

Application Server- wohin geht die Reise?

Situationsbeschreibung

Der Druck, Anwendungen in immer kürzerer Zeit bereitzustellen, dabei jedoch gleichzeitig die rasch expandierenden technischen Möglichkeiten zu nutzen, nimmt stetig zu. Der einzige Weg zum Ziel führt über deutlich spürbare Produktivitätssteigerungen. Diese lassen sich nur bedingt durch Rationalisierungen im Software-Entwicklungsprozeß erzielen, auch wenn in diesem Bereich durchaus Fortschritte spürbar sind (z. B. Design Patterns). Größere Potentiale können hingegen erschlossen werden, indem konsequent auf eine komponentenbasierte Anwendungsentwicklung gesetzt wird und erweiterbare Frameworks die Basis bilden.

Vor etwa zwei Jahren hielt eine neue Produktkategorie Einzug: die Application Server. Microsoft Transaction Server (MTS) kann wohl als erstes Produkt dieser Kategorie gelten. In den vergangenen Monaten ergoß sich ein wahrer Strom von Produkten über den Software-Markt und ein Ende ist nicht abzusehen. Schon jetzt gibt es mehr als 30 Anbieter und die Großen der Software-Industrie sind durchwegs mit mindestens einem Produkt vertreten. Sämtliche Hersteller außer Microsoft scharten sich um Sun Microsystems und deren Enterprise JavaBeans-Komponentenmodell. Eine ganze Reihe von Produkten sind schon verfügbar, viele jedoch sind erst angekündigt bzw. befinden sich im Beta-Test.

Was sind Application Server?

Application Server stellen ein Ausführungsumfeld für Geschäftslogik (im Three-tier-Modell der middle tier) zur Verfügung und verbergen die Komplexität der verteilten Umgebung. Ein Application Server entspricht einem Framework, das auf einer oder mehreren definierten Plattformen (Plattformen für verteilte Objekte, Komponenten-Plattformen) basiert und sämtliche systemorientierten Dienste bereitstellt, die zur Ausführung von Geschäftslogik erforderlich sind. Die Geschäftslogik kann ganz oder teilweise durch Komponenten als "Fertig-Bausteine" repräsentiert werden.

Application Server Frameworks implementieren ein architektonisches Anwendungsmodell und stellen eine Programm-Infrastruktur zur Verfügung. Das dem Framework zugrundeliegende Design wird im Prinzip mit jeder entwickelten Anwendung wiederverwendet. In dieser Weise sind Application Server Frameworks ein sehr geeignetes Mittel zur Steigerung der Entwicklungsproduktivität.

Da Bediener-Oberfläche, Geschäftslogik und Datenhaltung auf unterschiedlichen Computersystemen implementiert sein können, ist eine gemeinsame Verständigungsplattform erforderlich, um die Ebenen miteinander zu verbinden. Diese Plattform kann durch eine Plattform für verteilte Objekte (insbesondere COM/DCOM-, CORBA- und Java-Plattform) zur Verfügung gestellt werden. Der Application Server implementiert Container als Ausführungsumfeld von Komponenten. Die zugrunde liegenden Komponentenmodelle sind COM/ActiveX (Microsoft) und Enterprise JavaBeans (Sun Microsystems).

Darüber hinaus stellen *Application Server-Frameworks* weitere systemorientierte Infrastruktur zur Verfügung. Neben den "Standard-Diensten" wie beispielsweise Verzeichnis-Dienst (Directory Service), Transaktions-Dienst (Transaction Service) und asynchrone Nachrichtenübermittlung über Warteschlangen (Message Queuing) zählen dazu auch Dienste, die die Skalierbarkeit und Verfügbarkeit in verteilten Umgebungen erhöhen. Zu nennen sind insbesondere dynamische Lastverteilung, Daten-Caching, Connection Pooling (Zuordnung von Datenbank-Connections aus einem Pool) und der automatische Wiederanlauf. Anwendungs-Design und -Entwicklung können auf der reichhaltigen Basis bereits vorhandener Dienste aufsetzen und sich auf die rein applikations-spezifischen Aufgaben konzentrieren.

Der Begriff "Application Server" kann durchaus auf eine ganze Reihe bereits lange vorhandener wie auch relativ neuer Software-Produkte angewandt werden. Die Spannweite reicht von den (objektorientierten) TP-Monitoren (z. B. BEA M3), die vor allem OLTP-Anwendungen gut unterstützen bis hin zu den Web Application Servers (z. B. IBM WebSphere Application Server), die ursprünglich zur Unterstützung der überwiegend dokumentenorientierten Web-Anwendungen entwickelt wurden.

Selbstverständlich kann jedes Anwenderunternehmen selbst ein Application Server Framework entwickeln. Aufgrund der hochgradig deckungsgleichen Anforderungen der Anwenderunternehmen und des hohen Investitionsaufwands (mehrere Mio. DM) ist dies jedoch wenig sinnvoll. Es bietet sich an, bereits vorhandene Frameworks der Software-Hersteller in Erwägung zu ziehen. Diese bieten die folgenden Vorteile:

- Wiederverwendbarkeit: unterschiedliche Frontends (z. B. statische und dynamische HTML-Frontends, Java-Frontends (Applets, Java-Anwendungen), C++-Anwendungen,) können gemeinsam dieselbe Geschäftslogik nutzen.
- Infrastruktur: allgemeine, häufig benötigte Funktionen, werden bereits vom Framework zur Verfügung gestellt und müssen nicht selbst entwickelt werden.
- bessere Administrierbarkeit: die Zentralisierung der Geschäftslogik erleichtert die Administration im Vergleich zu *fat clients* (ein Großteil der Geschäftslogik wird auf der Anwender-Arbeitsstation ausgeführt).
- bessere Fehlertoleranz: die Verfügbarkeit des bzw. der Application Servers wird mit geeigneten Maßnahmen deutlich gesteigert,
- bessere Skalierbarkeit: Frameworks enthalten Funktionalität zur Leistungssteigerung. Ein Wechsel der System-Plattform kann hinausgeschoben werden.

Wie entwickelt sich der Markt?

Das prognostizierte Marktpotential für Application Server ist beträchtlich. Dadurch entsteht für Software-Hersteller ein enormer Anreiz, an dem zu verteilenden "Kuchen" zu partizipieren. Das Produktangebot ist gegenwärtig noch sehr unübersichtlich. Die Versuchung ist groß, das modische Etikett "Application Server" zu nutzen, auch wenn das Produkt den Mindestanforderungen nicht entspricht. Die meisten Produkte stehen noch ganz am Anfang ihres Lebenszyklus und sind deshalb noch nicht ausgereift.

Die überwiegende Zahl der Produkte nutzt einen CORBA-ORB als Object Bus, wobei Iona Orbix und Inprise VisiBroker die am häufigsten verwendeten Implementierungen sind. RMI ist als Object Bus wenig interessant, da nur Objekte miteinander kommunizieren können, deren Objektklassen mit Java als Programmiersprache entwickelt wurden. COM/DCOM ist für Software-Hersteller als Plattform völlig uninteressant, da Microsoft mit Microsoft Transaction Server (MTS) das Feld bereits besetzt hält. Für Software-Hersteller wäre es wirtschaftlich völlig unattraktiv, ein Konkurrenzprodukt zu entwickeln, zumal MTS zu einem Bestandteil von Windows NT evolviert wird. Es ist lediglich denkbar, daß Software-Hersteller add-ons zu MTS entwickeln (z. B. Generator für Geschäftsregeln).

Die Komponententechnologien sind der Schlüssel, um Application Servers zum Durchbruch zu verhelfen (die weitere Voraussetzung der Skalierbarkeit der Application Servers wird als erfüllbar vorausgesetzt). Abgesehen von den bereits etablierten ActiveX-Komponenten spielen Enterprise JavaBeans eine entscheidende Rolle, nicht zuletzt auch vor dem Hintergrund der indirekten Unterstützung durch die OMG. Gleichzeitig sind Komponententechnologien der Katalysator, um die Verschmelzung der bisher noch getrennten Linien der Web Application Server und der konventionellen Application Server zu forcieren. Als gemeinsamer Nenner würde sich dann die Bezeichnung „Enterprise Application Server“ anbieten.

Die Selektionskräfte des Marktes werden langfristig nur vier oder fünf Produkte übrig lassen (einschl. der Microsoft-Produkte), die zusammen mehr als 80% des Marktvolumens auf sich vereinigen werden. Das einzige Argument der Microsoft-Wettbewerber ist die Plattformunabhängigkeit ihrer Produkte. Microsoft wird durch äußerst niedrige Lizenzpreise (MTS ist derzeit Teil der Microsoft Backoffice Suite) dazu beitragen, daß sich das Lizenzpreisniveau auf einem insgesamt niedrigen Niveau halten wird.

Ungeklärte Fragen

Komponentenmodelle spielen eine zentrale Rolle. Von den derzeit drei wichtigsten Komponentenmodellen ist das CORBA-Komponentenmodell noch nicht abschließend spezifiziert. Microsoft hat mit dem COM/ActiveX-Komponentenmodell die Pflöcke bereits eingeschlagen, ebenso auch Sun Microsystems mit dem Enterprise JavaBeans-Komponentenmodell. Alle drei Komponentenmodelle weisen auf der technisch-konzeptionellen Ebene viele Gemeinsamkeiten auf und streckenweise hat es den Anschein als würde lediglich unterschiedliche Terminologie verwendet. Die Ernüchterung folgt jedoch in Gestalt unterschiedlicher Anwendungsprogramm-Schnittstellen (APIs).

Anwenderunternehmen erwarten, daß selbst entwickelte oder von Software-Herstellern lizenzierte Komponenten portabel sind, sofern sie auf demselben Komponentenmodell basieren. In der Microsoft-Welt werden die Erwartungen vollständig erfüllt, da Microsoft die Spezifikationen vorgibt und außerdem die Container selbst implementiert (z.B. MTS). Dem Entwickler bleibt kein Freiraum zur Einführung von Inkompatibilitäten.

Entwickler von EJB-Komponenten sehen sich mit einer völlig anderen Situation konfrontiert. Sun Microsystems entwickelt und pflegt die Spezifikation, implementiert jedoch selbst keine Container. Den Software-Herstellern wird damit Tür und Tor geöffnet, APIs um proprietäre Elemente zu erweitern und auf diese Weise Kundenbindung zu realisieren. In der Konsequenz kann dies dazu führen, daß Komponenten nicht mehr portabel sind. Sun Microsystems kann dieser für Entwickler wenig attraktiven Perspektive nur mit einer eigenen Referenzimplementierung entgegensteuern. In der Tat wird bereits an einer solchen gearbeitet. Mit einer Referenzimplementierung wird gleichzeitig noch ein weiteres Problem gelöst: die unterschiedliche Interpretation der EJB-Spezifikation durch die verschiedenen Hersteller. Container-Implementierungen, die sämtliche Konformitätstests durchlaufen haben, entsprechen damit der EJB-Spezifikation und können zertifiziert werden. Wie die Software-Hersteller reagieren werden, die an uneingeschränkter Portabilität kein Interesse haben können, läßt sich schon erahnen. Sehr wahrscheinlich werden sie die EJB-Spezifikation vollständig umsetzen, jedoch in gewohnter Manier zusätzliche proprietäre API-Erweiterungen einführen.

Derzeit sind die CORBA- und EJB-Komponentenmodelle nicht deckungsgleich. An einer Harmonisierung wird gegenwärtig gearbeitet. Welches Ergebnis am Ende stehen wird, ist heute noch nicht genau abzusehen. Interoperabilität ist jedenfalls angestrebt. Somit soll eine EJB-Komponente in einem CORBA-Container ausführbar sein und umgekehrt eine CORBA-Komponente in einem EJB-Container.

Im Unterschied zur Vergangenheit haben Hersteller mit proprietären Erweiterungen keinen unbegrenzten Spielraum mehr. Treiben sie das Spiel zu weit, verhindern sie das Aufblühen eines Komponentenmarkts durch Partikularisierung (insbesondere auch für Server-Komponenten) und spielen damit Microsoft in die Hände. Ein wichtiges Entscheidungskriterium für Anwender wird schließlich auch die Angebotsbreite und -tiefe im Komponentenmarkt sein. Daß die Hersteller so vernünftig sein werden, sich nicht kollektiv selbst ein Bein zu stellen, darf gehofft werden. Zum gegenwärtigen Zeitpunkt ist die Hoffnung jedoch noch nicht durch Fakten unterlegt.

Die absehbare Zukunft

Es ist absehbar, daß Application Server-Frameworks in der nahen Zukunft eine sehr wichtige Rolle spielen werden. Gleichzeitig wird eine gewisse Sogwirkung entfaltet, die zur Vereinigung bisher separater Produkte führen wird. Die "Alles-aus-einer-Hand"-Wünsche der Anwenderunternehmen werden dazu führen, daß z. B. Anbieter von Object Request Broker-Produkten ihre Produkte zu vollständigen Application Server-Frameworks ausbauen werden. Abgesehen von nach wie vor spezialisierten Frameworks (z. B. Datenbanksystem-Framework), die nach wie vor ihre Berechtigung haben werden, werden sich als größter gemeinsamer Nenner TP-Monitor-Frameworks als generische Application Server-Frameworks herausbilden. Dies schließt jedoch nicht aus, daß auch spezialisierte Frameworks ein oder mehrere Merkmale der Application Server aufweisen und beispielsweise EJB-Container implementieren können.

Application Server-Frameworks werden als Produktivitäts-Hilfsmittel unverzichtbar sein. Da die Unternehmensgrenzen in der Zukunft noch weniger als heute die Grenzen der betrieblichen Informationsverarbeitung definieren und die Unternehmen noch wesentlich intensiver zusammenwirken werden, ist vor allem die Interoperabilität eine wichtige Anforderung. Der von seiten der Anwenderunternehmen ausgeübte Druck wird die Microsoft-Wettbewerber daran hindern, proprietäre Konzepte ungehemmt durchzusetzen.