server, eine mit SSL erweiterte Version des Apacheservers. Neben dem Webserver umfasst die Standard Edition eine effiziente, konfigurierbare Engine für Java Servlets sowie Java Server Pages (JSPs). Servlets können mit Hilfe eines JDBC Pool effizient auf Datenbanken zugreifen. Ein XML-Parser und -Generator sowie LDAP-Integration und Personalisierung runden das Produkt ab.
- Advanced Edition (WAE): Die *Advanced Edition* ist ein Superset der Standard Edition (Bild 4). Sie richtet sich an Applikationsentwickler, die *Business-Logik* für neue Applikationen auf der Basis von *Enterprise JavaBeans* (EJBs) erstellen. Das verwendete Objektmodell erlaubt die Trennung der Business-Logik von Eigenschaften wie Sicherheit und transaktioneller Integrität. Die Dienstqualität (QoS) einer Applikation kann weit gehend nach Bedarf konfiguriert und verwaltet werden. Diese Edition erlaubt Lastverteilung und Umgehung bei Ausfällen.
- Enterprise Edition (WEE): Die Enterprise Edition wiederum ist eine Erweiterung

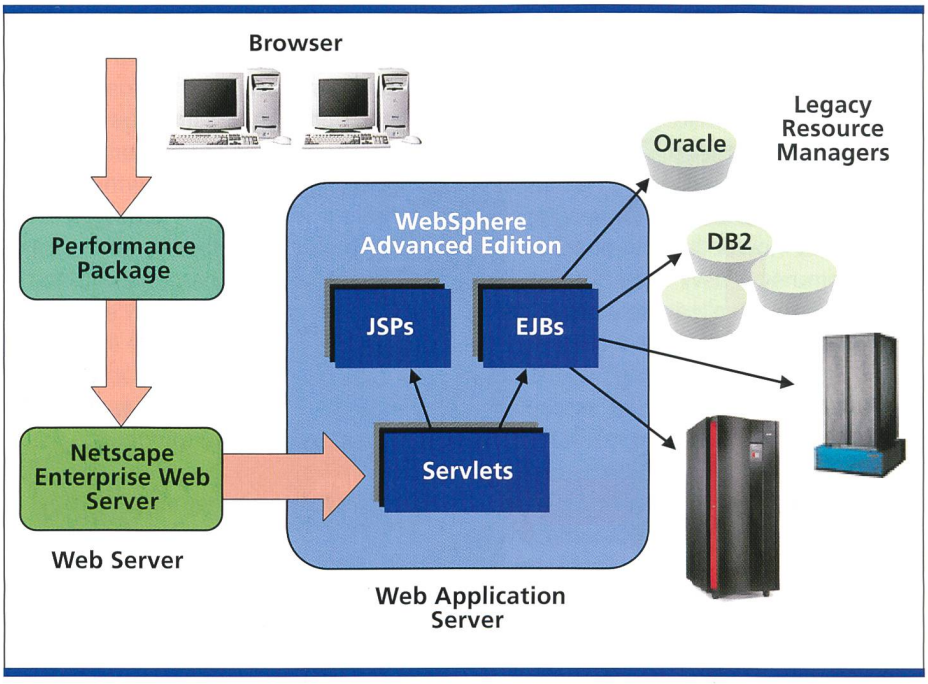

Bild 7. WebSphere Standard Edition, Advanced Edition mit Performance Package.

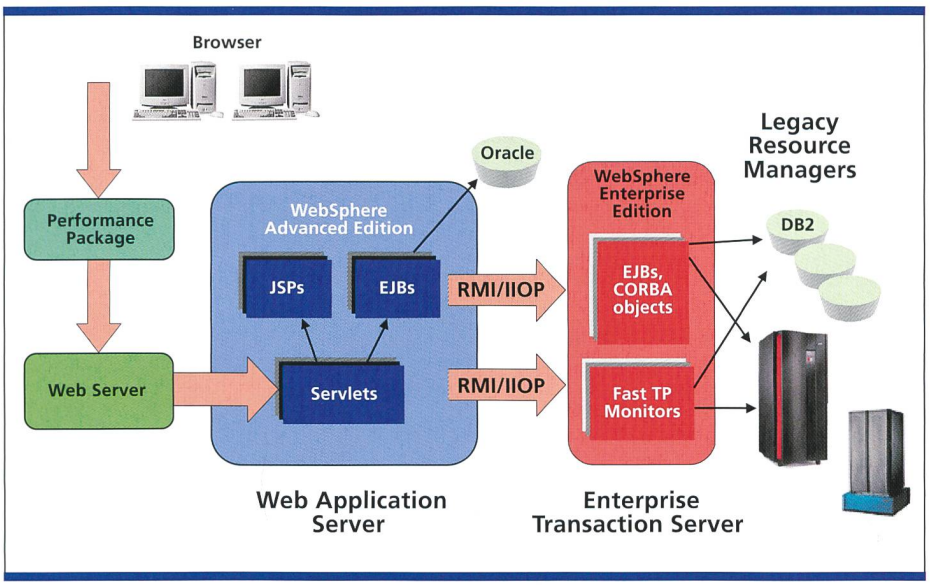

Bild 8. WebSphere Enterprise Edition und Component Broker.

der Advanced Edition (Bild 4) und fügt zwei neue Elemente, den *Component Broker* und *TxSeries,* hinzu. Sie ermöglicht die Integration sowohl *neuer* als auch *existierender* Business Logik (Legacy) mit *heterogenen Systemen* und lässt die Wahl zwischen verschiedenen *Programmiermodellen* offen. Der Component Broker führt die Unterstützung für verteilte *CORBA-Objekte* als auch *EJBs* ein. So kann beispielsweise der Component Broker die Transaktionskontrolle von EJBs, die auf DB2, Oracle, CICS, IMS und MQSeries abgebildet sind, in einer einzigen Arbeitsumgebung wahrnehmen. TxSeries offeriert dem Entwickler als Alternative zum CB die Wahl zwischen Distributed CICS und Encina.

Bild 5 dokumentiert die Rolle der verschiedenen WebSphere-Editionen auf dem Markt. Die Standard Edition unterstützt die fundamentalen Mechanismen, die im E-Business benötigt werden. Sie unterstützt alles, was zum grundsätzlichen Webeinsatz mit Einschluss der Generierung von dynamischen Web Pages und dem Zugang zu relationalen Datenbanken mit Java Servlets gehört. Wird mehr verlangt, bietet die Advanced Edition die Möglichkeiten einer verteilten Umgebung mit EJBs. Workload Management hilft der Verbesserung der Skalierbarkeit und der Fehlertoleranz. Die Advanced Edition fördert somit die Flexibilität und erhöht das Potenzial der Wiederverwendbarkeit bestehender Assets. Grössere Unternehmen werden sicher auf eine volle Integration ihrer existerenden Assets setzen und versuchen, diese zu modernen webbasierten Applikationen zu erweitern. Bei diesem Ansatz wird hohe Skalierbarkeit mit der Wiederverwendbarkeit existierender Applikationen, Datenbanken, Transaktionsumgebungen, fortschrittlichen CORBA Services, Securityabbildung (Mapping) und der anspruchsvollen transaktionellen Koordination verbunden. Ebenso wird in diesem Umfeld mehr und mehr messageorientierte Middleware integriert.

Einsatz von WebSphere

Wie können nun diese WebSphere-Editionen skalierbar eingesetzt werden? An einem Beispiel wird die Evolution einer einfachen Website zu einer hoch komplexen E-Business-Webapplikation skizziert.

Eine fiktives Unternehmen (FU) hat mit der Publikation einer statischen Website eine erste Webpräsenz erreicht. Darin werden die Firma, ihre Produkte zusammen mit Kontaktinformationen beschrieben und ein einfacher *E-Mail Service* für Kunden aufgebaut.

Um ein effizienteres E-Business zu erreichen, beschliesst FU, Kundenprofile zu erfassen und diese in einer neuen Oracle-Datenbank abzulegen. FU baut diesen Teil mit Hilfe von *WebSphere Studio* auf. Gewöhnliche *JavaBeans* werden aufgebaut, welche die Tabellen in der Oracle-Datenbank lesen können. Diese JavaBeans werden von *Java Servlets* aufgerufen, die ihrerseits auf *WebSphere Standard Edition* ausgeführt werden. Weiter werden *Java Server Pages* entwickelt, welche die dynamische Präsentation von HTML unterstützen (Bild 6).

Bereits in diesem ersten Projekt wird ein bedeutender Teil des benötigten Code durch WebSphere-Tools erzeugt.

Nachdem mit gewissem Erfolg die erweiterte Website in Betrieb gesetzt wurde, wird schnell erkannt, dass der eingesetzte *Netscape Enterprise Web Server* bereits am Rand seiner Kapazität läuft. FU entscheidet, *das WebSphere Performance Package* in die bestehende Topologie einzubauen, und erreicht damit mit jedem hinzugefügten weiteren Webserver im bestehenden Cluster eine nahezu lineare Skalierbarkeit.

Obwohl weitere Maschinen problemlos hinzugefügt werden können und die Anwendung effektiv skaliert, scheinen nicht alle Maschinen bis an ihre Grenzen belastbar. FU entscheidet nun, seine Software von der *Standard-* zur *Advanced Edition* aufzurüsten, um damit mehrere *Servlets* und *JSP Clones* auf ihren SMPs einzuführen. Mit dieser Massnahme werden eine höhere Stufe an Skalierbarkeit und bessere Auslastung der Hardware erreicht. Die *Advanced Edition* bringt FU zudem Administrationswerkzeuge, die den laufenden Unterhalt und die Verwaltung der Modelle und Clones vereinfachen.

Die entwickelte Lösung hat nun die Fähigkeit, auf eine erhöhte Anzahl von Anfragen zu reagieren. FU erkennt die Notwendigkeit, auf ändernde Geschäftsbedingungen reagieren zu können. Um beispielsweise der Kundschaft selektive Informationsabfragen zu jeder Tageszeit ohne Unterstützung eines Call Center zu ermöglichen, müssen die Dienstleistungen am Web erweitert werden. Diese Anforderungen verlangen zusätzliche Securitymassnahmen. Zudem muss die Softwareimplementation auf Business-Regeln abgebildet werden können. Die Business-Regeln müssen in der Implementation leicht identifiziert werden können, sodass Änderungen am Business-Modell effizient umgesetzt werden können. FU beginnt nun mit der Implementierung ihrer Kerngeschäftsprozesse und nutzt dazu Enterprise JavaBeans (EJBs). Diese Implementierung *modelliert* somit die *Policy* des Business; wenn diese ändert, können die Logik und die Struktur der EJBs entsprechend angepasst werden (Bild 7). Beim Einsatz von EJBs wird *VisualAge for Java* als Verbindungstechnologie zu den *Legacysystemen* eingesetzt. *VisualAge for Java* wird ebenso zur Erzeugung von Codes und zum interaktiven Test der neuen Logik in den EJBs eingesetzt.

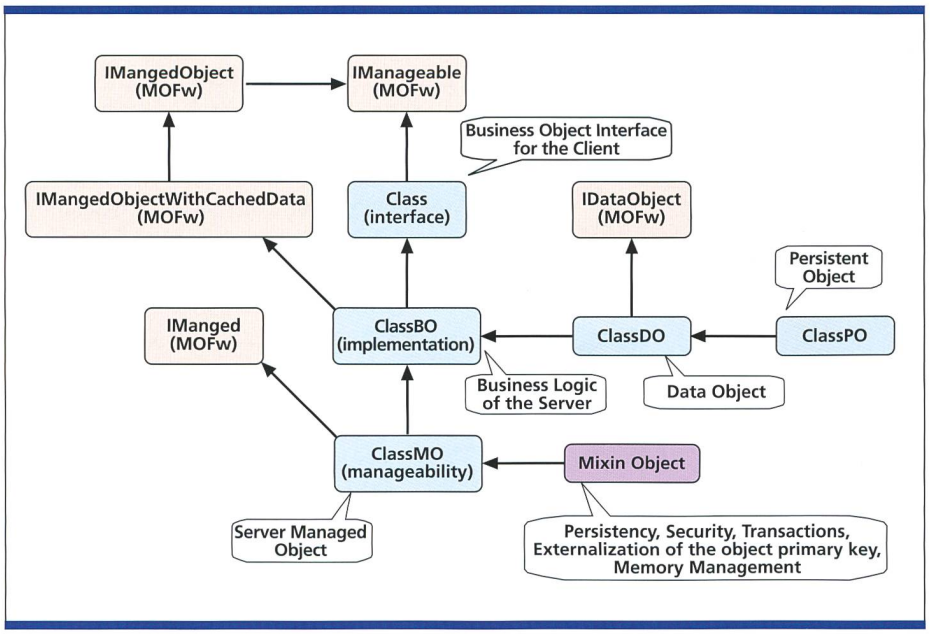

Bild 9. Ausschnitt aus Managed Object Framework (MOFw).

Obwohl sich bei FU der Erfolg im E-Business einstellt, wird auch erkannt, dass speziell der Einsatz der Komponententechnologie mehrere *Legacysysteme* involviert. Da Kunden nun ihre individuellen Profile und Präferenzen eigenständig verwalten, wird klar, dass entsprechende Updates der Legacy-Systeme gemeinsam synchronisiert und mit hoher *Integrität* und transaktionellen ACID-Eigenschaften zu erfolgen haben (ACID: *Atomic, Consistent, Isolation, Durability*). Um die höhere Dienstqualität (QoS) zu ermöglichen, entscheidet sich FU, die *Enterprise-Edition*-Technologie in Form des *Component Broker* einzusetzen (Bild 8). Mit seinen *Applikationadaptern* garantiert der Component Broker bei Updates von unterschiedlichen Systemen ein hohes Mass an *Integrität*. Eine optimistische Lockingstrategie ermöglicht weiter eine hohe *Concurrency*. Ausgewählte EJBs der Advanced Edition werden ohne oder mit geringen Änderungen mit höherer Dienstqualität in der Enterpriseumgebung neu in Betrieb genommen. *VisualAge Component Development* unterstützt neu die Arbeit eines ganzen Teams von Entwicklern.

FU erfreut sich nun seiner grossartigen Möglichkeiten im E-Business und wächst kontinuierlich in einer innovativen IT-Umgebung, die via Web leicht zugänglich, standardbasiert, hoch zuverlässig, serverzentriert, verwaltbar und leistungsstark ist. FU macht weitere Fortschritte dank der *WebSphere-Site-Analyzer-Technologie*, mit deren Hilfe die Nutzmuster der Sites untersucht werden können und durch entsprechende Verbesserungen der Services die Kundenzufriedenheit weiter verbessert werden kann. Weiter sind Verbindungen zu Business-Partnern möglich, was die Supply Chain vergrössert. MQ-Series spielt zusammen mit der Enterprise Edition eine wichtige Rolle bei der Realisierung von *transaktionellem Messaging*. Supply-Chain-CORBA-Schnittstellen werden durch die Einbindung von Third-Party ORBs mit Hilfe der Enterprise Edition ebenfalls erschlossen.

WebSphere Enterprise Edition (WEE)

WebSphere Enterprise Edition wurde als Erweiterung (Superset) der *Advanced Edition,* die Advanced Edition ihrerseits als Erweiterung der *Standard Edition* bezeichnet (Bild 3). Alle Möglichkeiten der Standard- und der Advanced Edition werden somit automatisch von der Enterprise Edition übernommen und unterstützt; die Enterprise Edition erweitert die Advanced Edition zudem mit dem *Component Broker* und *TXSeries*. Das strategische Herzstück der Enterprise Edition ist der *Component Broker (CB).* Dieses interessante Teilprodukt wird im folgenden Kapitel kurz beschrieben.

Der Component Broker

Der Component Broker unterstützt eine *integrierte, vollskalierbare, sichere* und *administrierbare* Application-Server-Umgebung für *CORBA-Objekte* und *Enterprise JavaBeans (EJB).* Die Spezifikationen der Enterprise JavaBeans von SUN gehen auf die innovative Architektur des Component Broker zurück; die letzten und bedeutendsten Beiträge an das *CORBA Component Modell* (CCM) der OMG haben ebenfalls ihre Wurzeln im Component Broker. Vor der Lancierung des Industriestandards EJB durch SUN hatte IBM bereits mehrere Jahre in den CB investiert. Der containerbasierte Ansatz im CB diente als Vorbild für EJB. Kurz nach Abschluss der Spezifikation der EJB war IBM bereits in der Lage, als Marktleaderin EJBs auf der Basis des Component Brokers zu demonstrieren. Das Produkt Component Broker offeriert heute auf dem Markt die wohl vollständigste Implementierung von CORBA 2.1 und mit WebSphere Enterprise Edition Version 4.5 (voraussichtlich 2Q 01) soll das Branding CORBA 2.3 angestrebt werden. Der Application Server der Advanced Edition wird in der Enterprise Edition durch den Component Broker ersetzt bzw. ergänzt, und liefert damit die Dienstqualität (QoS) einer Enterprise-Applikation. Die zusätzliche Funktionalität der Enterprise Edition wird weit gehend durch den Component Broker geliefert. Tabelle 3 stellt die wichtigsten Erweiterungen der *Enterprise Edition* gegenüber der *Advanced Edition* zusammen.

Im Folgenden werden kurz die wichtigsten – durch den Component Broker Plattform für Enterprise – Komponenten kurz beschrieben:

- CORBA-Unterstützung: Eine der wichtigsten Erweiterungen von WEE gegenüber WAE besteht in der erstklassigen CORBA-Unterstützung des Component Broker; *CORBA-Komponenten* können in der Component-Broker-Umgebung importiert, erzeugt, installiert und in Betrieb genommen werden (Deployment). Mit Hilfe geeigneter Werkzeuge gilt: *develop once, deploy many.* Mit einem allgemeinen Programmiermodell ist es zudem möglich, ohne Änderung der Business-Logik CORBA-Objekte beliebig zwischen verschiedenen Betriebssystemen zu verschieben. Der Component Broker bietet die wohl reichste und vollständigste Palette an CORBA-Services. In der CORBA-Realität gibt es sowohl in der Breite (Anzahl Services) als auch in der Tiefe (Vollständigkeit der Implementierung) einen grossen Unterschied zwischen den existierenden Implementationen. IBMs Component Broker unterstützt als Ganzes oder in Teilen zehn der CORBA-Services und ist mit Abstand

jeder anderen Implementation überlegen. CORBA ermöglicht ein Mix & Match von nicht-Java-basierenden Lösungen (C++, COBOL usw.), den Einbezug von EJBs und eine Auswahl der Implementationssprache bei Clients und Servern.

– Enterprise-Qualität: Ein Managable Object ist *ortsunabhängig* (location transparent), hat eine eigene *Identität*, sein Zustand kann *persistent* gesichert werden, kann *aktiviert* und *passiviert* werden, seine Zustandsänderungen können korrekt auf die *transaktionelle Semantik* mit flexibler Konfiguration der *Concurrency Control* abgebildet werden, sein *Lifecycle* kann von seiner Erzeugung bis zu seiner Zerstörung verwaltet werden, sein *Zugriff* kann auf einen Kreis *authentisierter* und *autorisierter* Benutzer beschränkt werden und *Security Credentials* können *delegiert* werden, ihnen kann ein *Name* zugeordnet werden, ein Objekt kann in das Resultat einer *dynamischen Query* so einbezogen werden, dass sein interner Zustand ein vorgegebenes *Suchprädikat* erfüllen kann, und es kann in *Ereignissen* und *Notifikationen* über ein verteiltes System teilhaben. Alle diese Möglichkeiten haben nur am Rand mit der Business-Logik zu tun, sind jedoch in einer verteilten Umgebung äusserst schwierig zu realisieren (Plumbing Code). Diese Realisierung wird mit einer reichen Ausstattung an *Frameworks*, mächtigen Werkzeugen und der *Container-Technologie* erreicht. Somit kann sich der Entwickler voll auf die Business-Logik konzentrieren und darf sich auf den Komfort und die Garantie der Services der Enterprise-Runtime-Umgebung verlassen.
– Sprachneutralität: Dank CORBA besteht zur Entwicklung von Client und Server kein Zwang, ausschliesslich eine Sprache, wie beispielsweise Java, einzusetzen. Diese Freiheit erhöht die Flexibilität der Anwender, da sie existierende Skills und Programmierkenntnisse in den Aufbau neuer Komponenten einsetzen können. Dank einer Abbildung COM-to-CORBA werden auch ActiveX Clients unterstützt.
– Transaktionelle Koordination: Der Component Brokers ist ein *Transaktionsmonitor*. Während die Advanced Edition bloss die Koordination von relationalen Datenbanken unterstützt, koordiniert der Component Broker neben verschiedenen Datenbanktypen zusätzlich prozedurale Umgebungen und MQSeries mit einem verteilten Two-Phase-Commit. «Out-of-the-Box» Pooling ist nicht bloss auf relationale Datenbanken beschränkt. Die Security kann vom Objektbereich mit der Abbildung der Credentials zurück auf ein Tiers-3-System delegiert werden. Alle diese Eigenschaften funktionieren dank der im *Component Broker* realisierten *Adaptertechnologie*.
– Applikation Adapter: Die *Application Adapters* unterstützen eine tiefe und nahezu vollständige Tiers-3-Integration. Sie schliessen die Koordination der Transaktionen prozeduraler Umgebungen wie CICS und IMS, Messagingsysteme wie MQSeries und relationale Datenbanken wie Oracle, DB2 und Informix in einer verteilten Umgebung ein. Dies ermöglicht jedem einzelnen dieser Resource Manager, an einer gemeinsamen Transaktion teilzunehmen, sodass *alles* oder *nichts* ausgeführt wird *(all or nothing)* und dabei volle transaktionelle Integrität garantiert bleibt. Es erfolgen entweder alle oder keine Updates der betroffenen Resource Managers. Zusätzlich ermöglichen die Adapter die Abbildung von *Credentials* auf die entsprechenden Tiers-3-Umgebungen und unterstützen damit eine vollständige Integration ganzer objektorientierter Dienste, was die Konkurrenz zurzeit nicht bieten kann. So unterstützen Adapter beispielsweise *Query Push-down,* womit anstelle einer verallgemeinerten objektbasierten Lösung in Tiers-2 ein optimaler Query-Auflösungsservice erst in Tiers-3, also direkt an der Database, erfolgt. Adaptertechnologie beschränkt sich normalerweise auf die Möglichkeit, über Java-Client-APIs darunter liegende Backend-Systeme anzusprechen (CICS Universal Client oder MQSeries Client for Java). Dieser Ansatz unterstützt eher eine *Konnektor-Lösung* (die Möglichkeit, sich an Tiers-3 anzubinden), jedoch keine Adapter-Lösung, da ein Adapter eine vollständige Update-Integrität mit komplexem *Commit Scope* über alle teilnehmenden Systeme garantieren sollte. Einfache Konnektoren hingegen können keine derartigen Garantien bieten.
– Managed Object Framework (MOF): Das *Managed Object Framework* (MOF) des Component Broker realisiert die Kernservices der Laufzeitumgebung (Ausschnitt dazu in Bild 9). Das Framework basiert auf einer offenen CORBA-

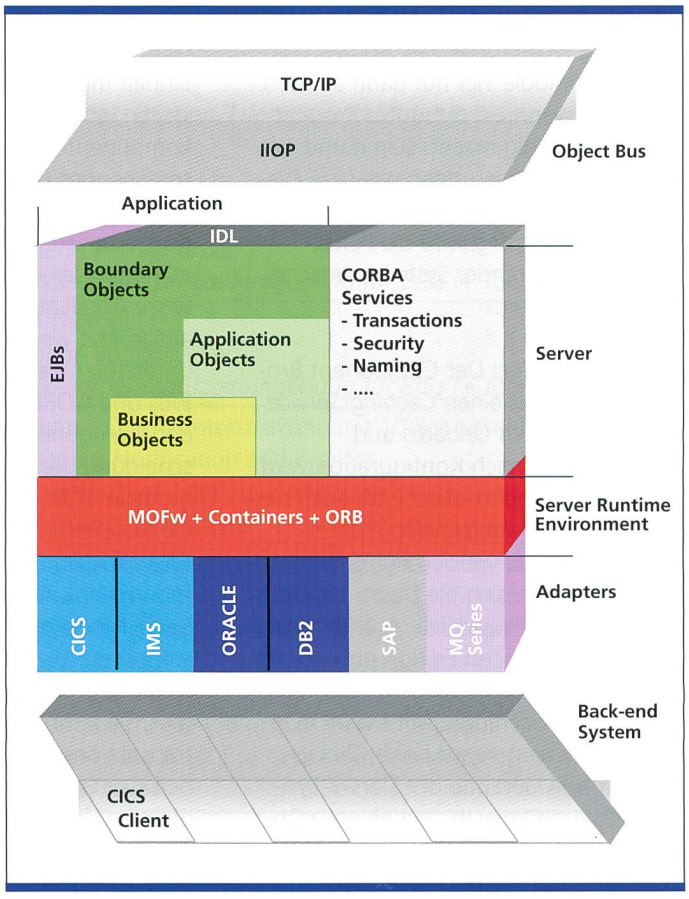

Bild 10. Architektur Component Broker.

Erweiterung und unterstützt neben CORBA-Komponenten auch EJBs. Indem das Framework sowohl die SW-Architektur als auch die Infrastruktur Out-of-Box unterstützt, reduziert es den Aufwand im *Softwaredesign* und in der *Entwicklung* und beschleunigt dadurch die *Time-to-Market*. Ebenso wird der Unterhalt der Applikation selber stark minimiert. Das Framework ist eine *Best-of-Practice*-Implementation; Objekte delegieren mit Hilfe des MOF Aufgaben an die Services der Laufzeitumgebung.

– Query Service: Mit Hilfe des CORBA Query Service im Component Broker kann eine Query auf Objektebene abgesetzt werden. Dieser Service unterstützt eine objektorientierte Version von SQL und unterscheidet sind grundsätzlich von JDBC: statt Kolonnen einer Tabelle werden Attribute von Objekten abgesucht. Zudem können Operationen auf Objekte als Teil einer Query ausgeführt werden. Man ist weder auf Java als Client-Sprache noch auf relationale Datenbanken als Tiers-3-Umgebung beschränkt. Der angebotene Query Service übersetzt einen objektorientierten Query-Ausdruck in den zugrundliegenden Query-Mechanismus des entsprechenden Resource Manager im Tiers-3. Dieser Mechanismus unterstützt sodann eine effiziente Ausführung ausserhalb des Objektbereiches, sodass Objekte im Middle Tier nur dann aktiviert werden, wenn ein erzeugtes Resultat dies notwendig macht. Damit und mit weiteren Optimierungen wie *Lazy Evaluation, Plan Caching, Pre-Fetch* und *Throttled Fetching* wird die Performance existierender Systeme entscheidend verbessert.

– Caching Service: Der Component Broker unterstützt einen *Caching Service* mit verzögerten *Updates* und *Refreshes*. Je nach Konfiguration wird sowohl eine optimistische als auch pessimistische *Lockingstrategie* unterstützt; Updates werden nur dann ausgeführt, wenn sich die Daten der Komponenten auch wirklich verändert haben. Die Advanced Edition unterstützt sowohl «Option A Caching» als auch «Option C Caching». Mit einem «Option A Caching» wird *hohe Concurrency* ohne Garantie der Tiers-3-Synchronisation erreicht. Mit einem «Option C Caching» wird *hohe Integrität* zwischen dem Objektbereich und dem Tiers-3 auf Kosten einer reduzierten Concurrency garantiert. Der Component Broker hingegen unterstützt mit seiner *Optimistic Locking Policy* das Beste beider Welten: *hohe Concurrency ohne Verlust an Integrität.* Erreicht wird diese Eigenschaft mit einem *Private Cache* für jede einzelne Transaktion, das bei jedem Update jeweils konsultiert werden muss.

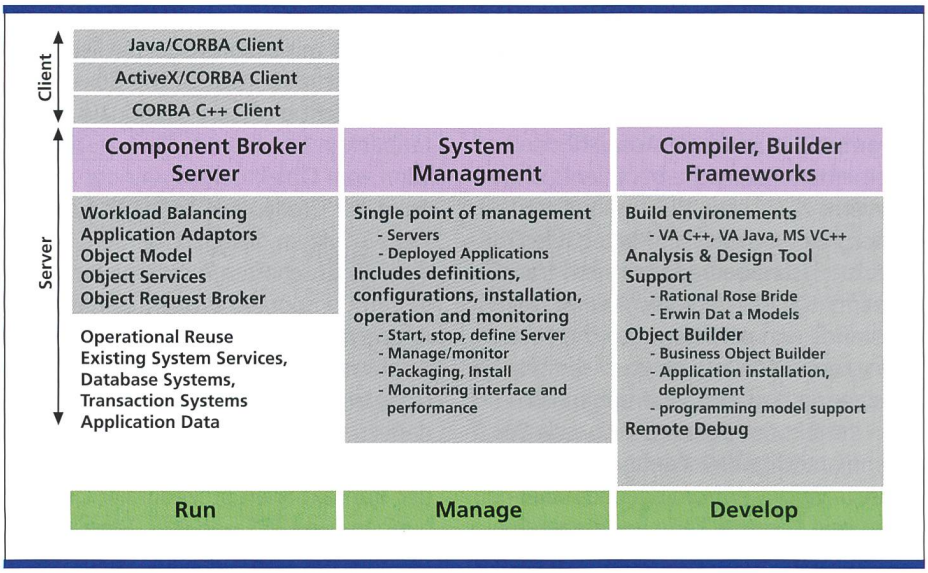

Bild 11. Drei Eckpunkte der Produktearchitektur.

– Session Service: Der CB unterstützt einen Sessionservice als Möglichkeit, zusammenhängende Aktivitäten, die nicht eine traditionelle, transaktionelle Einheit mit garantierter *Atomizität* darstellen, abzugrenzen. Dies unterstützt komponentenweise Updates in ein entsprechendes Backend, das nicht transaktionell koordiniert werden kann. Konzepte wie «Zero-phase»- und «One-phase»-Commit, die durch gewisse Applikationsumgebungen vorausgesetzt werden, können damit gut unterstützt werden.

– EJBs und CORBA-Komponenten: Für den Component Broker ist die Unterscheidung der beiden Komponentenmodelle *CCM und EJB* irrelevant; die im CB realisierten Services stehen sowohl EJBs als auch CORBA-Objekten zur Verfügung (vgl. Architektur des CBs). Beide Modelle können interoperabel eingesetzt werden, das heisst, sie verwenden denselben Security-Mechanismus, sie können in einer gemeinsamen transaktionellen Umgebung mit eigenen Containers koexistieren, sie haben dieselbe Concurrency Policy usw. Beide Modelle ermöglichen containerunterstützte Persistence und die architektonische Trennung von Logik und Zuständen. Der Component Broker ist mit diesen Möglichkeiten anderen Produkten weit voraus.

– Business Rule Framework: Der Component Broker bietet ein flexibles *Business Rule Framework. Rule Objects* können durch *Business-Analytiker* leicht administriert werden. Auf Änderungen des Business-Umfeldes kann mit diesem Regelsystem mit beschränktem IT-Know-how reagiert werden. Das Regelsystem unterstützt *Preconditions, Postconditions* und weitere *Triggerpunkte, Constraints, Invarianten, Derivations, Classifications* und *Scripts*. So können auch komplexe, verschachtelte Regeln definiert werden.

– Skalierbarkeit: Wie die Advanced Edition skaliert auch der Component Broker horizontal. Zusätzlich skaliert der CB jedoch auch vertikal über verschiedene Plattformen (Windows NT, Sun Solaris, AIX, HP-UX) bis hinauf zum OS390 Mainframe. Auf diesen Plattformen werden *Workload Management, Resource Management* und die Security durch die Implementation des Component Broker unterstützt. Für Aufgaben wie beispielsweise komplexe, koordinierte Updates auf multiplen Datenbanken mit *Logging* und *Recovery* bietet der Component Broker höchste Performance. Es ist jedoch zu beachten, dass für einfachere Aufgaben die Advanced Edition eine bessere Performance als der Component Broker aufweisen kann.

– Architektur des Component Brokers: Die Architektur (Bild 10) zeigt die we-

Literatur

[1] WebSphere Application Server, Enterprise Edition; IBM Worldwide WebSphere Sales Team, A White Paper on IBMs Premier Application Server for Distributed Components, 12. Februar 2000.

[2] WebSphere Application Servers, A Quality of Service Continuum, IBM Software Sales Team, 6. März 2000.

[3] Clemens Szyperski: Component Software, Beyond Object-oriented Programming, Addison-Wesley, ACM Press, New York, 1997.

[4] Jeri Edwards: 3-Tiers Client/Server at Work, Revised Edition, John Wiley & Sons, 1999.

[5] Robert Orfali, Dan Harkey, Jeri Edwards: Client/Server Survival Guide, Third Edition, Revised Edition, John Wiley & Sons, 1999.

[6] Ed Roman: Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition, John Wiley & Sons, 1999.

sentlichen Building Blocks des *Component Brokers*. Backend-Systeme (Legacy) können mit Hilfe der Adaptertechnologie gut und tief integriert werden. Das Managed Object Framework (MOFw), die Containertechnologie, der ORB und die Services bilden zusammen die Server-Runtime-Umgebung der Applikation. Diese Umgebung unterstützt einerseits die Integration der Adapter in die Semantik der Server, bildet aber gleichzeitig das Rückgrat der CORBA-Services und der Komponenten, die über den Objectbus (IIOP) von anderen Komponenten genutzt werden können.

Develop – Run – Manage

Von Beginn weg wurden bei der Entwicklung des *Component Broker drei Eckpunkte* berücksichtigt:

- eine *Entwicklungsumgebung* (Develop) mit State-of-the-art-Werkzeugen zur Erzeugung von componentenbasierten Business-Lösungen
- eine robuste *Laufzeitumgebung* (Run) mit Enterprisequalität der Services
- eine *Configurations-, Controll- und Deployment-Umgebung* (Manage) für verteilte Objekte

Entwicklung

Die Anforderungsanalyse sowie das Design einer neuen Anwendung werden mit Rational Rose in UML geschrieben. Die resultierenden Definitionen und Schnittstellen können direkt in den Object Builder, das Komponenten-Repository und Managementtool des Component Broker, importiert werden. Dort können die Java-Objekte mit VisualAge for Java Enterprise Edition konstruiert, codiert, getestet, und als EJBs oder CORBA-Komponenten in Betrieb gesetzt werden. Die Entwicklungsumgebung umfasst folgende Funktionalität:

- Mittels Rose-Bridge können Resultate aus Analyse & Design in den Object Builder übernommen und weiterbearbeitet werden.
- Komposition neuer Objekte aus bestehenden Objekten durch Bildung neuer Beziehungen, Aggregation und Vererbung.
- Erzeugen von Komponenten, die über mehrere unterschiedliche Plattformen, von Windows NT über UNIX bis

Quellen/Formen	Lifecycle	Verfahren	Wert
individuelle Zeilen Code	Implementierung	Copy & Paste	niedrig
Procedure Libraries	Implementierung	Import, Call	niedrig bis mittel
Domain-Analyse	Analyse	Analyseergebnis (z.B. UML)	mittel
Klassen-Libraries	(Design), Implementierung	Import, Vererbung, Call (Instanzierung)	mittel
Pattern	Analyse, Design (Implementierung)	semi-formale Ideen-Doc.	mittel bis hoch
Frameworks	Design (Software architektur), Implementierung	Import, Vererbung, Polymorphismus, Erweiterung	hoch
Komponenten, Plug-ins (OLE, ORB, CORBA)	(Analyse, Design), Implementierung, Deployment, Run-time	Service Delegation, Shared Services, Usage, Komposition	sehr hoch

Tabelle 1. Wiederverwendbarkeit: Quellen, Lifecycle und Verfahren.

Advanced Edition		*Sicherheit* *hohe Verfügbarkeit* *skalierbare Laufzeitumgebung für EJBs (+ Tools)*
Enterprise Edition	wie WAE	*Sicherheit* *hohe Verfügbarkeit* *skalierbare Laufzeitumgebung für EJBs (+ Tools)*
	zusätzlich	gute CORBA-Unterstützung mit 10-Schlüssel-Objekt-Services (Query, Notification and Events) freie Auswahl der Implentierungssprache (Java, C++,...) robuste Interaktionen mit ActiveX und C++-Clients
	macht Enterprise QoS aus	transaktionelle Koordination mit CICS, IMS, MQSeries, DB2 und Oracle Connection Pooling für CICS, IMS und MQSeries Abbildung der Security Credentials von Legacy-Systemen hoch entwickelte Cache-Interstützung und Locking Policies Unterstützung für Session Services ausgezeichnete EJB und CORBA-Integration Einbindung eines Business Rules Framework hohes Niveau an Skalierbarkeit, Scope und Manageability

Tabelle 3. Unterschiede zwischen WebSphere Advanced und WebSphere Enterprise Edition.

	2-Tiers	3-Tiers
Systemadministration	Komplex (mehr Logik ist auf dem Client zu verwalten)	weniger komplex (die Applikation kann zentral auf dem Applikation Server verwaltet werden – Applikations programme können den Standard-System-Management-Werkzeugen bekannt gemacht werden)
Sicherheit	niedrig (Sicherheit auf Stufe Daten)	hoch (kann auf Services oder Methoden heruntergebrochen werden)
Datenkapselung	niedrig (Datentabellen werden preisgegeben)	hoch (der Client stösst Services oder Methoden an)
Leistung	schlecht (viele SQL-Anweisungen werden über das Netz verschickt, ausgewählte Daten müssen zur Analyse zum Client heruntergeladen werden)	gut (nur Service Requests und Responses werden zwischen Client und Server ausgetauscht)
Skalierbarkeit	schlecht (nur beschränkte Anzahl Links können verwaltet werden)	exzellent (konzentriert ankommende Sessions; kann Last auf mehrere Server verteilen)
Wiederverwendbarkeit der Applikationen	schlecht (monolithische Applikationen auf dem Client)	exzellent (kann Services und Objekte wieder verwenden)
Ease of development	hoch	wird besser (Standardwerkzeuge können zur Kreation von Clients verwendet werden und Werkzeuge entwickeln sich so weiter, dass sowohl Client als auch Server entwickelt werden können)
Server-to-Server-Infrastruktur	nein	ja (mit der serverseitigen Middleware)
Integration von	nein	ja (mit Gateways/Adaptoren)
Legacy-Applikationen		
Internetsupport	schlecht (Beschränkungen der Internetbandbreite erschweren das Herunterladen von fetten Clients)	ausgezeichnet (schlanke Clients sind einfacher herunterzuladen als Applets oder Beans; Remote Invocation verteilt die Applikationslast auf die Server)
heterogene Datenbankunterstützung	nein	ja (3-Tiers-Applikationen können mehrere Datenbanken in derselben Business-Transaktion verwenden)
breite Auswahl an Kommunikation	nein (nur synchrone, verbindungs-orientierte RPC-like-Aufrufe)	ja (kann RPC-like-Aufrufe, verbindungsloses Messaging, Queued Delivery, Publish-and-Subscribe und Broadcast unterstützen)
Flexibilität der Hardwarearchitektur	beschränkt (es gibt Client und einen Server)	ausgezeichnet (alle 3 Tiers können auf unterschiedlichen Rechnern, der zweite und dritte Tier können beide auf demselben Rechner sein; der zweite Tier kann auch auf mehrere Rechner verteilt werden)
Verfügbarkeit	schlecht (kann nicht durch einen Back-up-Server verbessert werden)	ausgezeichnet (die Middle-Tier-Komponenten können auf anderen Rechnern neu gestartet werden)

Tabelle 2. 2-Tiers- contra 3-Tiers-Client/Server.

OS390, portierbar sind. Portierbar bedeutet dabei bloss ein neues Deployment, ohne dass sich der Code selber ändert.
- Wiederverwendbarkeit exsitierender Datenbanken, Transaktionen, Legacy-applikationen und messagebasierter Middleware.
- Architektonische Trennung von Zuständen einer Komponente und ihrer Business Logik.
- Clients können in Java und C++ geschrieben werden; ActiveX Clients werden ebenfalls unterstützt.

Runtime-Umgebung

Die Serverumgebung für Komponenten in WebSphere Enterprise Edition hat sich im Laufe der Jahre zu einer Weltklasse-Ausführungsumgebung für verteilte Komponenten entwickelt. Sowohl Enterprise Java Beans als auch CORBA-Komponenten können mit derselben Qualität in WebSphere Container betrieben werden. Unabhängig von plattformspezifischen Stärken kann die WEE-Runtime-Umgebung als Komponenteninfrastruktur mit folgenden Eigenschaften zusammengefasst beschrieben werden:

- Containers, die sowohl EJBs als auch CORBA-Komponenten mit nahezu identischen Charakteristiken unterstützen.
- Tief integrierte Applikationsadapter zu Datenbanken, prozeduralen Applikationsumgebungen und messagebasierter Middleware.
- Breiteste, heute erhältliche CORBA-Unterstützung mit Naming, Transaktionen, Lifecycle, Notifikation, Concurrency.

System Management

Das System Management des Compo-

nent Broker ermöglicht die Definition, die Konfiguration, das Management, das Monitoring, die Installation und das Deployment der gesamten verteilten Topologie. Es verbindet den Output der Entwicklungsumgebung mit der Runtime-Umgebung und ermöglicht das Management der entsprechenden Komponenten und anderer Runtime-Elemente von einem zentralen Punkt aus. Dabei laufen intelligente Agenten (oder Node Manager) auf jeder Maschine des Netzes und führen Instruktionen des zentralen Koordinators aus und geben den Status an diesen zurück. Das System Management ermöglicht die Transformation des logischen Modells eines Systems in ein aktuelles Runtime-Image, das die entwickelten Komponenten enthält. Das System Management enthält unter anderem folgende Funktionalität:

- Identifizierung und Kategorisierung von Maschinen in einer verteilten Topologie zu Zellen oder Domains.
- Erzeugung von Verwaltungszonen mit Gruppen von Ressourcen.
- Definition von Servergruppen.
- Automatischer Neustart und Recovery ausgefallener Server.
- Möglichkeiten für manuelles Starten und Stoppen der Server und Überwachen deren Zustände.
- Konfiguration von Sicherheits-, Resourcesharing- und Memory Management Policies.
- Automatische Remote-Installation von Applikationen.
- Verschiedene Werkzeuge, unter anderem zur Untersuchung von Logs und zum Deployment von Komponenten.
- Zuteilung von Servern zu Firewall Ports.

11

Hans Peter Gisiger, *Dr. sc. techn., Dipl. El.-Ing. und Informatik-Ing. ETH, diplomierte 1982 zum Elektroingenieur und 1986 zum Informatikingenieur an der ETH Zürich. Anschliessend arbeitete er mehrere Jahre am Institut für Technische Informatik und Kommunikationsnetze (TIK) an der ETH, wo er 1992 auch zum Dr. sc. techn. an der Abteilung für Informatik promovierte. Seit 1995 arbeitet Hans Peter Gisiger bei Swisscom, zuerst drei Jahre bei CT und seit zwei Jahren im Technologiemanagement bei CIT-AE.*

Summary

Component technology with WebSphere

The WebSphere Enterprise Edition product has become a full-blown, end-to-end solution while initially being developed as a modern client/server application with enterprise-level quality. The wide range of tools it offers, its compatibility with different standards and the option of incorporating a variety of runtime environments makes the WEE product particularly suited to the requirements of large companies with activities both in new developments and in the area of integrating complex applications, systems and data. For small and medium-sized projects, companies can use the standard or advanced edition. These solutions can then be upgraded to the next-highest class affordably.

Lösen DSP mittelfristig die MCU ab?

Wenn es nach John Scarisbrick, Vizepräsident von Texas Instruments (TI), geht, dann werden digitale Signalprozessoren das Herz typischer Internetgeräte werden. Da für solche speziellen Anwendungen konventionelle Prozessoren zu teuer und auch zu langsam sind, beschreibt TI bereits ein Ablöseszenario. Das Unternehmen belegt seine These mit den Wachstumsraten für DSP, die zurzeit etwa 30% pro Jahr zulegen. Für das Jahr 2003 rechnet Scarisbrick für DSP bereits mit einem Markt von 13 Mia. US-$. Nachdem DSP programmierbar sind und mittlerweile auch «offene Systeme» zulassen, sei der Siegeszug nicht mehr zu stoppen.

Texas Instruments
13510 N. Central Expressway
Dallas TX 75243
USA.
Tel. +1-214-995 2011
Fax +1-214-997 3198

Preiswerte Kunststofffaser für Datenübertragung

Etwa 30-mal dicker als gewöhnliche Glasfasern für optische Übertragung ist eine neue Kunststofffaser, die von Asahi Glass und der Keio-Universität entwickelt wurde. Sie macht eine Übertragungsrate von 10 Gbit/s möglich und lässt sich aufgrund ihrer geringeren mechanischen Empfindlichkeit innerhalb einer Minute mit dem PC verbinden. Der Preis soll nur wenig über demjenigen konventioneller Glasfasern liegen, die pro Meter für etwa 1.50 bis 2 US-$ verkauft werden. Muster wurden bereits ausgeliefert. Ab Herbst wird NTT diese Faser in grösserem Umfang beziehen.

Asahi Glass Co. Ltd.
1-2 Marunouchi 2-chome
Chiyoda-ku
Tokyo 100
Japan
Tel. +81-3-3218 5555
Fax +81-3-3211 5071

System-on-Chip für 3D-Anwendungen

Ein SoC-Baustein mit erheblich verbesserter 3D-Bildverarbeitung wurde von Mitsubishi Electric entwickelt. Mit vier eingebauten 32-MB-DRAMs kann ein Datentransfer zwischen Chip und Umfeld von mehr als 50 GByte/s erreicht werden. Die Verarbeitungsgeschwindigkeit wird intern nochmals verdoppelt, indem man für «Schreiben» und «Lesen» des Speichers verschiedene Ports benutzt. Die Taktgeschwindigkeit des Chips liegt bei 222 MHz. Ab Dezember 2000 soll der Chip erhältlich sein. Er wurde vor allem (aber nicht nur) für das digitale Fernsehen entwickelt.

Mitsubishi Electric Corp.
2-3, Marunouchi
2-Chome
Chiyoda-ku
Tokyo 100
Japan
Tel. +81-3-3218 3499/2111