Beitrag für den GI-Workshop: Komponentenorientierte betriebliche Anwendungssysteme am 30.3.1999 an der Universität Magdeburg

Dr. Sybille Möhle: Management von Komponenten

Oracle Deutschland GmbH

Für den Zusammenbau von Software aus Komponenten soll die technische Grundlage nicht ausschlaggebend sein. Entweder ist ein Baustein auf Entwurfssicht bzw. als Sourcecode verfügbar oder vorkompiliert, als ausführbare Datei konzipiert. Identifiziert wird eine Komponente auch durch ihren fachlichen Inhalt. Im folgenden wird diskutiert, was in technischer und fachlicher Hinsicht die Basis für das Management von Komponenten und damit für den Einsatz „barrierefreier", vollständig autarker Komponenten bzw. Standardisierungspotential ist:

1. **Identifikationsdaten** (in Anlehnung an Identifikationsdaten für wiederverwendbare Prozeßbausteine) für Objekte, die zu einer Komponente gehören (vgl. Möhle 1998).

Eine Anwendung aus Komponenten besteht intern wiederum aus einer Menge von Objekten, etwa aus Tabellen, Abfragen, Formularen, Berichten, Makros, Modulen, Datenblättern, Vorgängen, Ressourcen usw. Jedes der neugebildeten Objekte enthält *Identifikationsdaten* (vgl. Abb. 1), welche allgemeine Verwaltungsinformationen beinhalten und die Zugehörigkeit zu einer Komponente ausweisen. Es handelt sich um Identifizierer, Beschreibung, Bezeichnung und die Quelle der Informationen über den fachlichen Inhalt.

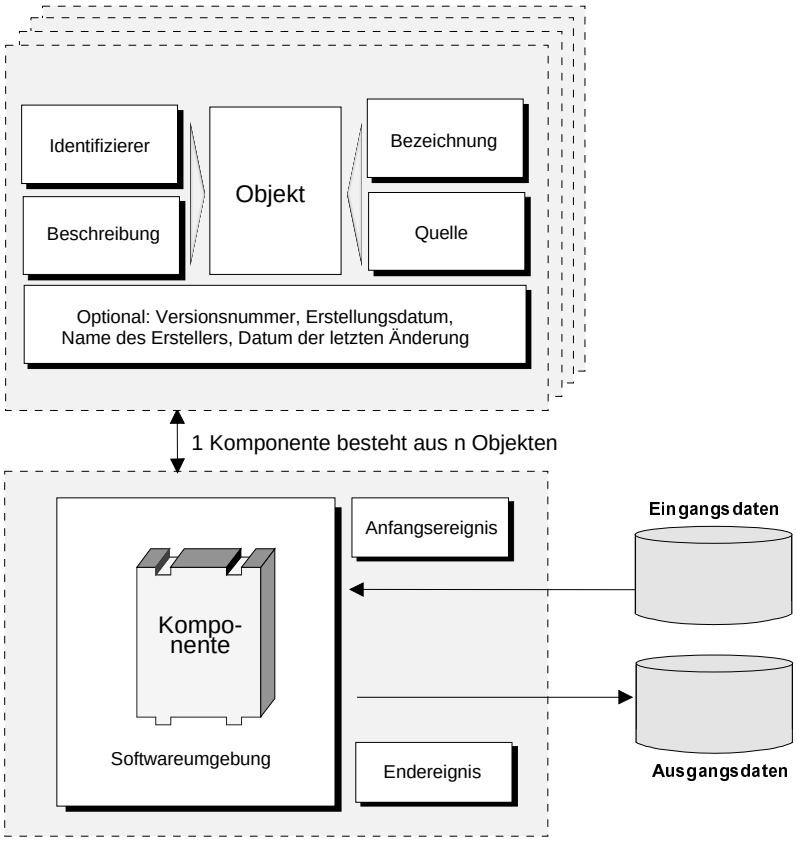

Abb. 1: Identifikationsdaten von Komponenten und Objekten

Die Identifikationsdaten sind auf Objektebene an entsprechender Stelle im Programm-Code abgelegt.

Für eine neugebildete Komponente sollten folgende Daten in einer Dokumentation festgeschrieben sein, um die Wiederverwendbarkeit und die Transparenz zu gewährleisten:

2. Anfangs- und Endereignisse: Jede Komponente weist mindestens ein Anfangs- und ein Endereignis auf. Sie beziehen sich immer auf einen kompletten Baustein und nicht auf untergeordnete Funktionalitäten. So ist eine gewisse Austauschbarkeit gewährleistet, denn die Bausteine werden genau zwischen den beiden Ereignissen eingeklinkt.

3. Aufgerufene Softwarepakete: Softwarepakete und zusätzliche Programmaufrufe für andere Softwarepakete sind deswegen wichtig zu dokumentieren, da diese Basis für das Funktionieren des Bausteins sind.

4. Komponente, die Voraussetzung für einen Einsatz des betrachteten Bausteins ist: Manche Komponenten bzw. deren Ausgangsdaten sind Basis für andere Komponenten, beispielsweise baut die Werkstattsteuerung auf den freigegebenen Aufträgen aus der Auftragsfreigabe auf.

5. Eingangs- und Ausgangsdaten: Je nach Architektur der Komponenten und des Gesamtsystems sind die Daten zentral in einer Datenbank oder in der Komponente selbst gespeichert.

Objekte und damit die Komponenten sowie die Daten können getrennt verwaltet werden: Tabellendatenbank und Objektdatenbank. Die Trennung von Daten und Objekten (vgl. Abb. 2) bietet den Vorteil, daß man verschiedene Objektdatenbanken anlegen kann, die alle auf den gleichen Datenstamm zugreifen, aber unterschiedliche Aufgaben bewältigen können.

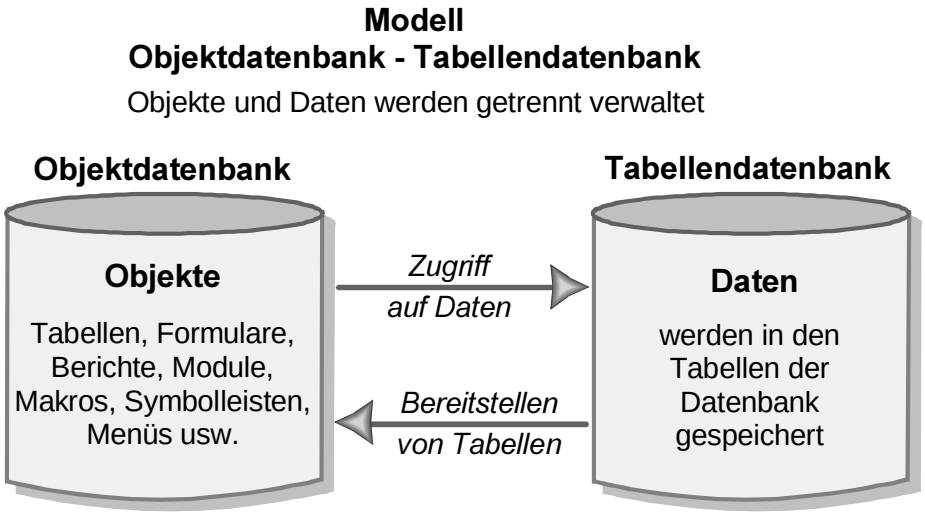

Abb. 2: Trennung von Daten und Objekten

6. *Vorgehensmodelle* bei einer Anwendungssoftwareentwicklung mit Componentware

Nach Bertram (vgl. [Bertram 1997] und [Möhle et al. 1998]) besteht der Prozeß, wie Bausteine entwickelt, klassifiziert und wiedergefunden werden, aus einer bestimmten Folge von Ablaufschritten (vgl. auch Abb. 3):

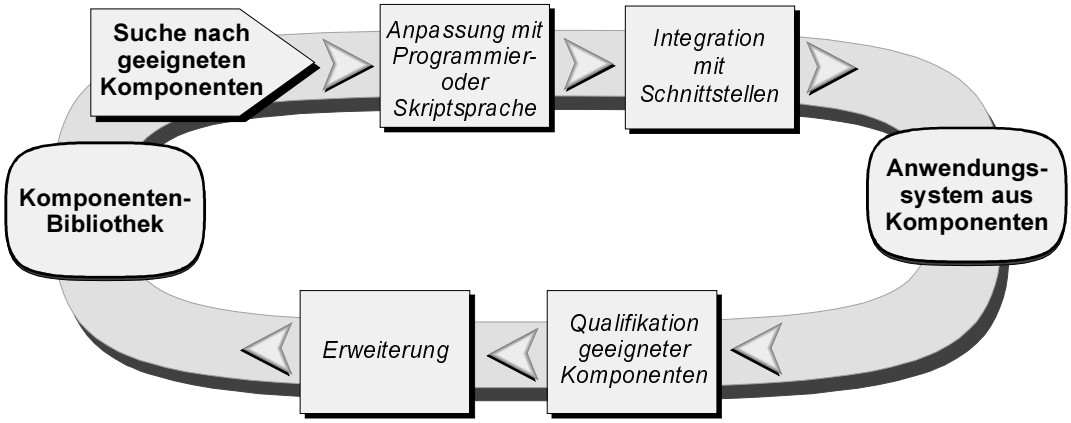

Abb. 3: Allgemeine Vorgehensweise bei einer Anwendungsentwicklung mit Componentware (in Anlehnung an [Bertram 1997]).

7. Ein morphologisches Merkmalsschema für die Anwendungssoftware mit Componentware

Zu Beginn einer Anwendungsentwicklung mit Componentware sind einige grundsätzliche Überlegungen anzustellen, um genauere Details zur Konzeption der Komponenten festzulegen. Als Hilfsmittel kann ein morphologisches Merkmalsschema dienen. Es kristallisieren sich fünf Merkmale heraus:

❶ Zielgruppe, für welche die Komponenten-Software gedacht ist,
❷ Grad der Wiederverwendung,
❸ Flexibilität,
❹ Softwareumgebung im Einsatzgebiet,
❺ Hardwareumgebung im Einsatzgebiet.

Die Merkmale haben Einfluß auf die: *Art der Codierung, Art der Schnittstellen, Art der Anbindung an die vorhandene IV-Umgebung, Architektur der Software, Art der Datenhaltung, Stufe der Granularität*.

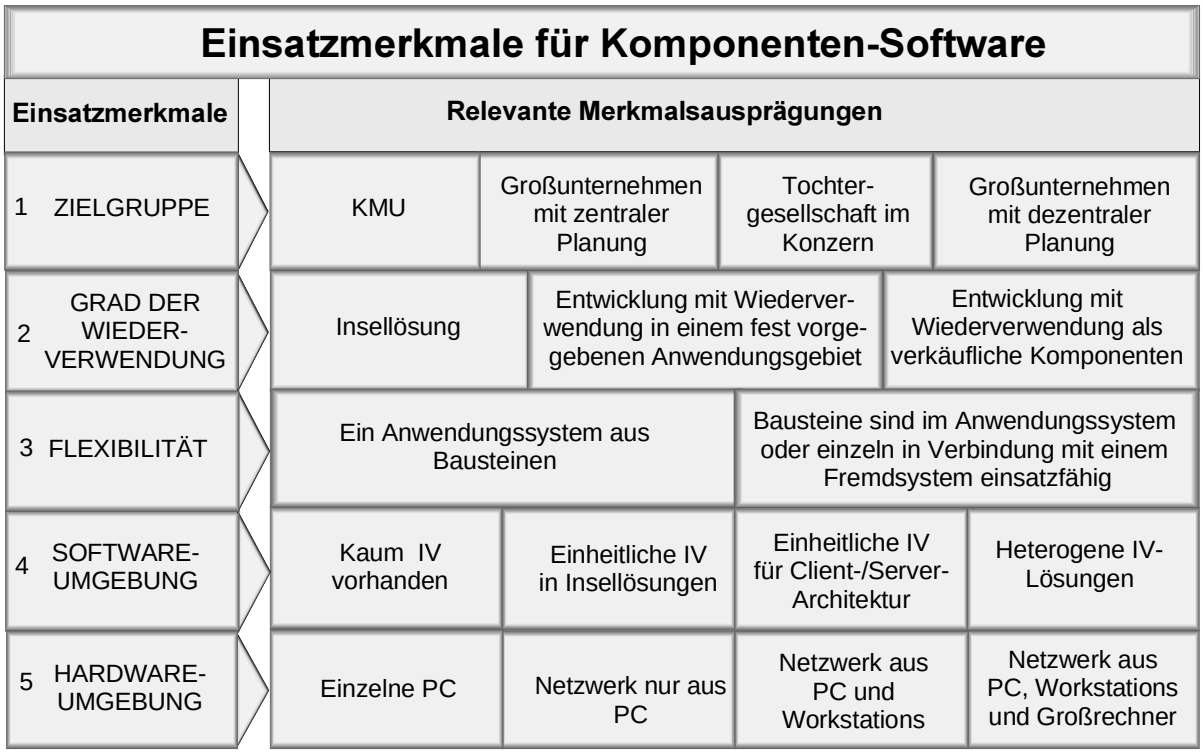

Abb. 4: Morphologisches Merkmalsschema: Einsatzmerkmale und ihre Ausprägungen

8. Ein Repository als Grundlage für das Handling von Komponenten, wie es etwa Uniface (Compuware Corporation) und Oracle anbieten.

Im Oracle Repository sind Objekte, Metadaten und Metamodelle gespeichert und können wiederverwendet werden. Auf Design-Ebene berücksichtigt es Open Java / CORBA API. Das Repository begleitet die Komponenten von der Entwicklung bis zum Einsatz (vgl. Abb. 5).

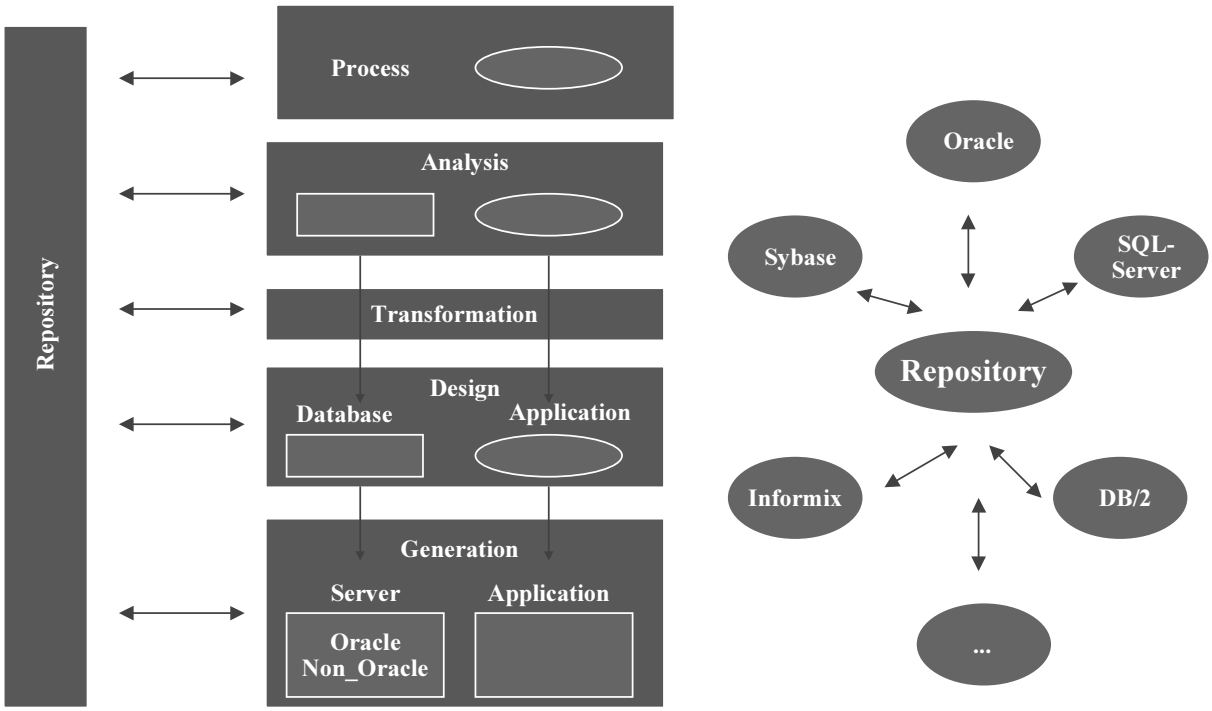

Abb. 5: Repository

Es gibt Pläne, daß das Oracle Repository ein Java-Komponentenmodell zur Verfügung stellt, welches es erlaubt, Java Beans und Enterprise Java Beans im Repository zu speichern. Enterprise Java Beans verwenden Klassen, die Netzwerk-Services wie Sicherheits-, Verzeichnis- und Transaktionsdienste bieten. Der Oracle Application Server ermöglicht es durch die komponentenbasierte Architektur, daß Anwendungen verteilt über heterogene Systemumgebungen agieren (vgl. Abb. 6). Auch ActiveX als weitere „typische“ Ausprägung von Bausteinen sind in Oracle-Software integriert (JDeveloper).

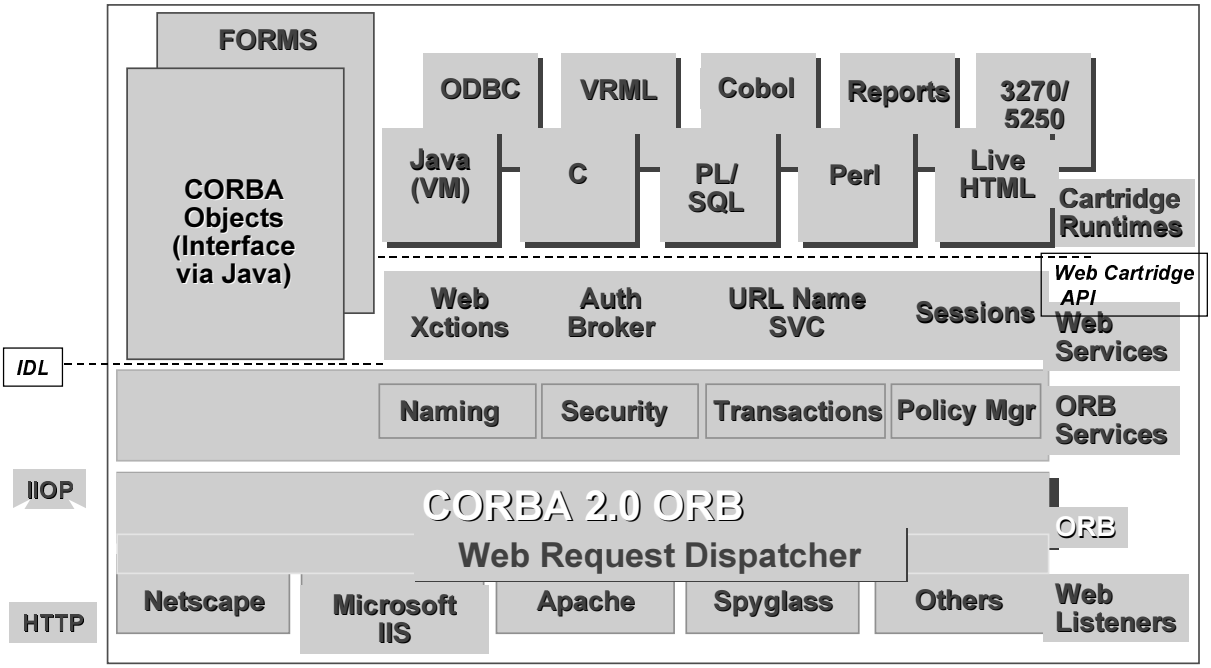

Abb. 6: Application Server

9. Kommunikation der Komponenten untereinander, z. B. Message Passing.

Literatur:

Bertram, M., Wiederverwenden - aber bitte mit System!, Diebold Management Report o.Jg. (1997) 1, S. 7-10.

Möhle, S., Die Entwicklung eines PPS-Systems mit Componentware, Dissertation, Universität Erlangen-Nürnberg 1998.

Möhle,S., Braun,M., Mertens, P., PPS-Systementwicklung mit Componentware, in: Luczak, H. und Eversheim, W. (Hrsg.), Produktionsplanung und -steuerung. Grundlage, Methoden und Konzepte, Berlin u.a. 1998.