Vorlesung
Software aus Komponenten

3. Komponentenmodelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2004/05

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponentenmodelle auf Quellcode-Ebene: Aufbau von Anwendungen aus Software-Bausteinen

Ziel: Sicherung plattform- und sprachübergreifender Kompilierungskompatibilität

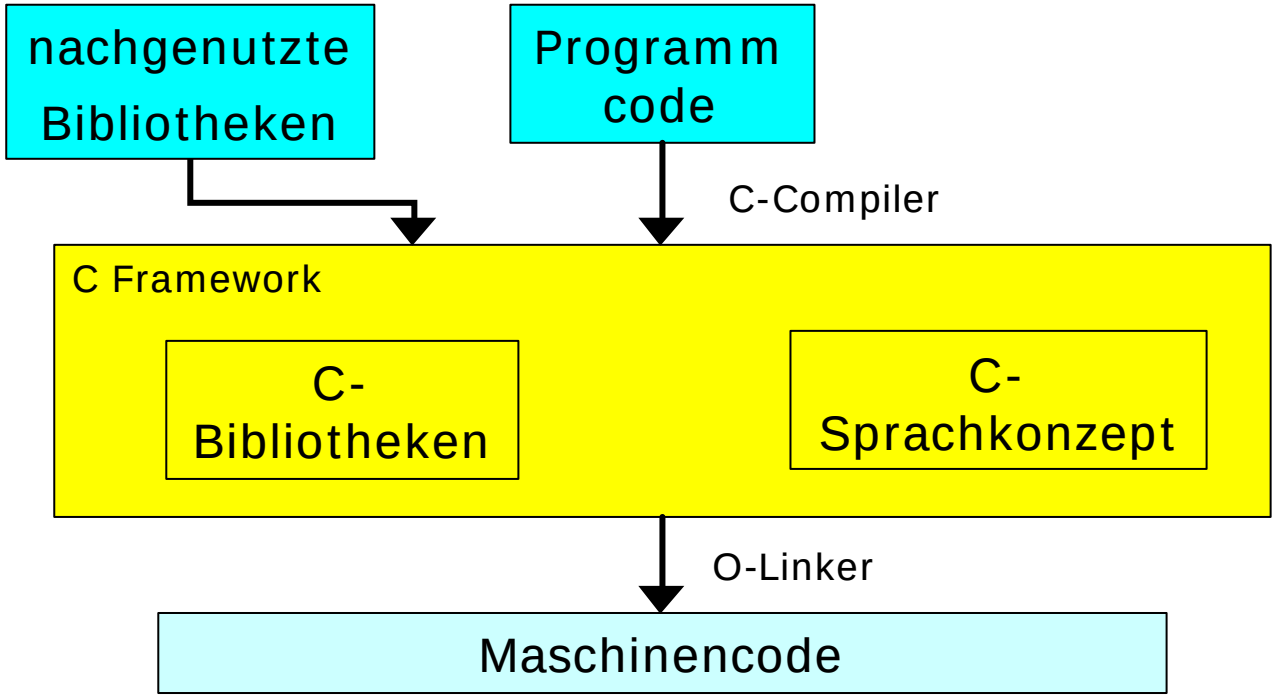
Anwendungsbereich: Desktop, Basiskomponenten

Grundlage: Gemeinsame Designprinzipien

Beispiele: C, Java, .NET

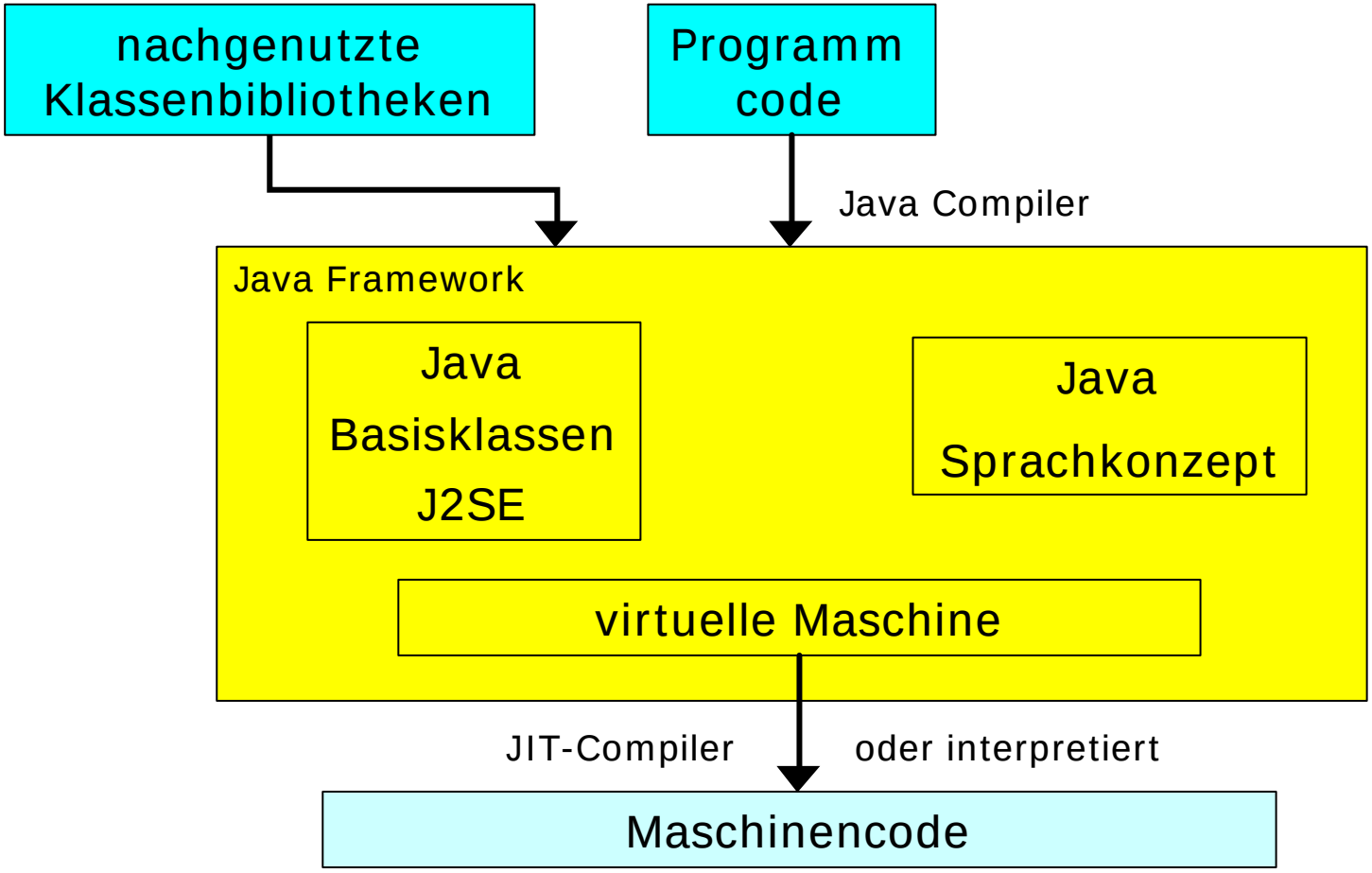
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Eine Sprache, eine Plattform: C



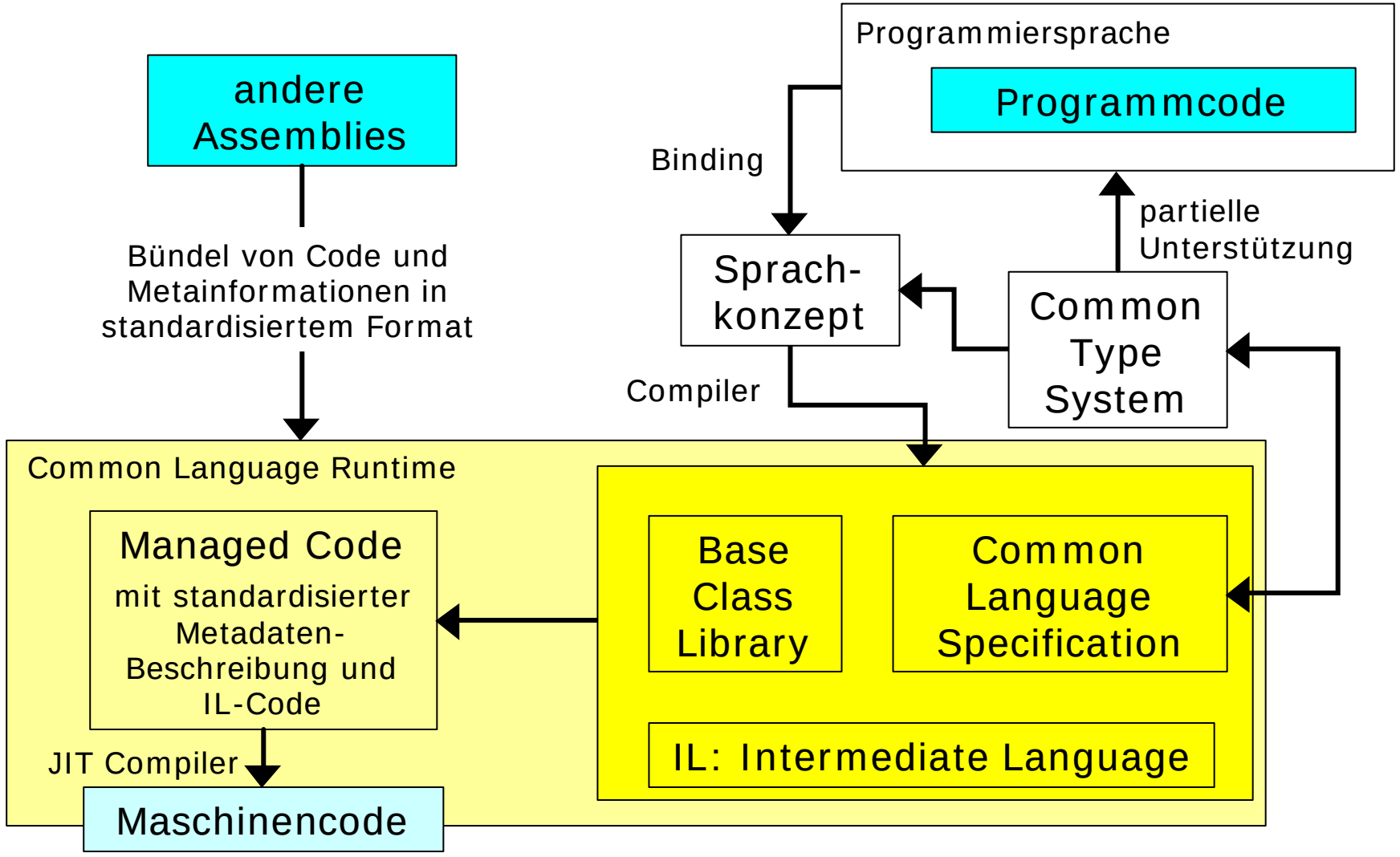
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Eine Sprache, mehrere Plattformen: Java



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Mehrere Sprachen, mehrere Plattformen: .NET



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponentenmodelle für verteilte Anwendungen: Aufbau von Anwendungen aus Komponenten

Ziel: Integration von Diensten in eine standardisierte verteilte Infrastruktur

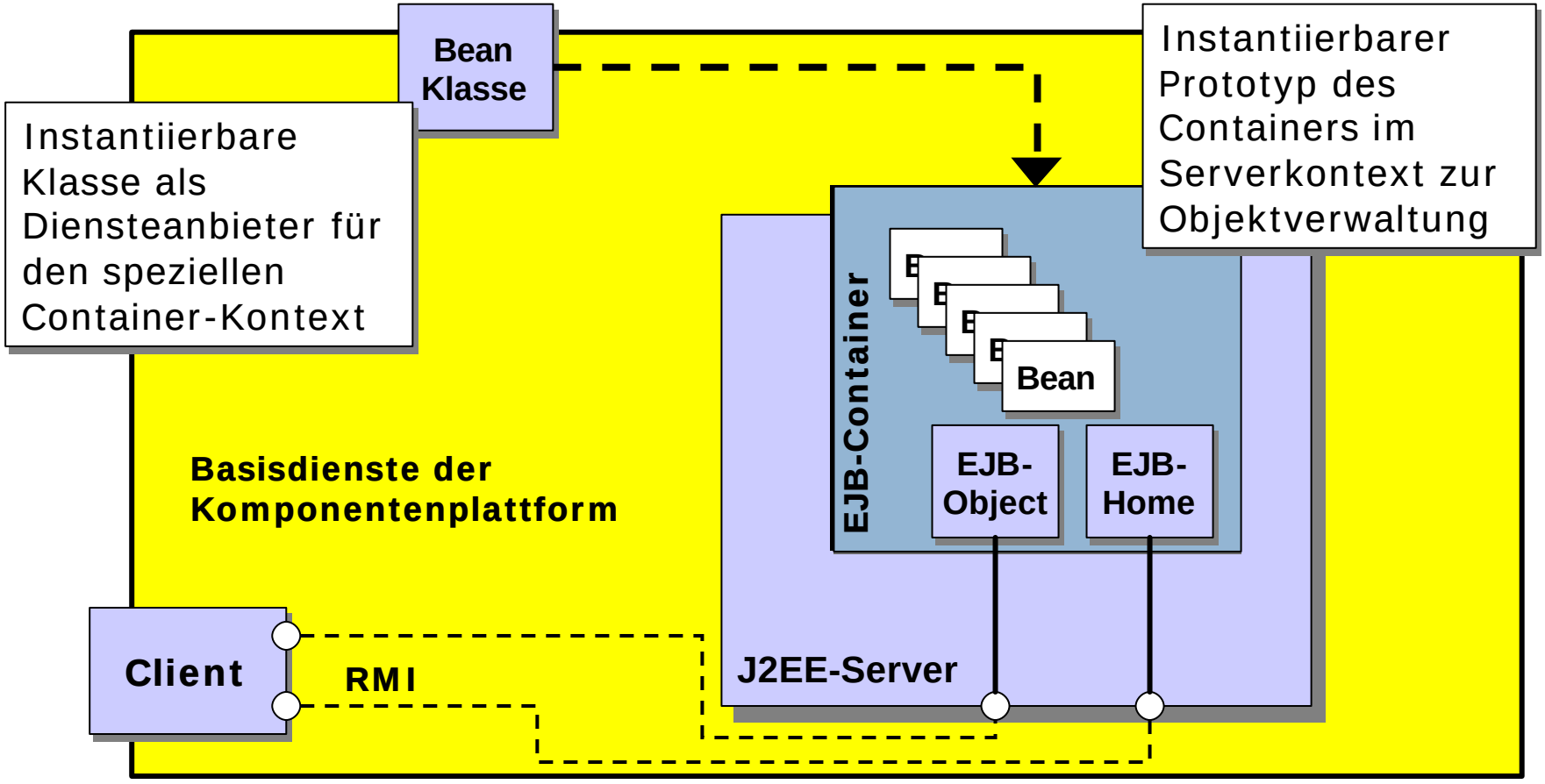
Anwendungsbereich: Middleware und verteilte Systeme

Grundlage: eng gekoppelte Client-Server-Architektur, gemeinsames Framework

Beispiele: EJB, Servlets, Server Pages, DCOM,

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Prototypischer Aufbau am Beispiel EJB



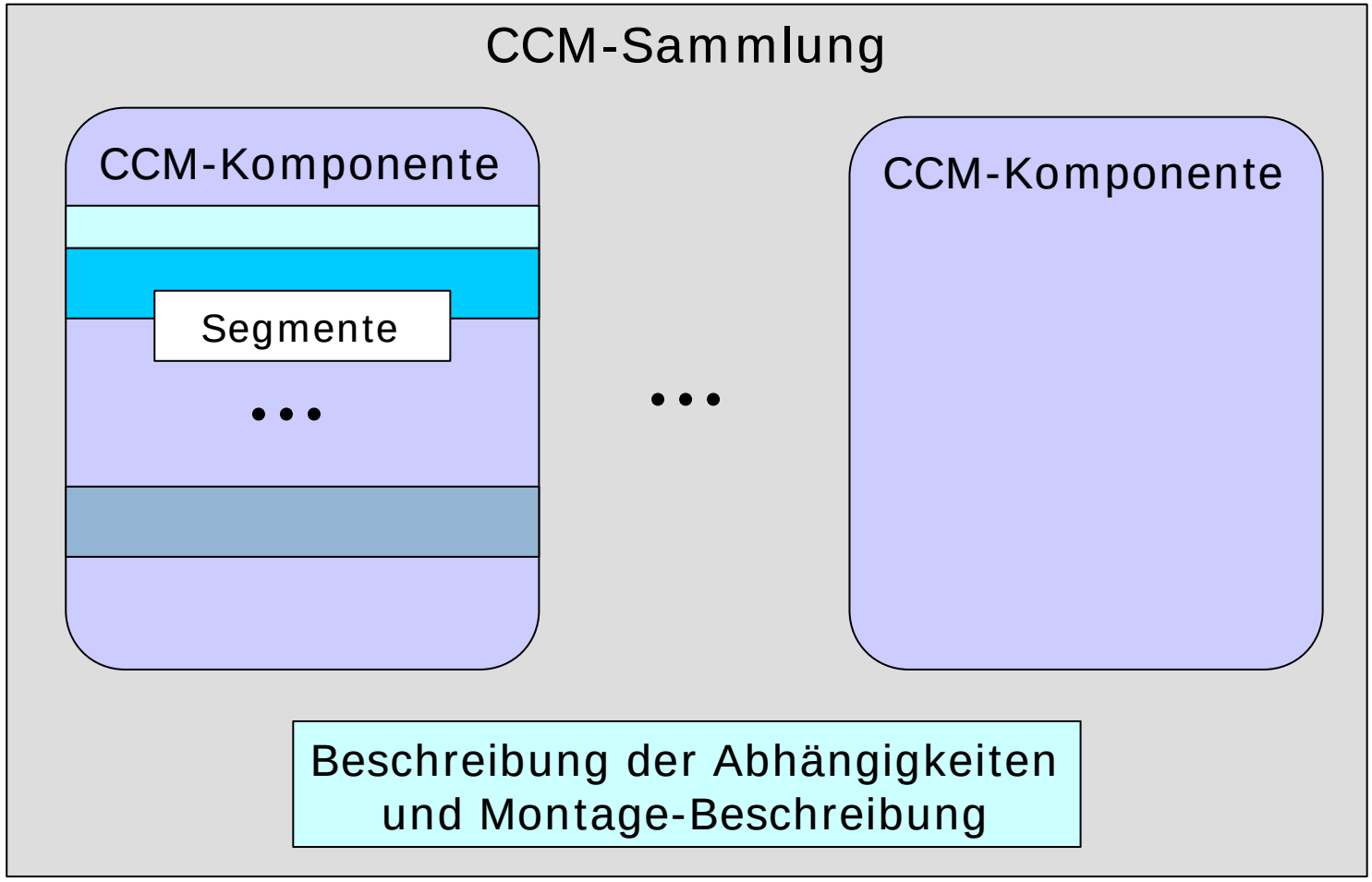
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Weiterentwicklung: Das CORBA Komponentenmodell (CCM)

- mit CORBA 3.0 endgültig spezifiziert
- Ambitionierte (logische) Erweiterung des EJB-Ansatzes
  - derzeit allerdings vor allem auf dem Papier
- 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
- 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)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Grundstruktur einer CCM-Sammlung



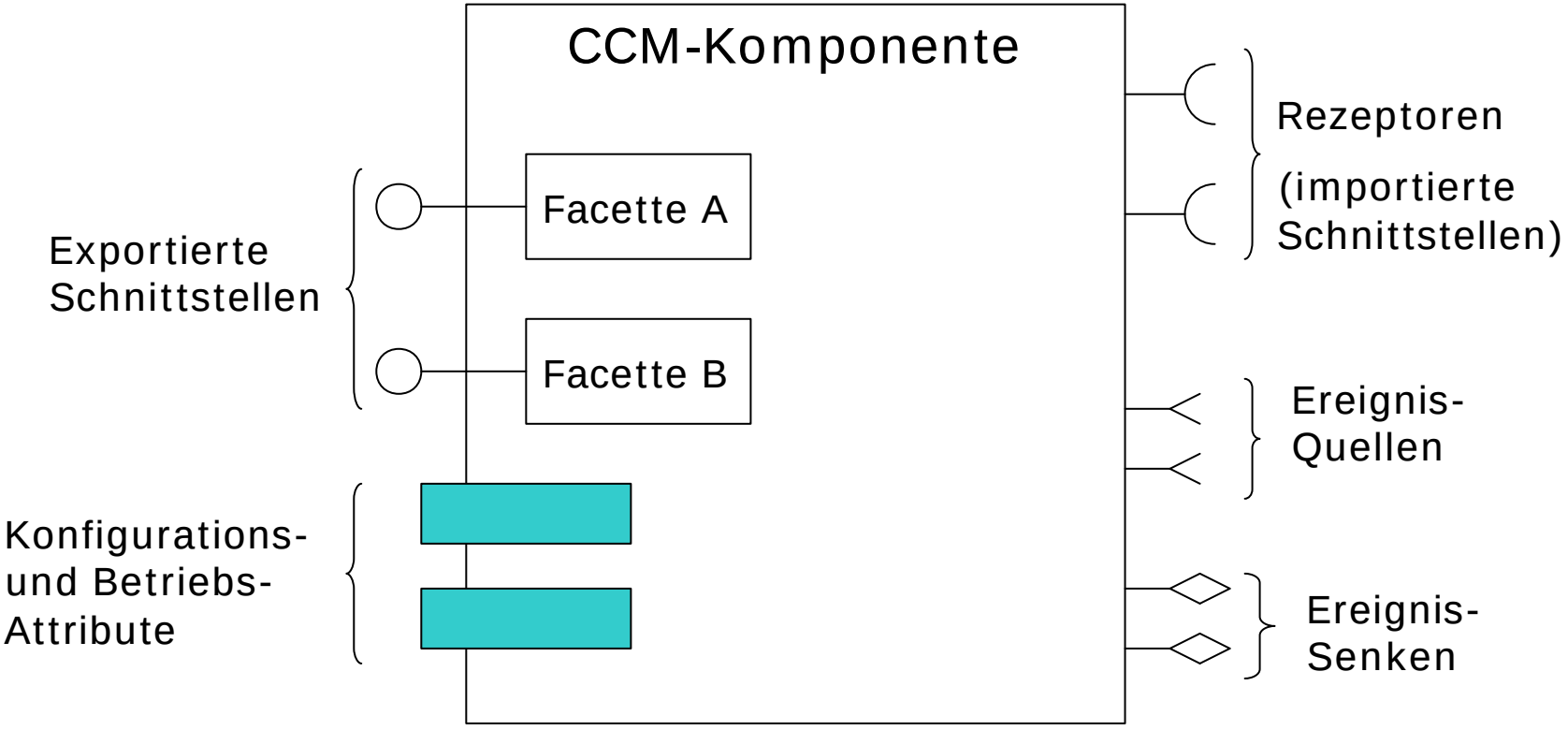
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

CCM-Kategorien

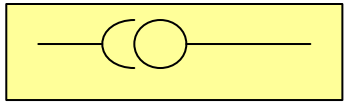
- CCM-Komponenten werden (ähnlich EJB) in **Kategorien** eingeteilt
  - **Service-Komponenten**
    - Instanzen sind Aufrufen zugeordnet und speichern keine Zustände über Aufrufgrenzen hinweg
  - **Session-Komponenten** (= stateful session EJB)
    - Verwaltung des Zustands innerhalb eines Transaktionszyklus (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

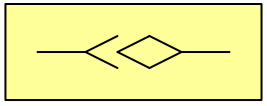
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Aufbau einer CCM-Komponente



Kopplung:





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

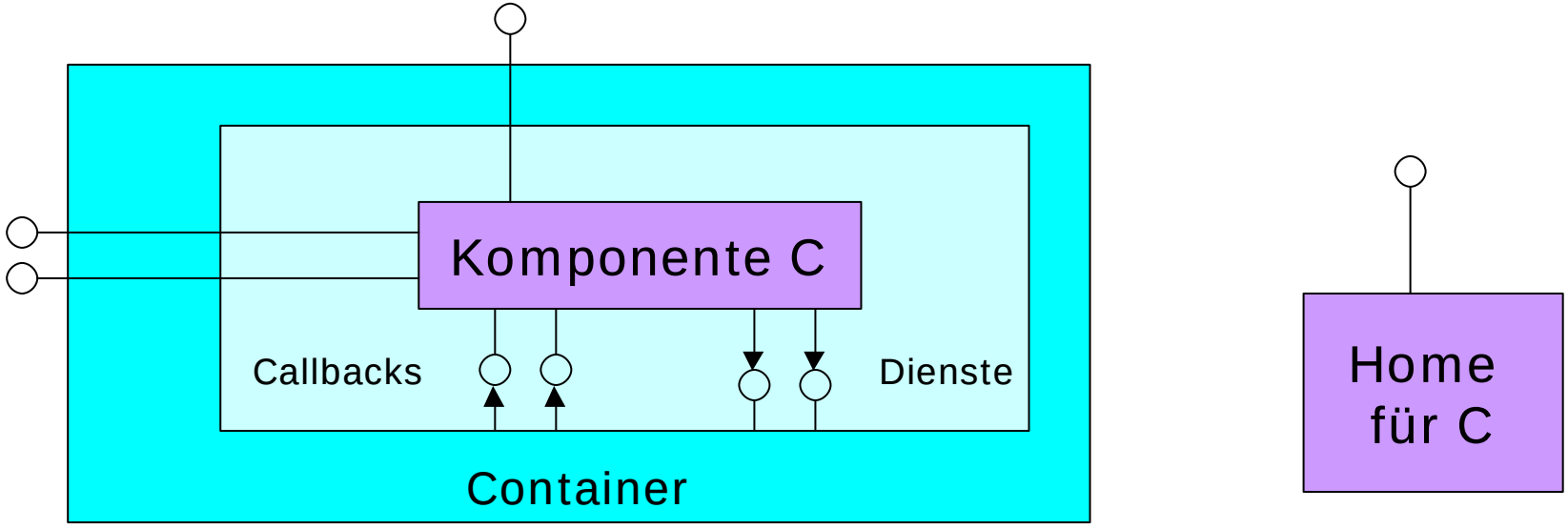
Ports von CCM-Komponenten (Fortsetzung)

- **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
  - analog der IUnknown-Schnittstelle im COM-Konzept
  - 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.



- Navigation zwischen den Schnittstellen aller Komponenten im Container
  - Container hat Kontrolle über exportierte und importierte Schnittstellen seiner Komponenten
  - kann nur von CORBA-3-konforme Clients genutzt werden

- CCM-Container ist ein spezieller POA mit vorgefertigten Basisdiensten (pre-packaged object services)
  - **Transaktionsdienst**: durch Container oder selbst
    - Komponente: Beschreibung enthält die Transaktionsanforderungen (supported, requireed, 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 (pesistent state description language)
  - **Sicherheitsdienst**: Zugriffsrechte können im CIDL-Format beschrieben und durch den Container geprüft werden
  - **Benachrichtigungsdienst**: Aufbau und Verwaltung von Ereigniskanälen

Zusammenfassung

- Entwicklungsumgebung, die auf Programmierer von Geschäftsanwendungen ausgerichtet ist
  - Umgebung, in welcher diese ihre speziellen Erfahrungen bestmöglich entfalten können
  - Konzentration auf die Logik auf Geschäftsprozess-Ebene möglich
- Stellt eine strukturierte CORBA-Laufzeit-Infrastruktur (packaged CORBA-computing infrastructure) zur Verfügung
  - höhere Abstraktionen für Persistenz, Transaktionalität, Sicherheit, Ereignisbehandlung
  - Zusammenbau durch visuell orientierte oder Scripting-Werkzeuge (CORBA scripting language)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponenten: Konzepte und Anforderungen

Eine Zusammenfassung

- Komponententechnologie und Softwaretechnik
- Komponentenkonzepte im Vergleich
- Konvergenz der Konzepte
- Differenzen der Konzepte

Softwaretechnik als Ingenieurtechnik

Ingenieurtechnik

- Standards, Vorgehensweisen und Zusammenhänge, die beim Bearbeiten einer Aufgabenstellung aus dem jeweiligen Gebiet von einer qualifizierten Fachkraft zu berücksichtigen sind.
- technologische Einbettung der für das jeweilige Gebiet verfügbaren Technik

Softwaretechnik ist eine ingenieurtechnische Disziplin

- Lehre von Planung, Erstellung, Einsatz, Wartung und Weiterentwicklung von komplexen Software-Systemen in einem arbeitsteiligen Prozess
- und den dabei zweckmäßig zum Einsatz kommenden Prinzipien, Methoden und Werkzeugen. [Balzert]

- Im Zentrum steht dabei die **Beherrschung der Komplexität** der Anforderungen aus dem Lebenszyklus von Software-Systemen.
- Als typische Arbeitsschritte haben sich bewährt
  - Anforderungsanalyse, Entwurf, Modellierung, Realisierung, Montage, Einsatz

Komponententechnologie aus ingenieurtechnischer Sicht

- Die **Nutzung von Komponenten** ist ein Charakteristikum jeder entwickelten Ingenieurtechnik
  - Neue Produkte werden aus vorgefertigten, standardisierten, dem Stand der Technik entsprechenden Bestandteilen nach allgemein anerkannten Standards und eigener Kreativität zusammengebaut.
  - Form der Komplexitätsreduktion
- **Komponententechnologie** hat zum Gegenstand das Zusammenspiel von Komponentenentwicklung und Komponenteneinsatz
  - Rolle: **Komponentenentwickler**, Perspektive: Zulieferer-Sicht
    - Komponenten für möglichst breites Einsatzfeld entwickeln
  - Rolle: **Komponentenmonteur**, Perspektive: Dienstleister-Sicht
    - Komposition von Anwendersystem aus geeigneten Komponenten
- Ansatz findet über mehrere hierarchische Ebenen der Komposition statt
  - Treiber – Betriebssysteme
  - Laufzeitbibliotheken – Hochsprachen-Programme
  - der in dieser VL besprochene Komponentenbegriff

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponententechnologie und Softwaretechnik

- **Ziel**: Montage eines IT-Systems, das als **verteilte Anwendung** auf einem System von mehreren miteinander verbundenen Rechnern aus **Komponenten unterschiedlicher Hersteller** konzipiert ist.

- ***Anforderungen:***
  - formal fundiertes **Komponentenkonzept** als Basis
  - **Beschreibungstechniken** für derartige Komponenten
  - Entwicklung eines **Prozessmodells** zur Entwicklung, Verwaltung und Zusammensetzung von Komponenten
    - Unterstützung der Zuweisung verschiedener Rollen
  - **Werkzeuge**, welche die Beschreibung und das Prozessmodell unterstützen
    - zur Systemgenerierung selbst
    - zur Dokumentation
    - zur Verifikation und Sicherung wichtiger und kritischer Systemeigenschaften