

Componentware: Objekt-Technologie in neuem Gewand

Die Diskussion über Componentware und Component Based Development erfährt durch Internet und Java eine neuen Aufschwung. Objekt-Technologie bildet dabei in unterschiedlichen Formen die Basis für die zukünftige Art der Anwendungsentwicklung.

Software so zu entwickeln, wie es die Industrie seit Jahrzehnten vormacht - das Zusammenfügen getrennt erstellter Bauteile zu einem neuem Ganzen - bewegt die Softwareingenieure schon einige Zeit. Auf die Softwareindustrie übertragen heißt dies: Anwendungserstellung durch die Montage unabhängig voneinander erstellter Softwarebausteine. Durch CASE glaubte man sich diesem Ziel schon einmal nahe gekommen zu sein. Die Anlehnung des Begriffs CASE an das Computer Aided Design (CAD) der Industrie löste allein die Problematik nicht. Aber Techniken wie CASE und der folgende Trend zur Objektorientierung mit dem Anspruch der Erstellung wiederverwendbarer Objekte waren bereits die ersten Anläufe einer 'industriellen Software Produktion'. Beide Innovationen scheiterten aber an ihren eigenen, zu hoch gesteckten Erwartungen, wodurch letztendlich eine flächendeckende Einführung verhindert wurde.

Die furiose Entwicklung im Umfeld des Internet und eine durch Java neu entfachte Diskussion über Methoden, Objektorientierung und verteilte Objekte, haben der ursprünglichen Idee der industriellen Produktion von Softwarebausteinen wieder zu neuem Aufschwung verholfen. Java und Internet stellen sich somit als Hoffnungsträger einer komponentenbasierten Anwendungsentwicklung mit dem Ziel der Schaffung eines Komponentenmarktes dar.

Insgesamt gesehen steht wohl außer Frage, daß eine Lösung benötigt wird. Der enorme Druck, der durch die Globalisierung und Deregulierung von Handel, Industrie und Dienstleistung auf den Unternehmen lastet, läßt auch dem IT Bereich nicht mehr viel Spielraum zur Selbstverwirklichung. Die neue, kundenorientierte Sichtweise der Unternehmen zwingt die IT, liebgewonnene Vorgehensweisen zu überdenken und neue Wege zu beschreiten. Diese neue Sichtweise erfordert aber auch eine neue Sicht auf die Informationprozesse und deren Implementierung. Erfolg hat nur der, der es schafft diese neuen Prozesse in enger Zusammenarbeit mit den Fachabteilungen zu realisieren, d.h. die Geschäftsanforderungen - unter Einbeziehung neuester Technologie - werden zur treibenden Kraft bei der Entwicklung neuer Applikationen.

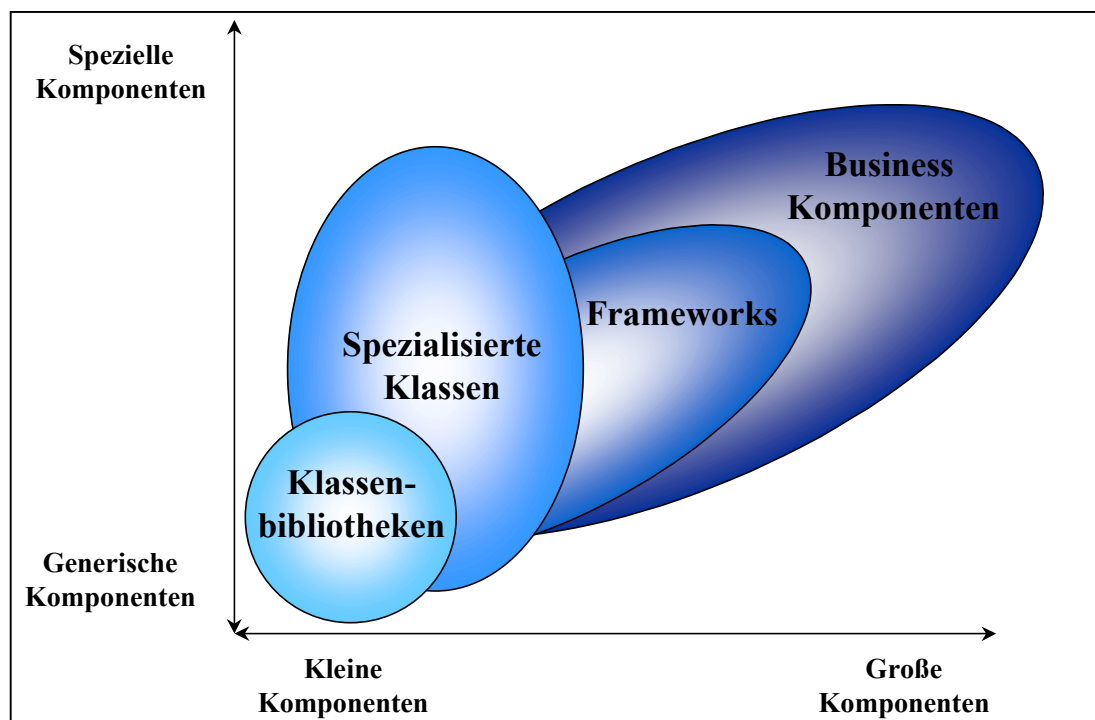
Bei der Lösung der anstehenden Probleme stehen die Verantwortlichen mehr denn je vor der Frage: Make or Buy ? Das heißt, individuelle Entwicklung von Lösungen versus der Standardsoftware von der Stange. Beide Richtungen sind mit entsprechenden Vor- und Nachteilen behaftet. Eine Verschmelzung beider Vorgehensweisen unter Ausnutzung der jeweiligen Vorteile und Eliminierung der Nachteile verspricht die Lösung zu sein. Und genau hier setzen Componentware bzw. Component Based Development an. Aus Make *or* Buy wird Make *and* Buy. Standardbausteine, z.B. extern eingekaufte Komponenten werden mit selbst entwickelten Modulen zu einer neuen Lösung kombiniert. Wobei der jeweilige prozentuale

Anteil von Make and Buy an der Gesamtlösung stark schwanken kann. Der Vorteil liegt auf der Hand: Überall dort, wo Standardsoftware die Geschäftsprozesse unterstützen kann, werden fertige Komponenten eingesetzt und nur dort wo Individualsoftware den entscheidenden Wettbewerbsvorteil garantiert, wird in eigene Entwicklung investiert. Einige wesentliche Faktoren bei diesem Szenario sind:

- Die Verfügbarkeit fertiger Komponenten, d.h. ein gut ausgestatteter Komponentenmarkt
- Einfache Integration gekaufter Komponenten mit Individualsoftware
- Bereitstellung entsprechender Werkzeuge zur Komponentenerstellung und -integration
- Ausgereifte und verfügbare Standards

Der Weg zum Komponentenmarkt

Wie viele andere Ausdrücke in der Softwaretechnologie hat der Begriff Komponente unterschiedliche Bedeutungen für unterschiedliche Personen. Die einen verstehen darunter eine Klassenbibliothek, andere ein OCX-Control - und beide haben Recht. Komponenten gibt es in verschiedenen Ausprägungen, unterschieden durch den Grad der Spezialisierung. So ist eine Klassenbibliothek ebenso eine Komponente wie ein Business Object, das sich aus mehreren Komponenten zusammensetzen kann. Eine Komponente kann, muß aber nicht, mit Objekt- Technologie erstellt werden. Letzteres erleichtert aber sicher die Integration in neue Welten und stellt somit auch einen Investitionsschutz für die Zukunft dar.



Komponententypen

Was ist eine Komponente

Eine Komponente ist ein Stück Software mit einem klar abgegrenztem Aufgabenbereich und dem Ziel, dem Entwickler einen wesentlichen Teil seines Programmieraufwands abzunehm-

men. Komponenten kapseln ihre Funktionalität und stellen sie über definierte Schnittstellen der Außenwelt zur Verfügung. Eine Komponente kann sowohl andere Komponenten beinhalten, wie auch Teil eines Ganzen sein. Komponenten können von unterschiedlichen Herstellern gekauft und mit selbst entwickelten Komponenten vermischt werden um daraus eine Anwendung für den Endanwender in der Fachabteilung zu erstellen.

Komponentenverteilung

Komponenten sind traditionell eng verbunden mit der Entwicklung der visuellen Elemente einer Anwendung. Hier tragen sie erheblich zur Verkürzung der Implementierungszeiten bei (z.B. durch Kauf einer GUI Klassenbibliothek). Dieser Aspekt wird auch weiterhin Gültigkeit behalten, jedoch werden Komponenten in erhöhtem Maße im gesamten Lebenszyklus einer Anwendung zum Einsatz kommen. Besonders das einfache Austauschen von (z.B. fehlerhaften Komponenten wird dabei von Bedeutung sein.

Bei der Verwirklichung dieses Ziels spielt die zunehmende Intensität mit der Unternehmen ihre weitere Anwendungsentwicklung mit dem Internet bzw. Intranet verknüpfen, eine wesentliche Rolle. Objekt-Technologie wird in neuer, anderer Form allgegenwärtig sein. Sie wird immer mehr eingesetzt als begleitende, unterstützende Technologie, als um ihrer selbst Willen.

Bei der Realisierung einer auf Objekt-Technologie basierten Komponentenentwicklung nimmt die notwendige Verteilung eine besondere Position ein. In diesem Umfeld stehen sich CORBA von der OMG und Active Server/COM von der Open Group, in der Microsoft vertreten ist, gegenüber. Die Frage welche Technologie zum Einsatz kommt ist abhängig davon, welches Ziel man verfolgt. Für Active Server/COM von Microsoft spricht die Integration in Microsoft zentrierten Client/Server Umgebungen in der Skalierbarkeit nur eine untergeordnete Rolle spielt und man auf Plattformunabhängigkeit verzichten kann. Zudem können DCOM Anwendungen selbst mit Tools wie Visual Basic erstellt werden, was nicht unerheblich zu einer größeren Akzeptanz führen kann. CORBA's Vorteil hingegen liegt in der Verarbeitung größerer Transaktionsraten wie sie z.B. für Internet Anwendungen nötig sind. CORBA bietet eine höhere Flexibilität im Sinne der unterstützten Plattformen. Diese wird jedoch mit einem höherem Programmieraufwand erkaufte.

Beide Technologien werden noch einige Zeit für Probleme bei der Interoperabilität und Skalierbarkeit aufwerfen. Es kann jedoch davon ausgegangen werden, daß diese Probleme bis zur Jahrtausendwende weitestgehend gelöst sind und eine DCOM/CORBA Bridge zur Verfügung stehen wird.

Komponentenarchitekturen

Der Begriff Komponente wurde den meisten Softwareentwicklern wohl das erste Mal durch die Visual Basic Controls (VBX) von Microsoft näher gebracht. Geradezu explosionsartig entstand ein Komponentenmarkt für VBX Controls. Der Nachfolger, die auf OLE/COM basierenden OCX Controls, stellen heute im wesentlichen den Standard für die Windows Plattformen dar.

Durch das Internet und die Browseroberflächen entwickelte sich eine weitere Art von Komponenten - nämlich die, die den Browser als Container benutzen. Dies sind zum einen Java Applets und ActiveX Komponenten, als auch Netscape Plug-ins.

Für die weitere Entwicklung der Komponenten Technologie sind im wesentlichen zwei Architekturen von Bedeutung: JavaBeans von JavaSoft und ActiveX von Microsoft. Für beide Architekturen stehen auch entsprechende serverbasierte Technologien zur Verfügung: Enterprise JavaBeans und Active Server.

Microsoft's ActiveX verfügt z. Zt. nur über eine eingeschränkte Unterstützung unterschiedlicher Plattformen. Demgegenüber steht allerdings, daß Active X die ausgereifere Technologie darstellt, da sie auf der schon länger etablierten OLE Technik basiert. Tausende kommerziell verfügbarer Controls (oder Komponenten) bilden eine umfassende Bibliothek aus der Kunden wählen können. Die Granularität der Komponenten ist, ähnlich wie bei Java Applets, relativ hoch. Die einzelnen Komponenten sind typischerweise für kleinere Aufgaben mit klar definierten Ein- und Ausgaben bzw. Ereignissen konzipiert.

Ein Vorteil der JavaBeans Architektur gegenüber ActiveX ist die Flexibilität und Interoperabilität. Dies ist um so wichtiger auf den Server-Plattformen. Für viele Unternehmen in denen UNIX eine strategische Server-Plattform ist, sind die Multi-Plattform Fähigkeiten von Java und JavaBeans attraktiver.

Komponentenentwicklung

Die eingangs beschriebene Konzentration auf die Lösung der Geschäftsprozesse führt zu der Forderung, daß die eigentliche Programmierung (im Sinne von Codierung) einen immer geringeren Anteil am Gesamtentwicklungsaufwand haben sollte. Das heißt, Software soll zur Lösung eines Problems beitragen und nicht selbst ein Problem sein. Die Entwicklung der letzten Jahre bei den Werkzeugen folgt dieser Forderung, ohne ihr jedoch in bisher ausreichendem Maße zu genügen. Durch Component Based Development ergibt sich nun aber die Möglichkeit den Entwicklungsprozess neu zu strukturieren und in zwei unterschiedlichen aber miteinander verknüpften Vorgehensweisen zu realisieren:

1. Komponentenentwickler

Dieser Personenkreis ist verantwortlich für die Erstellung von Spezial- oder Basiskomponenten und arbeitet aller Voraussicht nach mit objektorientierter Technologie, ausgehend von einer OO Analyse und Design bis zur Fertigstellung der Komponente. Hier ist Spezialwissen und eine genaue Kenntnis der jeweiligen darunterliegenden Technologie erforderlich. An dieser Stelle kann auch die Integration extern eingekaufter Komponenten erfolgen. Diese Art Entwickler ist derjenige, den wir heute üblicherweise in den Unternehmen antreffen.

2. Komponentenmontierer

Dieser neue Typ eines Entwicklers ist derjenige, der über das Wissen der Geschäftsprozesse verfügt. Ausgestattet mit einer Komponenten-Toolbox und werden neue Anwendungen montiert. Der Komponentenmontierer kümmert sich nicht mehr um Technologie, sondern um Geschäftslogik und deren optimale Abläufe. Ein solcher Entwickler kommt in der Regel aus dem Fachbereich. Komponentenmontierer findet man heute nur recht selten. Dies liegt u.a. daran, daß die zur Verfügung stehenden visuellen Entwicklungswerkzeuge mehr oder weniger auf den Komponentenentwickler ausgelegt sind. Der Prozeß des Zusammenfügens und Montierens ist in der kommerziellen Anwendungsentwicklung bisher noch nicht ausreichend implementiert.

Unsere Untersuchungen zeigen, daß Unternehmen verstärkt nach Paradigmas suchen, die es ihnen erlauben, die Erfahrungen der Fachabteilungen mit der zur Verfügung stehenden Komponententechnologie zu verknüpfen. Die Softwareindustrie wird sich dementsprechend bis zum Jahr 2002 in erhöhtem Maße in Richtung komponentenbasierter Architekturen orientieren. Dies wird zu einem Wachstum in allen Bereichen der Objektorientierten Technologie führen, einschließlich der objektorientierten Analyse und Design (OOAD). Der

Markt für OOAD, ca. US\$ 100 Millionen in 1996, wird sich um ca. 30% per Annum bis zum Jahr 2000 erweitern.

OOAD Werkzeuge erlauben den Entwicklungsgruppen die Ausnutzung von Objekt-orientiertem Design ohne notwendigerweise in einer 3GL wie C++ programmieren zu müssen. Der Pluspunkt der OOAD Werkzeuge ist, daß immer mehr Funktionalität auf den Aspekt der Generierung gelegt wird, was wiederum in den meisten Fällen zu einer Reduzierung des Kodierungsaufwands führt.

Bei den Entwicklungsumgebungen gab es bisher eine relativ eindeutige Trennung zwischen mehr grafisch orientierten Umgebungen wie IBM VisualAge, Microsoft Visual Basic, Borland Delphi oder Sybase PowerBuilder und den mehr auf die Sprache fokussierten Umgebungen wie Borland C++, Sybase Optima++ und Microsoft Visual C++. Hier zeichnet sich nun eine Verschmelzung der höheren Produktivität des visuellen Ansatzes mit der besseren Skalierbarkeit der 3GL Sprachen ab. Diese Entwicklung wird sich auch auf die Tools zur Java Entwicklung ausbreiten, was dazu führen wird, daß innerhalb der nächsten zwei Jahre Entwicklungs-Suites entstehen, die einen Code Austausch zwischen Java und C++ erlauben. Am weitesten fortgeschritten ist hier VisualAge.

So begrüßenswert all diese Entwicklungen auch sind. Letztendlich adressieren sie nach wie vor nur den Bereich der Komponentenerstellung, d.h. sie bringen nur dem Komponentenentwickler Vorteile. Für den Komponentenmontierer sind sie immer noch zu komplex und zu technisch orientiert. Für ihn wäre eine rein grafisch, d.h. visuell orientierte Art des Zusammenstellen von Anwendungen die einfachste Vorgehensweise.

In anderen, zumeist von der industriellen Produktion geprägten Bereichen, finden sich hierfür bereits Beispiele. So z.B. in der Meßtechnik oder Prozeßsimulation. Die hier zum Einsatz kommende Form des visuellen Programmierens zeichnet sich dadurch aus, daß keinerlei Art von textueller Kodierung zur Anwendungserstellung nötig ist. Allerdings sind diese Anwendungen zumeist extrem spezialisiert da sie in der Regel auf proprietären Architekturen basieren und dementsprechend nicht offen sind. Die Erstellung traditioneller Applikationen mit den Anforderungen der heute anzutreffenden Infrastruktur ist hiermit so gut wie nicht möglich.

Es gibt Ansätze, die visuelle Programmierung aus der technisch wissenschaftlichen Nische in den kommerziellen Bereich zu übertragen. Ein Tool, das diese Brücke schlagen will, ist z.B. VISUA-Softwarestudio der Triple-S GmbH, Deutschland. VISUA-Softwarestudio basiert auf dem Konzept, Bausteine mittels eines Diagrammeditors grafisch zu einer Anwendung zusammenzufügen. Der Komponentenentwickler erstellt ein Modul mittels Standardtools wie z.B. Delphi oder Microsoft C++ und generiert danach eine VISUA 'Hülle', woraus aus diesem Modul ein VISUA-Baustein wird. Dieser Baustein steht dann in einem Diagrammeditor für den Komponentenmontierer zur Verfügung. Die fertig montierten Anwendungen lassen sich noch während des Editierens ausführen, da sie interpretativ sind. VISUA-Softwarestudio wird mit einer Reihe fertiger Bausteine ausgeliefert, so z.B. für sämtliche visuellen Elemente, wie man sie auch von Visual Basic oder Delphi kennt. OCX Controls, JavaBeans oder andere Komponenten lassen sich ebenfalls integrieren. Allerdings erst, nachdem auch hierfür die entsprechende 'Hülle' generiert wurde, d.h. aus einem OCX ein VISUA-Baustein geworden ist.

Der Vorteil des grafischen, visuellen Programmierens dieses Tools wird allerdings mit der Abhängigkeit vom Hersteller erkauft, da die mit dem Diagrammeditor montierten Anwendungen auf ein Runtime-System angewiesen sind.

Dieses Beispiel zeigt, daß Ansätze aus Randbereichen der kommerziellen EDV hilfreich bei der Suche nach Lösungen zur Vereinfachung von Component Based Development sein können. Ob diese Lösungen allerdings für die stetig wachsenden Anforderungen in den Unternehmen tragfähig sind, muß sich in der Praxis beweisen.

Bottom Line: Objekt-Technologie wird bis zum Jahr 2000 alle Bereiche der Softwareentwicklung durchdringen. ActiveX wird weiterhin eine führende Rolle im Markt für visuelle Komponenten spielen, wobei Java basierte Applets an Sichtbarkeit und Popularität gewinnen werden. Visuelle Tools, wie z.B. VISUA, bieten für die Zukunft von Componentware eine interessante Ergänzung, die allerdings noch in den Anfängen steckt, aber ein gewisses Potential verspricht.

META Group Research

Copyright © 1997 META Group Germany Research & Consulting

85737 Ismaning, Oskar Meßter Straße 24 RGB. <http://www.metagroup.de>

Telephone xx49 (89) 99696-05. Facsimile xx49 (89) 99696-169. All rights reserved.