Vorlesung
Software aus Komponenten

4. Neuere Entwicklungen

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2014/15

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

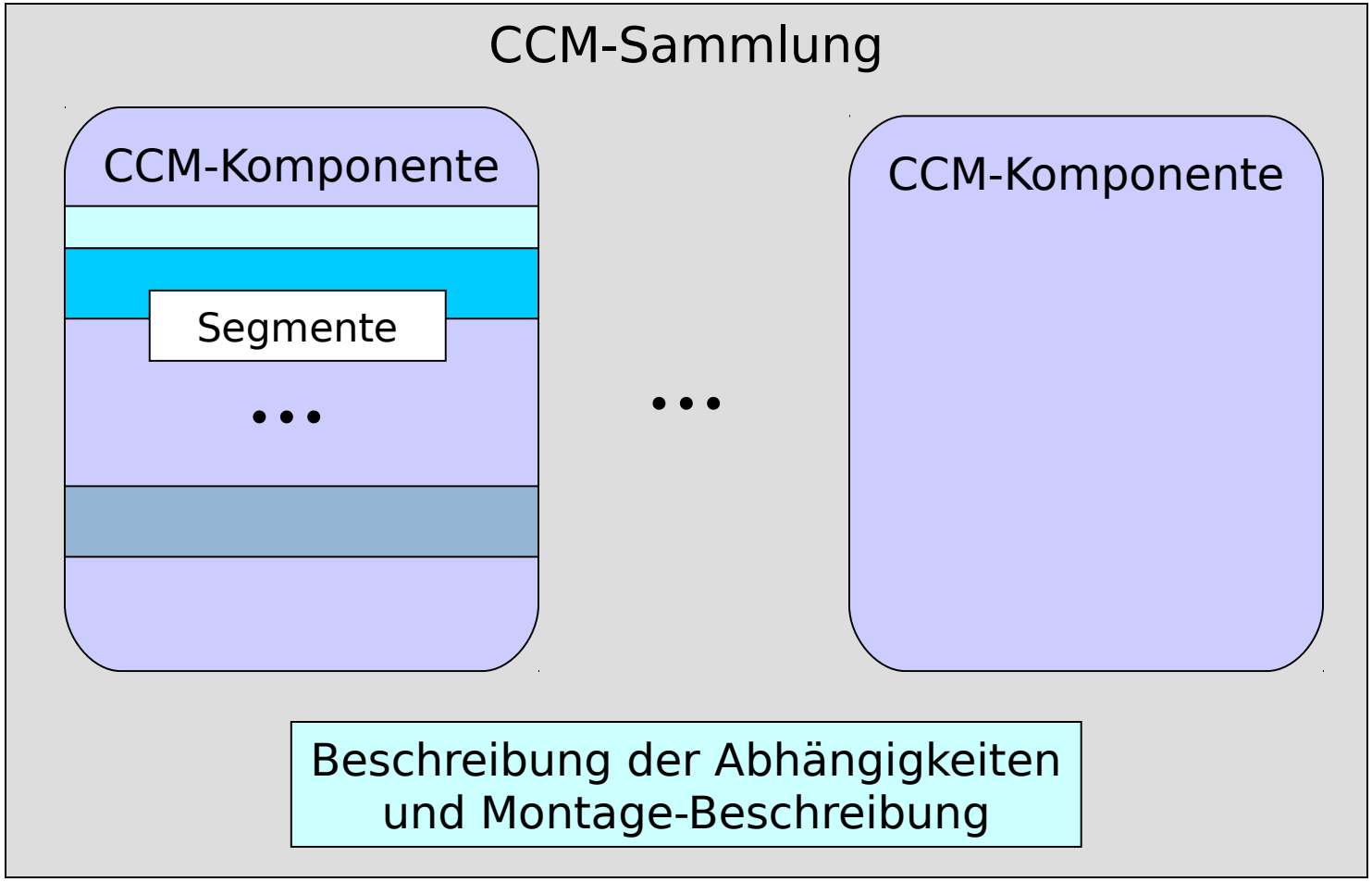
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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

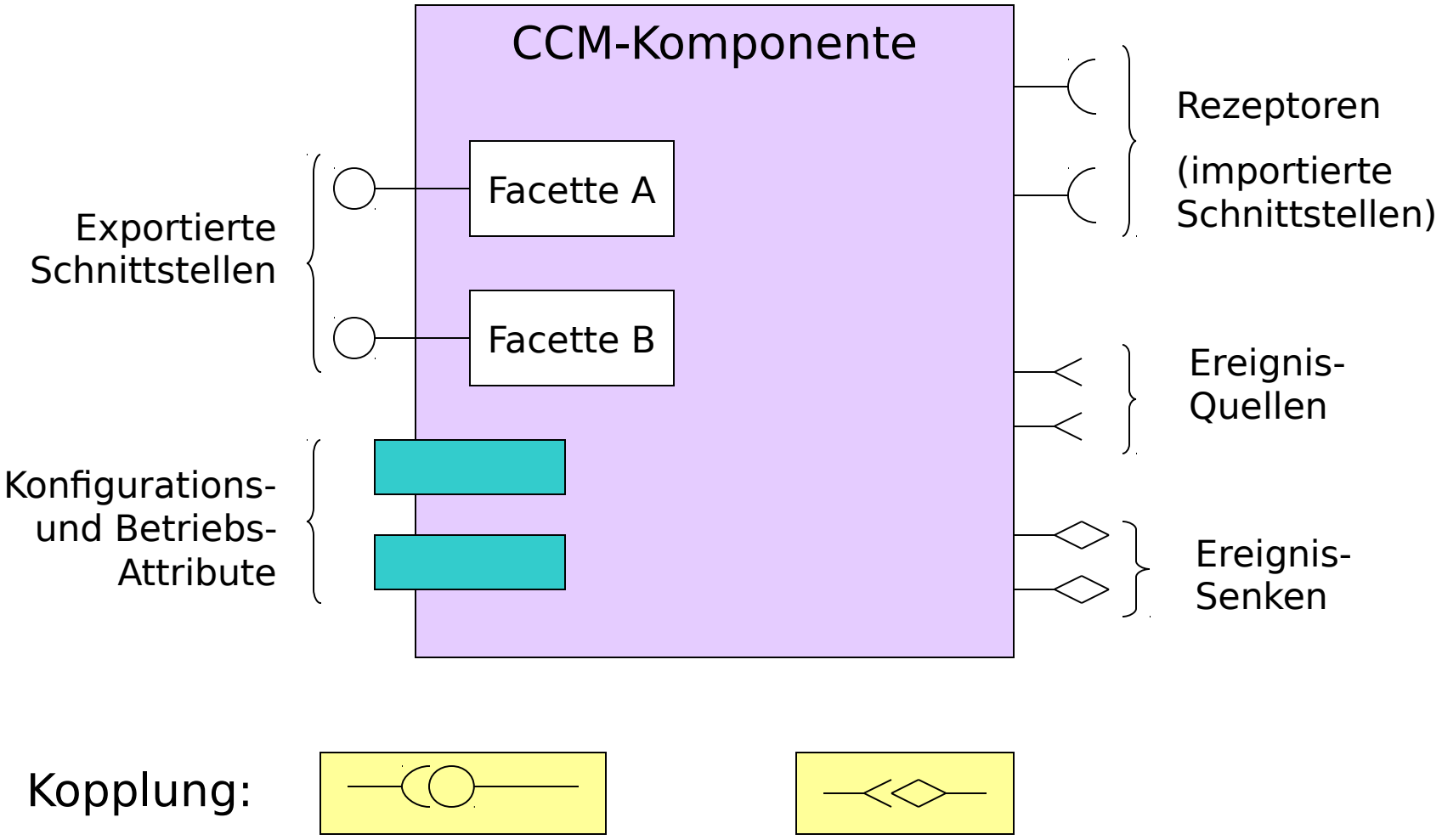
CCM-Kategorien

- 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

Aufbau einer CCM-Komponente



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

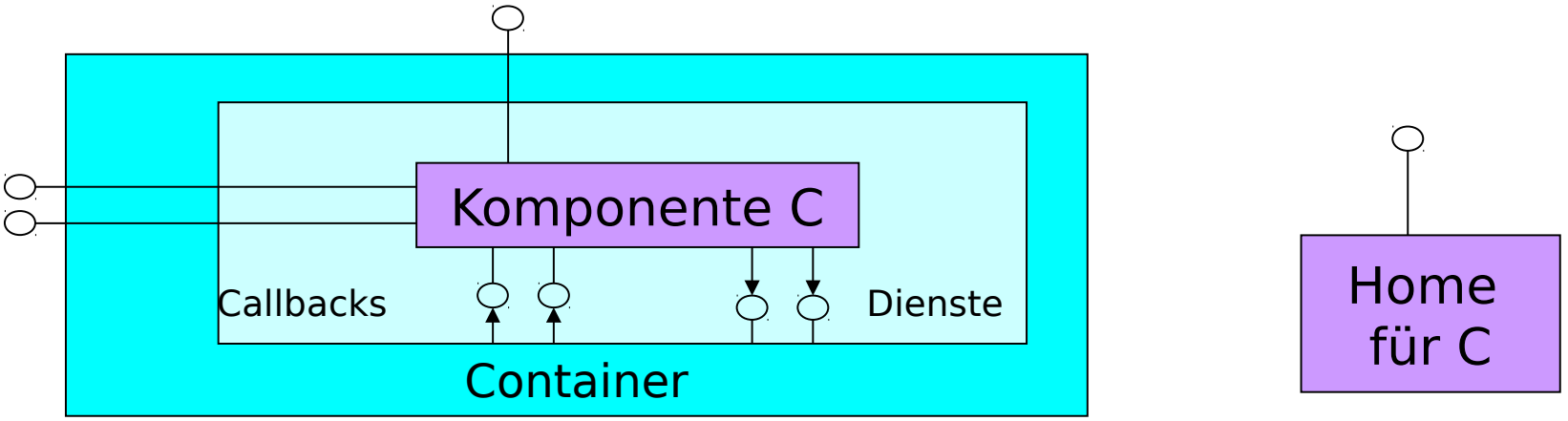
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- **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

Vorlesung
Software aus Komponenten

5. Komponentenkonzepte im Vergleich

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2013/14

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponentenkonzepte und Anforderungen im Vergleich

Eine Zusammenfassung

- Komponententechnologie und Softwaretechnik
- Komponentenkonzepte im Vergleich
- Konvergenz der Konzepte
- Differenzen der Konzepte
- Komponenten und Objekte
- Kontraktspezifikationen für Komponenten
- Komponenten und Softwaretechnik
- Komponenten-Montage
- Komponenten und Berufsprofile

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Zwei grundlegende Herangehensweisen

Eng gekoppelte Architektur (CORBA, Java, EJB, .NET, Spring)

- Laufzeitsystem als Infrastruktur, in der Informationen über Objektinstanzen ausgetauscht werden, in denen Zustand und Funktionalität des Gesamtsystems lokal gespeichert sind.
- fein granulares Konzept, Technik der Interaktion steht im Fokus
- Erweiterung objektorientierter Ansätze von einer Einzelplatzanwendung auf eine verteilte Umgebung
- grundlegendes Konzept: RPC und dessen Verallgemeinerungen

Lose gekoppelte Architektur (Webservices)

- hohe Autonomie der Rechner, die nachrichtengesteuert gegenseitige „Dienste“ erbringen
- grobgranulares Konzept auf höherer Abstraktionsstufe
- näher am Geschäftsprozess-Modell
- in dieser Vorlesung nicht besprochen

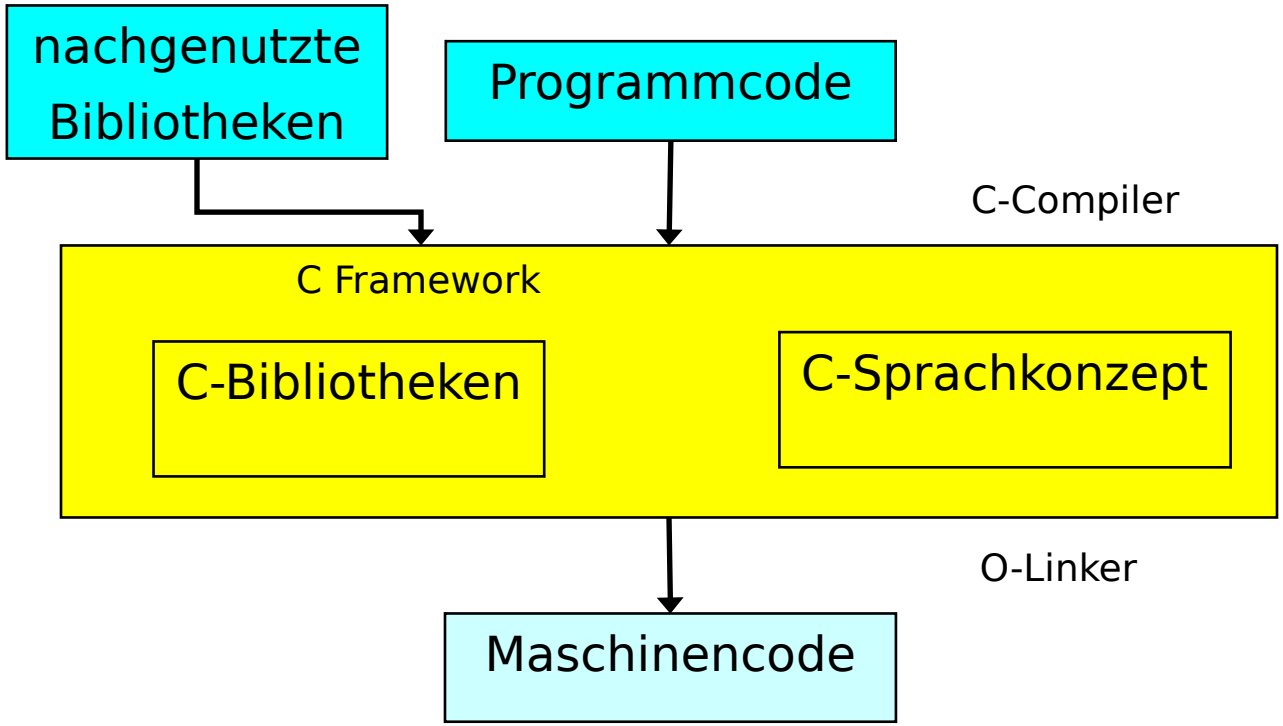
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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

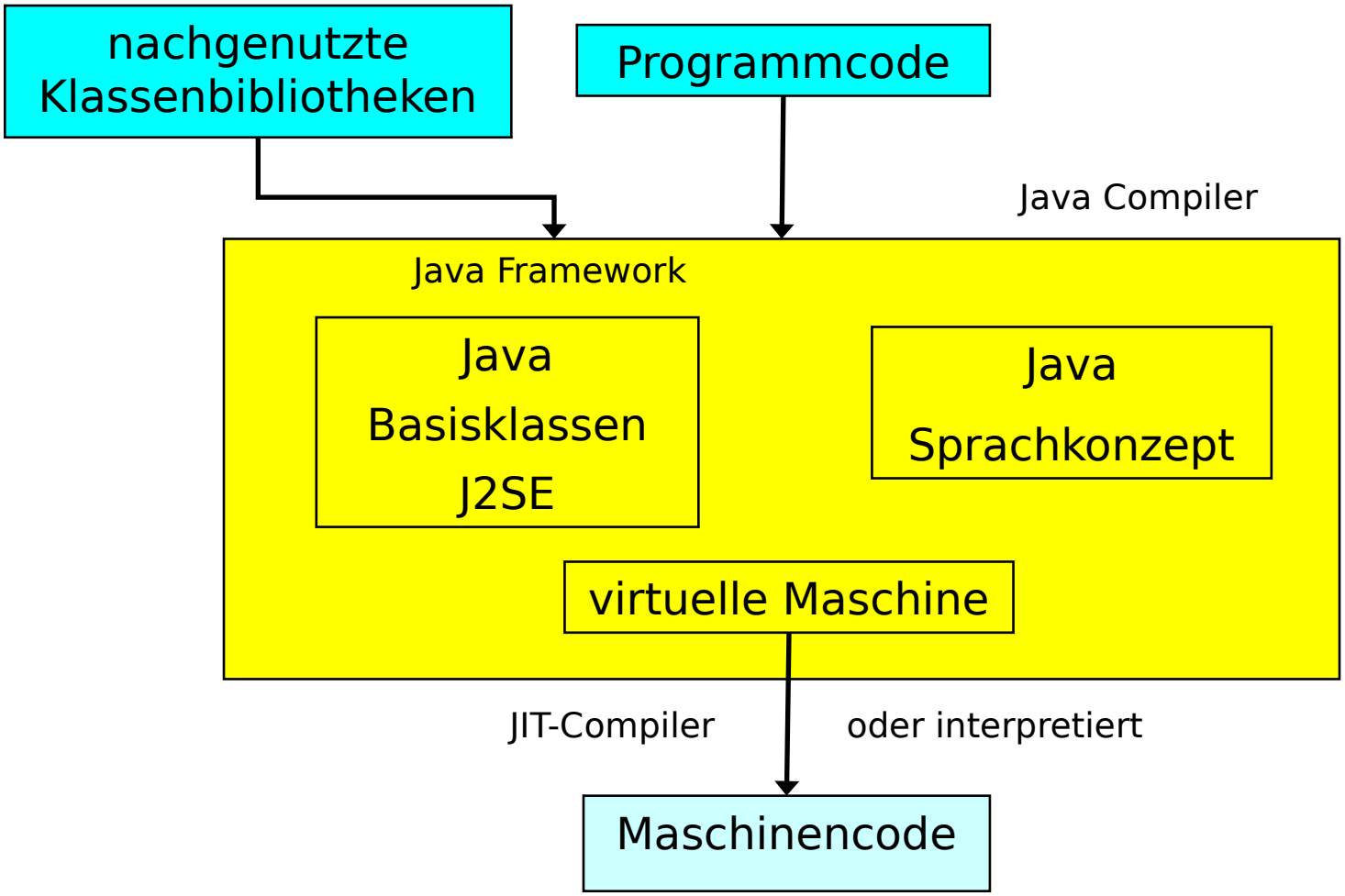
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Eine Sprache, eine Plattform: C



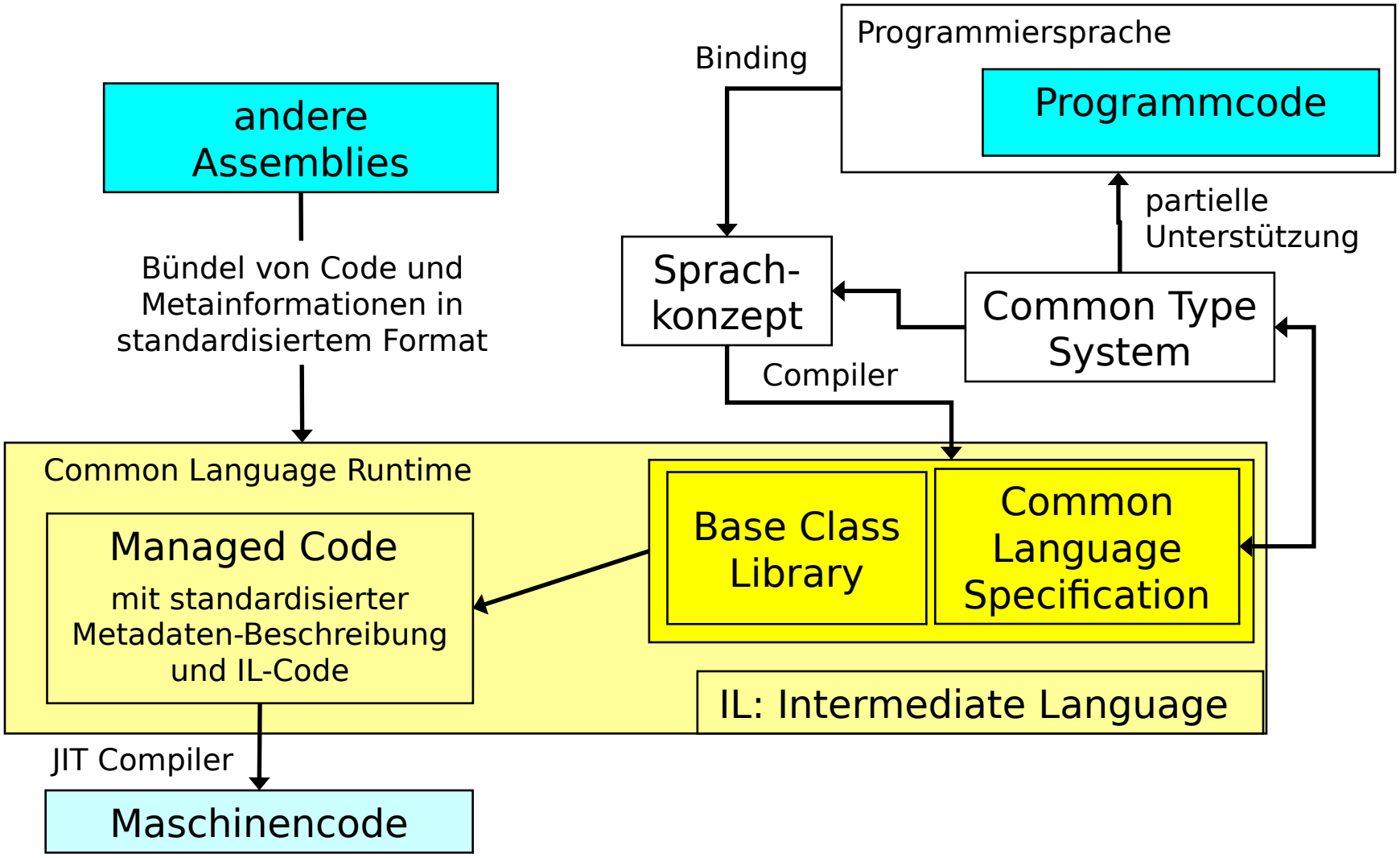
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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

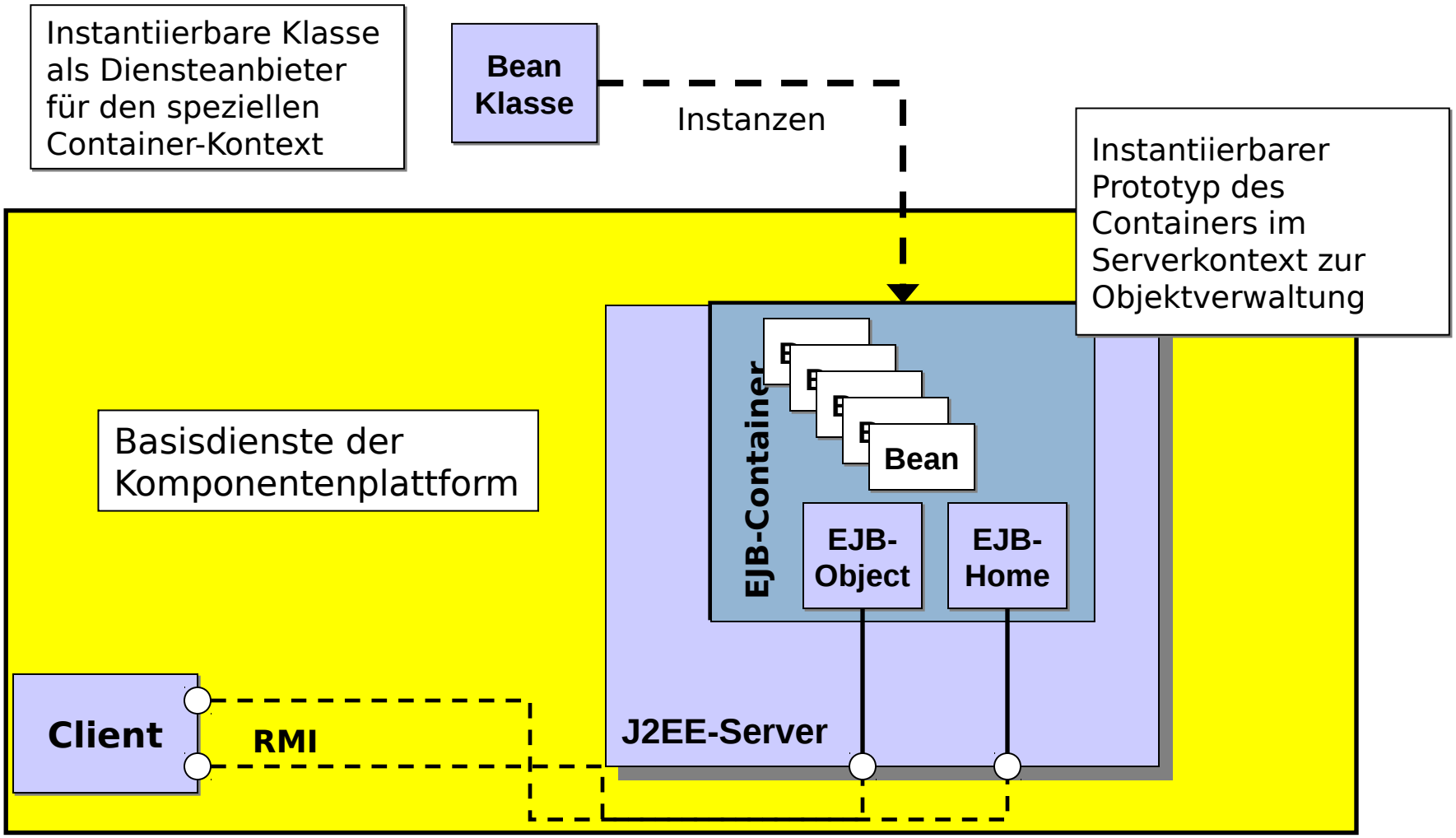
Anwendungsbereich: Middleware und verteilte Systeme

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

Beispiele: CORBA, EJB, Spring, Corba Component Model

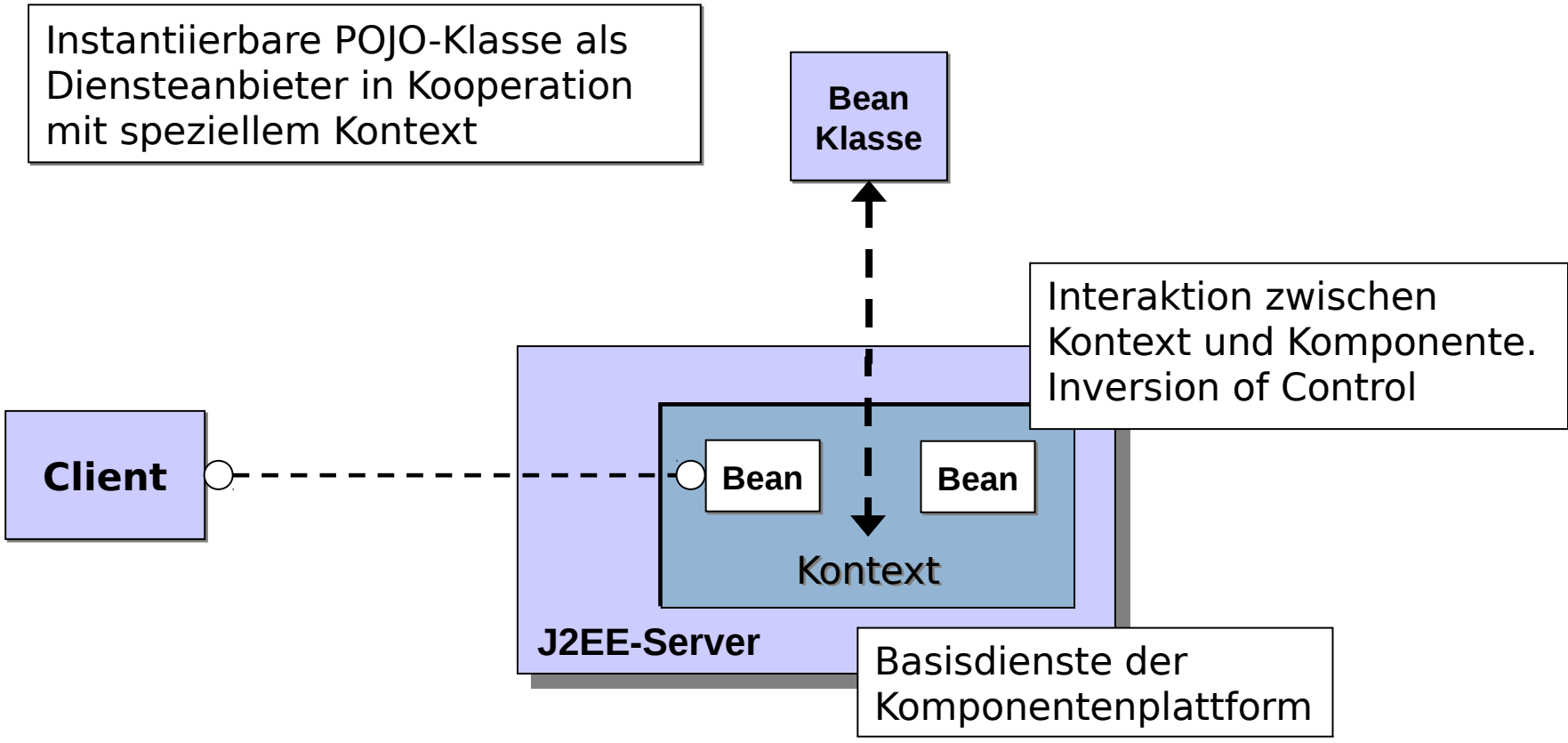
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Prototypischer Aufbau von CORBA und EJB 2



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Prototypischer Aufbau leichtgewichtiger Komponenten-Frameworks



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Konvergenz auf der Ebene der Konzepte

- Alle Zugänge unterstützen spätes Binden, Kapselung, dynamische Polymorphie, Vererbung auf Schnittstellenebene
- Standardisierte Komponenten-Transfer-Formats unterschiedlicher Leistungsfähigkeit
  - Java: *.jar, CLI: Assemblies
- Uniformer Datentransfer
  - einheitliche Konzepte der Serialisierung von Objekten
  - Entwicklung von Persistenzmechanismen auf dieser Basis
- Ereignis- und Ereigniskanal-Konzept
- Metainformationen können über Introspektion und Reflektion zur Laufzeit ausgelesen werden
- Einsatz von Konfigurationsinformationen
  - Montage-Beschreibungen
  - attributbasiertes Programmieren (CLI) – custom attributes
  - Annotationen, Übergang zu Beschreibungssprache für Abhängigkeiten

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Verteilte Speicherverwaltung und Garbage Collection

- Komplizierte Aufgabe in Systemen mit verteilten Objekten
- explizites Management des Lebenszyklus: CORBA
- Referenzzähler-Konzept: COM/DCOM
  - verlangt Kooperation aller Komponenten
  - skaliert schlecht in offenen verteilten Umgebungen
- Object leasing = Objektreferenzen haben nur beschränkte Lebensdauer
  - Java: GC von Java-RMI mit sehr guter Performance in verteilter Umgebung.
  - CLR: verwendet ähnlichen Ansatz