von Ulla Schönhense

Über den Autor
Ulla Schönhense, Manager Technical Services, E-Business Solutions bei Micro Focus

Komponentenmodelle spielen in der Software-Entwicklung eine immer größere Rolle. Von den Möglichkeiten dieser Technologie will auch COBOL profitieren. Mit modernen Werkzeugen lässt sich COBOL heute ohne weiteres mit COM+, Corba oder EJB verwenden.

Ob Lochkarten oder das Internet eingesetzt werden, die zentrale Aufgabe der IT ist es immer, Prozesse zu automatisieren – je nach Stand der technischen Entwicklung entweder ganz einfache Abläufe oder komplexe Geschäftsmodelle. In jedem Fall müssen zunächst die Prozesse in einem Format abgebildet werden, das den IT-Systemen verständlich ist. Mit den Anforderungen, aber auch mit den Möglichkeiten einer fortschreitenden Technologie, hat sich das Vorgehen, wie man die Realität für die Zwecke der Softwareentwicklung zu abstrahieren versucht, im Laufe der Jahre stark gewandelt.

Künstler Burgy Zapp

In den Anfängen war es üblich, Prozesse in sequentielle Abläufe zu zerlegen. Dies wurde auch durch die sogenannten strukturierten Programmiersprachen unterstützt. Anfang der 90er Jahre wurde die Objektorientierung eingeführt und mit ihr ein neues Abbildungsverfahren für die Realität. Inzwischen verbreitet sich immer mehr eine Kombination von beiden Methoden.

Programmierung: prozedural und objektorientiert

In der prozeduralen Programmierung, welche die IT lange Jahre dominiert hat und in weiten Bereichen bis heute das wichtigste Modell darstellt, stehen die Abläufe im Mittelpunkt. Diese müssen beschrieben und dabei „sequentialisiert“ werden. Dabei entstehen logische Einheiten, sogenannte Module, die manchmal auch in mehreren Programmen genutzt werden können. Letztlich gilt es, diese Module zu einer sequentiellen Abfolge zu verbinden, dafür dienen „Prozeduren“ und/oder Funktionen bei deren Aufruf Daten unter den Modulen ausgetauscht werden. Dabei entstehen sogenannte Aufruf-Hierarchien. Prozedurale Programme beschreiben also Abläufe, die Komplexität der Programme liegt damit in den Algorithmen. Die Anwendungen sind daher

empfindlich, wenn der Ablauf selbst sich ändern sollte.

War das Abstraktionsverfahren der prozeduralen Programmierung die Zerlegung der Abläufe, so versuchte die objektorientierte Programmierung das Abstraktionsverhalten des Menschen auf die IT zu übertragen. Dabei wird die Realität als eine Ansammlung von Objekten aufgefasst, die klassifiziert werden können. Eine Klasse beschreibt also den Bauplan eines Objekttyps mit seinen Eigenschaften, den Attributen, und seinen Fähigkeiten, den Methoden. Darauf basierend entsteht also eine Anwendung, indem Objekte dieser Klassen Aktionen ausführen, so können sich Objekte zum Beispiel untereinander manipulieren.

Der objektorientierte Ansatz hat zum Ziel, die Programmierung zu vereinfachen, indem ein „natürliches" Verhalten des Menschen – Klassifizierung, Abstraktion und Bilden von Hierarchien – angewendet wird. Allerdings entsteht dabei eine neue Komplexität: die in der Beziehung von Objekten untereinander. Eine Veränderung der Beziehungen eines einzelnen Objektes kann weitreichende Folgen auf andere Objekte haben.

Software-Entwicklung mit Bausteinen

Das größte Problem der Softwareentwicklung ist immer die Zeit, die benötigt wird, vorhandene Systeme zu ändern, anzupassen und zu erweitern. Die objektorientierte Programmierung hat zwar das Programmieren entscheidend verändert, ihre ursprüngliche Zielsetzung – die Vereinfachung der Softwareentwicklung – jedoch nur zum Teil erreicht. Im Gegensatz zur Software ist die Entwicklung von Hardware heute viel schneller. Hier kommunizieren einzelne Bausteine über standardisierte Verbindungen und können via Plug & Play einzeln ausgetauscht werden. Bei der Komponenten-Architektur geht es im Grunde um nichts anderes. Plug & Play soll auch für Softwarebausteine funktionieren und so immer kürzere Entwicklungszyklen ermöglichen.

Mit Einführung der Komponenten-Architektur versucht man, die Komplexität der Prozesse in überschaubare Bausteine zu unterteilen. Dabei übernimmt jeder Baustein eine Teilaufgabe des gesamten Prozesses und realisiert diesen Teil völlig eigenständig. Die Implementierung der Bausteine kann an kleine Projektguppen weitergegeben werden, die wiederum beliebig objektorientiert oder strukturiert vorgehen können. Um aus den so entstandenen Bausteinen eine vollständige Anwendung zusammenzubauen, müssen die einzelnen Teile in Beziehung zueinander gebracht und eventuell mit Kommunikationsschnittstellen versehen werden.

Eine Komponente ist ein Softwarebaustein, der für sich eine vollständige Anwendung darstellt, die jedoch nur einen kleinen Teil des gesamten Systems realisiert. Softwarebausteine kann man selber entwickeln oder auch als fertige Komponenten einkaufen, beispielsweise Microsoft Word als Baustein, der die Funktion einer Textverarbeitung übernimmt.

Um eine gute Übersicht über alle vorhandenen Bausteine zu haben, ist es wichtig, ein Repository anzulegen. Dabei müssen die Bausteine mit ihrer Schnittstelle, welche die Kommunikation mit anderen Bausteinen regelt, und den realisierten Aufgabenstellungen beschrieben werden. Will man nun eine Applikation realisieren, analysiert man zunächst, welche Aufgaben diese Applikation realisieren soll. Dann lassen sich im Repository die entsprechenden Komponenten ermitteln, die verschiedene Teilaufgaben übernehmen können. Sind alle notwendigen Bausteine beisammen, kann man die eigentliche Applikation realisieren, indem man die verschiedenen Bausteine aufruft.

Komponenten – Typen und Technologien

Zwei Typen von Komponenten lassen sich unterscheiden. Die In-Process-Komponenten sind lokale Komponenten, die als ein Bestandteil des Anwendungsprozesses gelten. Diese Komponenten sind im

Prinzip gebundene Unterprogramme, die jedoch über die standardisierte Schnittstelle angesprochen werden. Typische Vertreter sind COM- und JavaBean-Komponenten.

Demgegenüber gibt es die verteilten Komponenten. Für die Anwendung sind es entfernte Komponenten, welche verteilt im Netzwerk liegen können und die auch extern zur Anwendung verwaltet werden. Diese Modelle beinhalten also eine Netzwerk- Kommunikation, auch wenn die Komponente selbst eventuell sogar auf dem gleichen Rechner liegt. Bei den verteilten Architekturen haben sich in den letzten Jahren drei unterschiedliche Technologie etablieren können:

> COM+ / DCOM — Distributed Component Object Model Corba / CCM — Common Object Request Broker Architecture/Corba Component Model J2EE/EJB — ava 2 Plattform, Enterprise Edition/Enterprise JavaBean

Das COM+-Modell wurde von Microsoft definiert und basiert auf einer von Programmiersprachen unabhängigen API (Application Programming Interface). Das benötigte Runtime-Modul wird als Bestandteil der Microsoft Betriebssysteme mitgeliefert. Die Information, wie eine Komponente anzusprechen ist und wo sie im Netzwerk zu finden ist, wird in der Windows Registry hinterlegt. COM+ unterstützt Transaktionssteuerung und Security direkt auf der Kommunikationsebene. Weitere Dienste wie COM-Datenbankzugriffe werden von Microsoft angeboten.