Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung
Institut Praktische Informatik

H A U P T S E M I N A R
im WS 02/03

Das CORBA Component Model
(CORBA Components vs. Enterprise Java Beans)

Themenbereich Middleware und Webtechnologien

29.11.2002, CC445

Abstract

Komponententechnologie im Wandel?

- Präsentationskomponenten heute eine „Pflichtübung“ für Entwickler
- Aber: Verschiebung der Betrachtung vom Desktop hin zum Server
- Konzentration auf Abbildung von Prozess- und Geschäftslogik

Bedeutung für die IT-Branche

- Time to Market bei Softwareprodukten (Produktivitätssteigerung)
- Baukastenartige Softwareentwicklung (Anwendungsentwicklung mit „Fertigbauteilen“)
- Hoher Kundennutzen bei Client/Server Architekturen (Skalierbarkeit, Transparenz, Sicherheit, …)

Mögliche Konsequenz: komponentenbasierte IT-Infrastrukturen

Inhaltsverzeichnis

Inhaltsverzeichnis

Allgemeines zu Komponententechnologien 1/4

Ausgangssituation

- Mittelpunkt des Entwicklerinteresses von Desktop auf Server verlagert
- komplexe verteilte Systemumgebungen mit hohen Connectivity - Ansprüchen
- Ein möglicher Ansatz: in Eigenentwicklung die Integrationsaufgaben lösen!

Sinnvoll?

- Problem: sehr hoher Entwicklungsaufwand (Geschäftslogik abbilden!)
- Üblicherweise Aufgabe der Server-Infrastruktur
- Ständig wiederkehrende Problemstellung

- ***Lösung: Middle-Tier-Komponententechnologie***

In Anlehnung an Michael Stal: /Im Reich der Mitte – Die Komponententechnologien COM+, EJB und „CORBA Components“/ veröffentlicht in: OBJEKTspektrum 3/00, S.26 ff., SIGS-Datacom GmbH, Troisdorf 2000

Allgemeines zu Komponententechnologien 2/4

Problemfelder

- Ebene von Geschäftsobjekten und Datenbanksysteme derzeit ohne universellen Standard
- nur ein objektorientierter Kommunikationsmechanismus (RMI, CORBA, SOAP) **reicht nicht aus**, um komplexe Integrationsprobleme zu lösen!

- zu diesen Problem gehören:

  - Zugriffsmanagement
  - Sicherheitsmanagement
  - Lebenszyklusmanagement
  - Konfigurationsmanagement
  - Deployment/Wartbarkeit
  - Unabhängigkeit
  - Anbindung
  - Verteilung
  - Namens- und Verzeichnisdienste
  - Persistenz

In Anlehnung an Michael Stal: /Im Reich der Mitte – Die Komponententechnologien COM+, EJB und „CORBA Components“/ veröffentlicht in: OBJEKTspektrum 3/00, S.26 ff., SIGS-Datacom GmbH, Troisdorf 2000 und Stefan Denninger, Ingo Peters: /Enterprise JavaBeans 2.0/ S. 34, Addison-Wesley Verlag, München 2002

Allgemeines zu Komponententechnologien 3/4

Was sind Komponenten?

"A software component is a static abstraction with plugs." Nierstrasz/Dami

"A reusable software component is a logically cohesive, loosely coupled module that denotes a single abstraction." Booch

•Typische Eigenschaften:

- •Eigenständigkeit – sind fertige (Teil)Programme
- •Grobkörnigkeit – sind groß (im Vergleich zu Objekten)
- •Black Box Prinzip – sind bereits kompiliert und „verpackt"
- •Schnittstellen - Zugriff über definierte öffentliche Pfade
- •Konfigurierbarkeit - Wiederverwendbarkeit erleichtern
- •Verknüpfbarkeit - Ein- und Ausgangsschnittstellen für Events

In Anlehnung an Frank Griffel: /Componentware: Konzepte und Techniken eines Softwareparadigmas/, dpunkt-Verlag, 1.Aufl., Heidelberg 1998

Allgemeines zu Komponententechnologien 4/4

Komponentenarchitektur

- Rahmenarchitektur für Komponenten mit spezieller Laufzeitumgebung
- Ziel ist der Einsatz von verteilten, serverseitigen und möglichst transaktionsorientierten Komponenten
- Architektur soll alle notwendigen Dienste zur Verfügung stellen
- Soll den Einsatz voneinander unabhängiger Komponenten ermöglichen

Anforderungen an eine Komponentenarchitektur

- •Ortstransparenz
- •Trennung von Schnittstelle und Implementierung
- •Unabhängigkeit von der Umgebung
- •Selbstbeschreibende Schnittstellen
- •Integrations- und Kompositionsfähigkeit
- •Plug'n'Play

In Anlehnung an Frank Griffel: /Componentware: Konzepte und Techniken eines Softwareparadigmas/, dpunkt-Verlag, 1.Aufl., Heidelberg 1998

Inhaltsverzeichnis

Das CORBA Component Model 1/5

Defizite bisheriger Modelle

- Distributed Object Computing (CORBA, RMI, DCOM) ohne umfassende Gesamtarchitektur
- unterstützen nicht die Konfiguration und Verteilung von Applikationen
- Explizite Programmierung von struktureller Anforderungen
- Technische Aspekte werden mit Fachobjekten vermischt
- Package- und Deployment-Richtlinien fehlen
- Keine Standards für Installation und Ausführung
- Nur Ad-Hoc Tools und Zwischenlösungen verfügbar

In Anlehnung an Raphaël Marvie, Phillipe Merle: /CORBA Component Model: Discussion and Use with OpenCCM/, Laboratoire d'Informatique Fondamentale de Lille, Villeneuve d'Ascq 2001

Das CORBA Component Model 2/5

Was ist das CCM?

„CCM ist eine Spezifikation für die Erstellung von serverseitigen, skalierbaren, sprachneutralen, transaktionsorientierten, mehrbenutzerfähigen und sicheren Enterprise Applikationen.“ Gopalan Suresh Raj

„The Industry‘s First Multi-Language Component Standard.“ Phillipe Merle

Kernkonzepte

- Eine Architektur zur Definition von Komponenten
- Container Framework für:
  - injecting lifecycle, (de)activation, security
  - transactions, persistence and events
- Interoperationalität für Enterprise Java Beans
- Packaging-Technology (für standardisierte Deployment-Prozesse)
- Vollständig plattform- und sprachneutral

In Anlehnung an G.S.Raj: /CORBA Component Model/ Web Cornucopia, http://my.execpc.com/~gopalan/corba/ccm.html; Abruf 2002-23-11 und Phillipe Merle: /CORBA Component Model Tutorial/ OMG Meeting, Yokohama 2002

Das CORBA Component Model 3/5

Kernkonzepte

- CCM ist Teil der CORBA 3.0 Spezifikation
- Erweitert das ursprüngliche CORBA Objektmodell durch neue Features und Services
- Serverseitiges Modell für d. Erstellung von CORBA-basierten Anwendungen
- Standardisierte Umgebung, mit der Anwendungsentwickler über ein einheitliches System
  - K‘s implemententieren, verteilen, konfigurieren und integrieren
  - und Teile des Codes automatisch generieren können.

Die Rolle der OMG

- OMG Prinzip: Sprachunabhängigkeit - Aufgrund dieser Anforderung ist die Spezifikation grundsätzlich straffer und weniger ausschweifend konzipiert als beispielsweise die JAVA-basierte EJB-Spezifikation
- Sehr starke Ähnlichkeit zu EJB, CCM übernimmt Entwurfsmuster (Design Patterns) und Architekturansätze

In Anlehnung an Dave Bartlett: /CORBA Component Model (CCM) - Introducing next-generation CORBA/IBM developerWorks, http://www-106.ibm.com/developerworks/library/co-cjct6/index.html, Abruf 2002-23-11, New York 2001

Das CORBA Component Model 4/5

CCM Überblick 1/2

Abstract Model	• Interface - Design • Interface Definition Language(IDL) • Port-Definition (Multi-Interfaces)
Programming Model	• Component Implementation Framework (CIF) • (Non)Functional Class Design • Container Zugriffsfunktionen
Packaging Model	• Definiert die Komponenten-Packages • XML basiert • Open Software Description (OSD)

In Anlehnung an Raphaël Marvie, Phillipe Merle: /CORBA Component Model: Discussion and Use with OpenCCM/, Laboratoire d'Informatique Fondamentale de Lille, Villeneuve d'Ascq 2001

Das CORBA Component Model 5/5

CCM Überblick 2/2

Deployment Model	•Definiert einen Prozess •Komponenten-Packages: •Installieren, Einbetten •Integrieren, Pflegen
Execution Model	•Definiert Ausführungsumgebung •Technische Aspekte der Container (Persistenz, Performance, ..)
Meta Model	•MOF Metamodelle der Konzepte •Abstrahiert die CCM Types (Facets, Receptacles, Sinks, ...)

In Anlehnung an Raphaël Marvie, Phillipe Merle: /CORBA Component Model: Discussion and Use with OpenCCM/, Laboratoire d'Informatique Fondamentale de Lille, Villeneuve d'Ascq 2001

Inhaltsverzeichnis

Architekturkonzepte im CCM 1/5

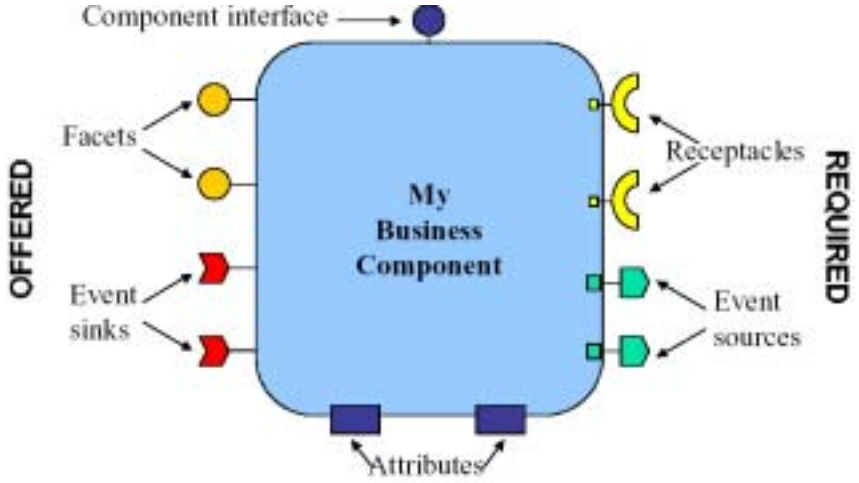
CORBA Components

- Spezifikation einer Komponente per „Component IDL“ (CIDL)
- Schnittstellen werden über die IDL beschrieben
- Auch vielfältige (mehrere) Schnittstellen möglich

- **Facets**
  - Schnittstellen, zum Export von Funktionalitäten
- **Receptacles**
  - Eingänge zum Import von externen Schnittstellen
- **Attributes**
  - Konfiguration der Komponente kombiniert mit Exception-Handling
- **Event Sinks & Event Sources**
  - Ereignisbehandlung und -erzeugung in und durch Komponenten
- **Component Interface**
  - „Äquivalenzschnittstelle“, die die Komponente selbst repräsentiert

Vgl. Phillipe Merle: /CORBA 3.0 New Components/, Chapter 61, OMG TC Document ptc/2001-11-03, Framingham/USA

CORBA Components



Vgl. Phillipe Merle: /CORBA 3.0 New Components/, Chapter 61, OMG TC Document ptc/2001-11-03, Framingham/USA

Arten von Komponenten

- **Service Components**
  - Steht exklusiv **einem** Client zur Verfügung
  - Bei Verbindung wird die Komponente erstellt und anschließend wieder freigegeben
  - Lebenszyklus ist beschränkt auf eine funktionale Anfrage
- **Session Components**
  - Ähnlich einer Service Component
  - Unterschied: mit Zuständen versehen
  - Im Container werden die Zustände **nicht** persistiert
- **Process Components**
  - Zugriff und Nutzung durch mehrere Clients möglich
  - Zustände werden persistiert und dauerhaft festgehalten

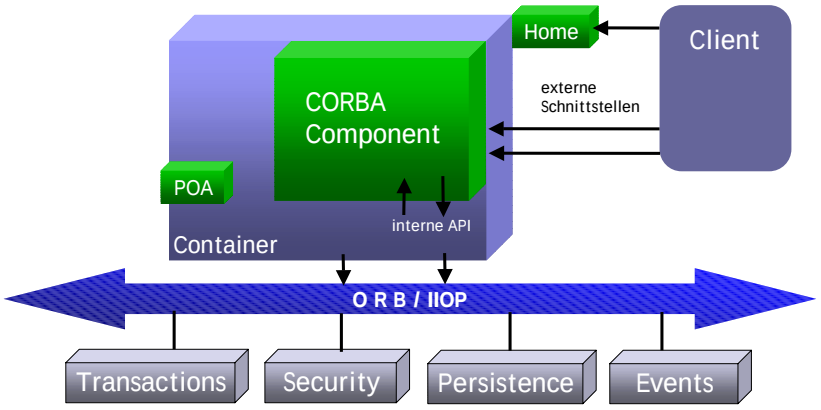
Arten von Komponenten

- **Entity Components**
  - Ähnlich der Process Component
  - Unterschied: jede Entity Component verfügt über einen eindeutigen Primary Key
  - eindeutige Identifikation der Komponente möglich
  - Späteres „Wiederaufwinden“ durch PK

Beispiele

- Service Components : Überprüfung einer Systemaktivität
- Session Components : Aufstellen eine Geschäfts-Bilanz
- Process Components : Geschäftsprozess „Urlaubsantrag“
- Entity Components : „Bankkonto“ oder „Kunde“

Prinzipskizze des CCM



In Anlehnung an Michael Stal: /Im Reich der Mitte – Die Komponententechnologien COM+, EJB und „CORBA Components“/ veröffentlicht in: OBJEKTspektrum 3/00, S.26 ff., SIGS-Datacom GmbH, Troisdorf 2000

Inhaltsverzeichnis

Enterprise Java Beans und CCM 1/3

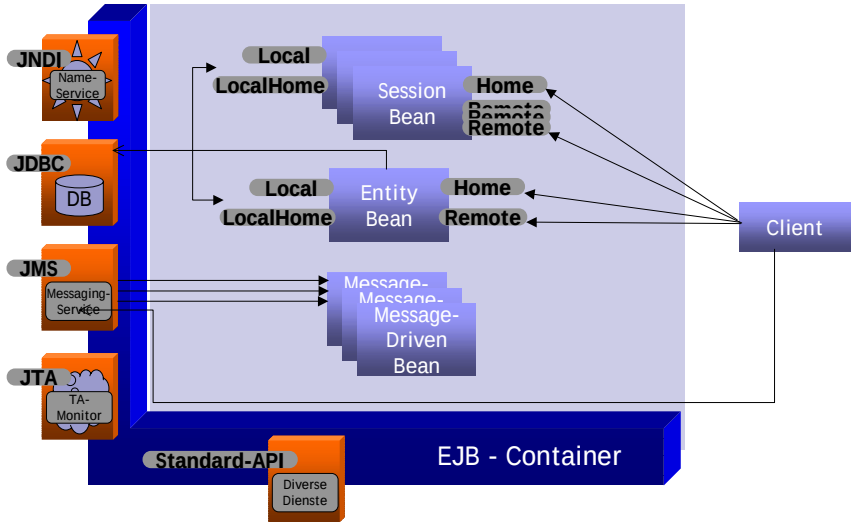
Enterprise JavaBeans (EJB) Überblick

- Bestandteil der J2EE in der Version EJB-Spezifikation 2.1
- Beans = Komponenten (Wortlaut von SUN Microsystems)
- JavaBeans vs. Enterprise JavaBeans (undeployable vs. deployable)
- Zentraler Baustein für die Entwicklung verteilter Applikationen
- Vollständige Ausrichtung auf die Programmiersprache Java

Gründe für den Erfolg von EJB

- Hoher Verbreitungsgrad der Java2 Enterprise Edition
- Bewährte Konzepte bei vertrauten Sprachkonstrukten
- Vielzahl von fertigen Services
- Umfassende Abbildung des Entwicklungskonzeptes
- Großer Beliebtheit bei WebServices

Enterprise JavaBeans (EJB) Gesamtüberblick



Vergleich

	CCM	EJB
Architekturkonzept	Container & Interception	Container & Interception
Schnittstellen pro Komponente	mehrere Interfaces über Ports realisiert (Facets&Receptacles)	ein allumfassendes Interface
Komponentenarten	Service-, Session-, Process-, Entity-Component	Session-Bean, Message-Driven-Bean, Entity-Bean
Metadaten der Komponenten	XML basierte Deskriptoren „.car“	XML basierte Deskriptoren „.ear“
Aktivierung	Container & POA	Container
Middleware	CORBA ORB/IIOP	RMI / RMI-IIOP
Persistenz	Persistent State Service (PSS)	Java DataBase Connectivity (JDBC)
Namensdienst	CosNaming	Java Naming and Directory Interface
Integrations-möglichkeiten	EJB to CORBA Mapping explizit in der Spezifikation	Remote Invocation Interoperability – explizit in der EJB-Spezifkation
Sicherheit	CORBA Security Levels	Java Security Package
Installation	Installation Deployment Process	Herstellerabhängige Prozesse

Inhaltsverzeichnis

Fazit

Fazit

- CCM ist ein universeller & umfassender Ansatz für moderne komponentenbasierte Softwareentwicklung
- Komplexer & vollständiger Standard mit sehr hohem Potential
- Stärke der CCM Spezifikation: Sprachneutralität
- CCM hat starke Ähnlichkeiten mit den Architekturkonzepten von EJB

Thesen

- CCM und EJB lassen sich *„als unterschiedliche Seiten der selben Medaille betrachten“* Michael Stahl
- Spezifikation (CCM) vs. Technologie (EJB) ?
- OpenCCM als erste vollständig implementierte OpenSource Lösung

Vielen Dank für Ihre Aufmerksamkeit!