Vorlesung
Software aus Komponenten

4. Moderne Komponentenkonzepte

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2010/11

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Komponenteneinsatz auch auf der Ebene der Applikationsentwicklung
  - Applikationsdienst wird im Zusammenwirken mehrerer Komponenten (diese werden **Applikationsobjekte** genannt) erbracht, die **innerhalb** der Applikation als Kontext konfiguriert werden.
  - dabei Rückkehr zu plain old Java Objekten (POJO)
- Schichtenmodell mit Standardisierung auf verschiedenen Abstraktionsebenen in eigenen Teilprojekten
  - Konfiguration und Basisarchitektur: Spring Core
  - Kommunikation und Datenaustausch: Spring Integration
  - Prozessmodellierung: Spring Web Flow
  - Benutzerschnittstellen (UI)-Entwicklung: Spring Faces, Spring Web
  - Verteilung und Performance: Spring JEE
  - Datenbankschicht: Spring ORM, Spring DAO
  - Querschnittsfunktionen: Spring AOP

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Komponenten **und** Kontext (als leichtgewichtiger Container) sind Gegenstand der Entwicklung
  - Lösung der Applikation von der inhärenten Bindung an einen J2EE-Server
  - Komponente trifft keine unnötigen Annahmen über den Kontext
    - Applikationsobjekte werden zur Laufzeit von außen konfiguriert.
    - Konfiguration der Komponente transparent für den Anwender, er hat Zugriff auf die Dienstschnittstelle, nicht aber auf die Zuordnung der Ressourcen. Entsprechende Konfigurationsmethoden sind nur auf der Implementierungsklasse bekannt.
    - Explizites Deployment wie bei EJB wird damit von vornherein vermieden.
  - Nutzung von Dependency Injection (Inversion of Control) und Annotationen zur Kommunikation zwischen Komponente und Kontext

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Spring bietet speziellen Support für typische Laufzeitumgebungen an
  - Spring MVC – Framework für Webanwendungen
  - Spring Web Flow – Framework für Abläufe auf einer Web-Site
  - Spring Security (vormals Acegi) – Framework für Sicherheit
  - Spring Rich Client – Framework zur Verteilung der Dienst-erbringung auf Server und Client
- Unterstützung aspektorientierte Ansätze
  - Idee: Basisdienste (cross cutting concerns) werden vom Kontext angeboten und über entsprechende deklarative Schnittstellen (Interzeptoren) in die Komponente eingebunden.
  - Spring bietet eigene Annotationen u.a. im Bereich Transaktionen und Bindung an die Persistenzschicht.
- Fazit: Ein und dasselbe Komponentenmodell kann sowohl für grob granulare Fassadenkomponenten (etwa EJB) wie auch für fein granulare Applikationsobjekte verwendet werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Das CORBA Komponentenmodell (CCM)

- mit CORBA 3.0 endgültig spezifizierte ambitionierte (logische) Erweiterung des EJB-Ansatzes
  - Aktuell CCM 4.0 (April 2006)
- CCM-**Anwendung** besteht aus CCM-**Komponenten**
  - EJB erfüllen die CCM-Komponenten-Spezifikation
- CCM-Komponenten sind in **Komponentenpaketen** zusammengefasst
- CCM-**Sammlungen** (CCM assemblies) enthalten Komponentenpakete zusammen mit einer **Beschreibung** der Abhängigkeiten und der Montage-Beschreibung im XML-Format
- Eine CCM-Komponente kann aus mehreren **Segmenten** bestehen
  - CCM-**Laufzeitumgebungen** laden Anwendungen segmentweise

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Das CORBA Komponenten-Entwicklungsmodell

- Mit Komponentenmodell wird auch ein Entwicklungsmodell mit vier Ebenen spezifiziert
  - **Abstract Component Model**: Äußere Eigenschaften von CCM Komponenten und Definition von Komponententypen in IDL
  - **Component Implementation Framework**: Programmiermodell zur Erstellung von Komponentenimplementierungen sowie zur Beschreibung der Relationen zu Implementierungen anderer Komponenten
  - **Container Programming Model**: Beschreibung der Container-Architektur und der Schnittstellen zum ORB und zu den Komponenten.
  - **Packaging and Deployment**: Verpacken von Komponenten und Assemblies, Spezifikation von Deskriptoren sowie des Deployment-Vorgangs

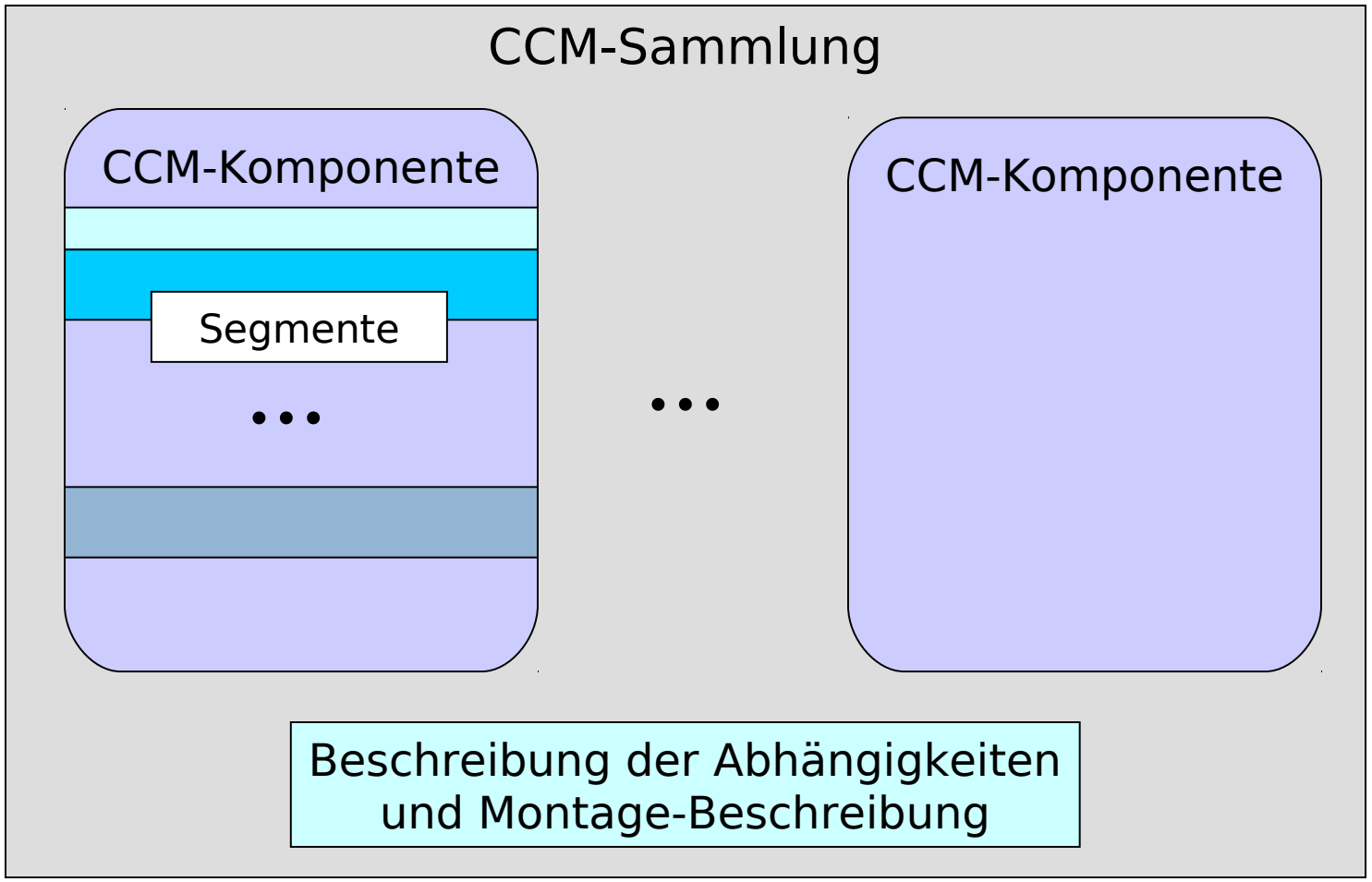
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

CCM in der Praxis

- Im Gegensatz zu Spring ist CCM auf grob granulare Komponentenstrukturen der Anwendungsschicht ausgerichtet und nicht auf Zusammenarbeit von leichtgewichtigen Komponenten innerhalb eines Dienstes
- CCM-Anwendungen laufen nur mit CORBA-3-konformen ORBs
  - wird auch auf der Client-Seite benötigt, wenn die ganze CCM-Funktionalität (etwa Navigation) ausgenutzt werden soll
  - CCM-Standard unterstützt aber abgerüstete Klienten auf pre-CORBA-3-Plattformen (component-unaware clients)
- Es gibt im Gegensatz zu J2EE/EJB und .NET bisher kaum Plattformen, ORBs oder Applikationsserver, die CCM vollständig unterstützen.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Grundstruktur einer CCM-Sammlung



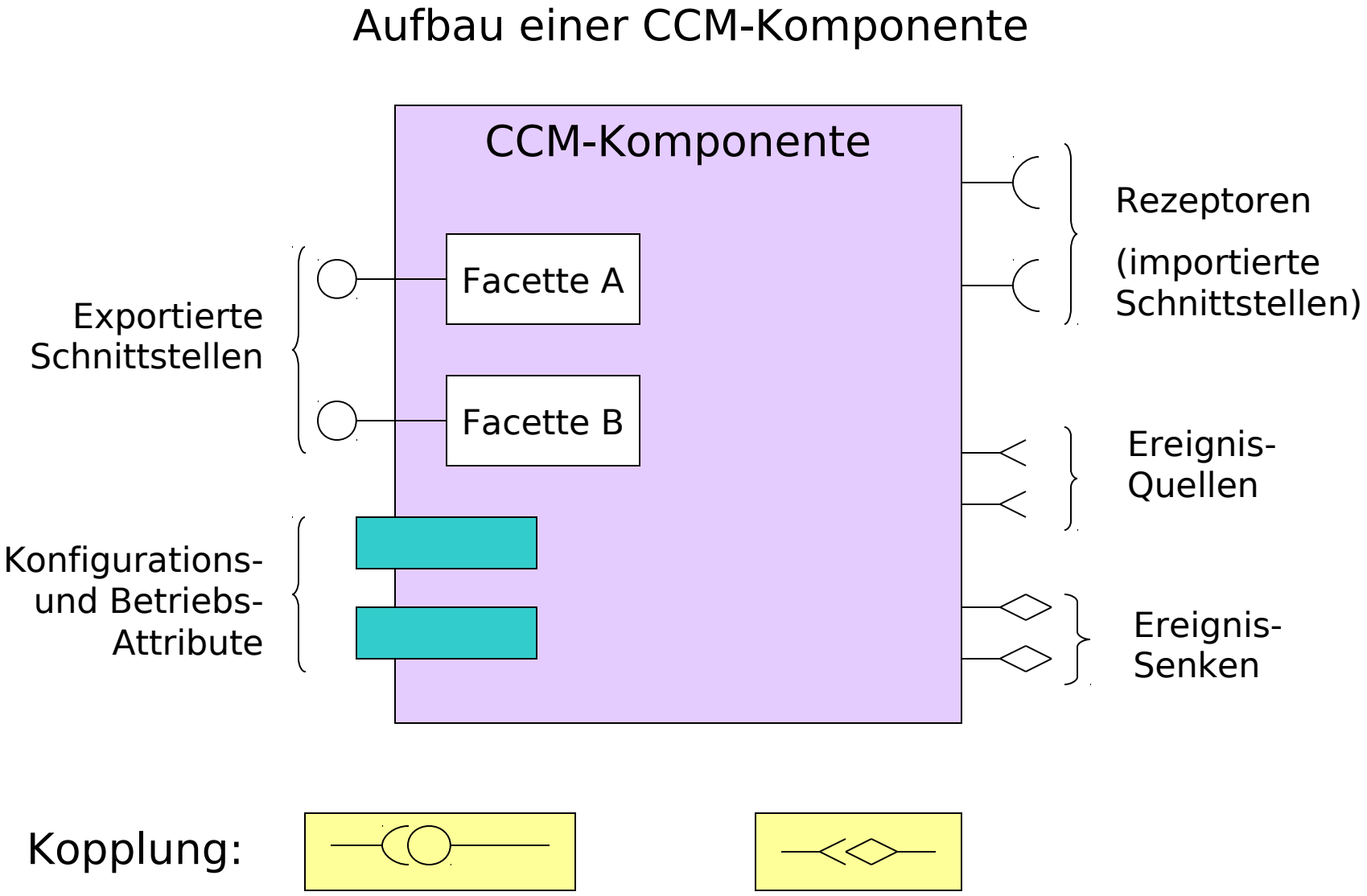
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

CCM-Kategorien

- CCM-Komponenten werden (ähnlich EJB) in **Kategorien** eingeteilt
- **Service-Komponenten**
  - Instanzen sind Aufrufen zugeordnet und speichern keine Zustände über Aufrufgrenzen hinweg
- **Session-Komponenten** (= stateful session EJB)
  - Verwaltung des Zustands innerhalb eines Transaktionszyklus (transactional session)
- **Entity-Komponenten** (= entity EJB)
  - Instanzen haben persistenten Zustand, entsprechen Datenbankeinträgen
  - können über Primärschlüssel aus einer Datenbank gefunden werden
- **Prozess-Komponenten**
  - persistent, Lebensdauer an die des Prozesses gebunden, der bedient wird

- CCM-Anwendung enthält deklarative Informationen über Komponentenkategorien und Komponentenaufgaben

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Ports von CCM-Komponenten

- **Facetten** (facets)
  - exportierte Schnittstelle, gewöhnlich einem Teilobjekt der Komponente zugeordnet
- **Rezeptoren** (receptables)
  - importierte Schnittstellen, intern Referenzen auf externe Objekte, die zum Komponentenbetrieb benötigt werden
  - connect / disconnect Operationen
  - können explizit in der Montage-Beschreibung gefordert oder zur Laufzeit eingebunden werden
- **Ereignisquellen** (event sources) und **Ereignissenken** (event sinks)
  - durch Ereigniskanäle zu verbindende Ports
- **Primärschlüssel** (nur Entity-Komponenten)
- **Konfigurations-** und **Betriebs-Attribute**
  - benannte Werte, die über **Zugriffsfunktionen** (accessor) oder **Modifizierer** (mutator) nach außen sichtbar sind

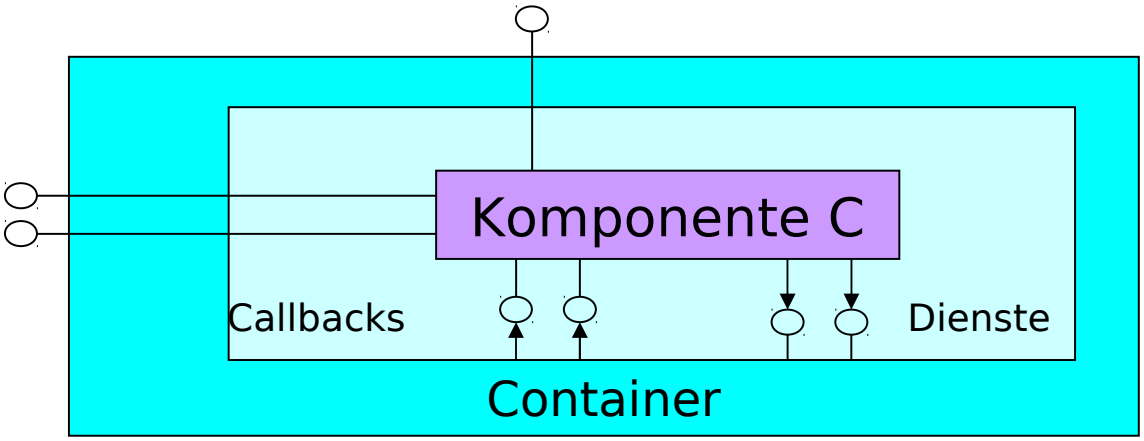
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

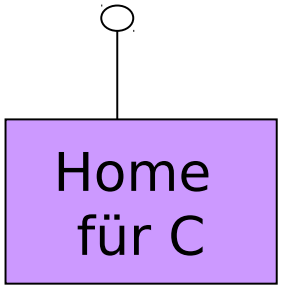
- **Home-Schnittstelle**, über welche die Komponenten-Factory erreicht werden kann
  - in der Komponenten-Klasse implementiert
  - also Komponentenbegriff verschieden von dem in der Vorlesung
  - Management des Lebenszyklus von Komponenten-Instanzen
- Spezielle Facette **E-Schnittstelle** (equivalent interface), über die zwischen den Facetten der Komponente navigiert werden kann
  - Clienten müssen CORBA-3 unterstützen, um diese Navigationsmöglichkeiten auszunutzen
- **Konfigurations-Schnittstelle** (configuration interface)
  - Unterstützung der initialen Konfiguration neuer Komponenten
  - spezielles call-Signal schließt die Konfigurationsphase ab
  - erst danach sind Aufrufe der operationalen Schnittstellen möglich, Aufrufe der Konfigurations-Schnittstelle dagegen untersagt

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

CCM-Container

- CORBA 3 definiert ein **Komponenten-Implementierungs-Gerüst** (component implementation framework, CIF)
  - Generatoren erzeugen aus Eingaben im **CIDL-Format** (component implementation description language) Code, der den Komponentencode ergänzt
- Jede Komponenten-Instanz ist in einem **CCM-Container** untergebracht, über den die Anbindung der Facetten und Rezeptoren erfolgt. Rezeptoren und Dienste können in einem solchen Container per Callback gebunden sein.





- Der CCM-Container ist ein spezieller POA

Vorgefertigte Basisdienste (pre-packaged object services)

- **Transaktionsdienst**: durch Container oder selbst
  - Komponente: Beschreibung enthält die Transaktionsanforderungen (supported, required, required new, not supported)
  - Container: Ausführung der Transaktionen entsprechend der Spezifikation der einzelnen Komponenten
- **Persistenzdienst**: durch Container oder selbst
  - Komponente: Beschreibung der Anforderungen im PSDL-Format (persistent state description language)
- **Sicherheitsdienst**:
  - Zugriffsrechte können im CIDL-Format beschrieben und durch den Container geprüft werden
  - Benachrichtigungsdienst: Aufbau und Verwaltung von Ereigniskanälen