Vorlesung
Software aus Komponenten

3. Komponentenmodelle

apl. Prof. Dr. Hans-Gert Gräbe
Wintersemester 2009/10

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komponenten als Packungseinheiten - Assemblies

- atomare, selbstbeschreibende Einheit, inklusive Metadaten
  - Portable Executable (PE-Datei)
- Metadatenstruktur wird **Manifest** genannt
- "single file assembly" - Manifest ist Teil des eigentlichen Codes - oder
- "multi file assembly" - Manifest ist eigenständige Einheit
- ein Manifest enthält ...
  - Identität der Assembly (Name, Version, ...)
  - die Namen aller Dateien in der Assembly
  - kodierte Hashwerte aller Dateien im Assembly
  - Details der vorhandenen Klassen, Methoden und Eigenschaften
  - Namen und Hashwerte aller referenzierten Assemblies
  - Sicherheitseinstellungen

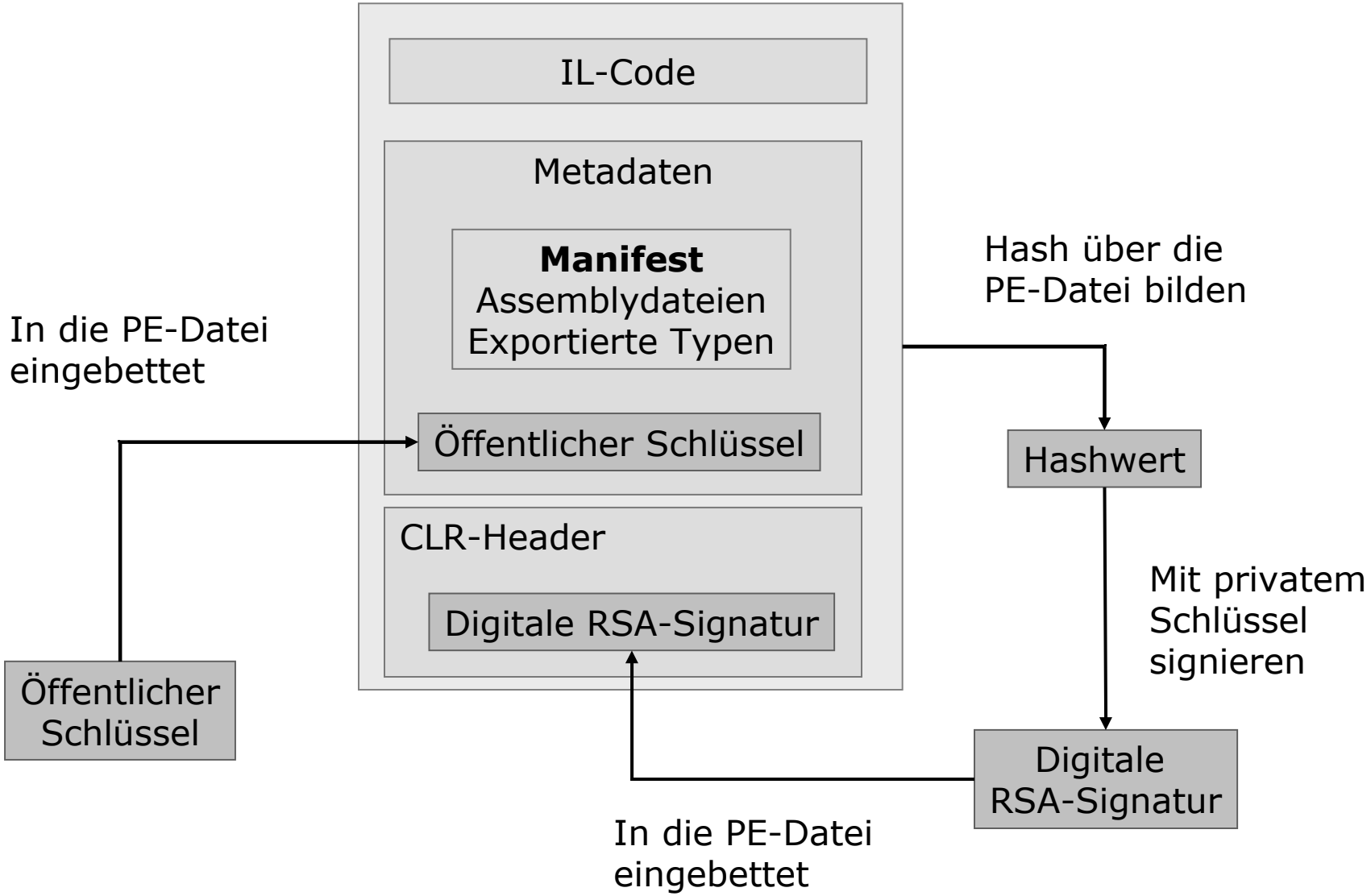
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- starke Namen (strong names) zur eindeutigen Identifizierung einer Assembly
  - zusammengesetzt aus Dateinamen (ohne Erweiterung), Versionsnummer, Kultur, Token für einen öffentlichen Schlüssel
  - festdefinierte Struktur:

    Name_Versionsnummer_Kultur_Schlüsseltoken
- Signierung = Möglichkeit zur Sicherstellung der Unversehrtheit des Codes
- Assemblies können *privat*, *gemeinsam* (*shared*) oder *global* sein.
- globale Assemblies werden auf dem Rechner mit dem Werkzeug *gacutil* im globalen Assembly-Zwischenspeicher (Global Assembly Cache (GAC)) gespeichert.
  - Vorteil: Assembly kann global genutzt werden
  - Nachteil: Keine einfache Installation mehr möglich

Assemblies

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Weitergabe von Assemblies

- Kopieren von Dateien in Zielverzeichnis, z.B. per Batch
  - alle abhängigen Verweise und Typen sind in Assembly enthalten
  - referenzierte Assemblies im Anwendungsverzeichnis
- Konventionelle Installation möglich (cab, msi ...)
  - Verknüpfungen auf Desktop, Startleiste ...
  - Einbindung in Softwareverwaltung von Windows
  - zukünftig eventuell Automation von Verknüpfung
- Private Assemblies
  - werden in Anwendungsverzeichnis installiert
  - werden **nur** von jeweiliger Anwendung benutzt
  - es werden nur Typen gebunden, die für die Anwendung passen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Weitere Features

- Verbindung von managed und unmamaged code über Interop-Technik
  - Einbindung traditioneller COM-Programme über .NET-Kapsel möglich
  - unmanaged code oft an performanzkritischen Stellen
  - Bedeutung relativiert durch die Erfahrung, dass es oft effizienter ist, die Algorithmen zu verbessern statt die Implementierung
- Komplexes Sicherheitskonzept
  - Authentizität des Herstellers
  - Schutz der Programme vor Veränderung
  - codebasiertes und nutzerbasiertes Sicherheitsmodell zur Beschreibung der Vertrauenswürdigkeit des Codes

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Weitere Features

- Attributbasiertes Programmieren
  - Dependency Injection zur Übernahme von Diensten der Umgebung (Assembly) in das Programm
  - wurde von Java als Annotationen z.B. in EJB 3.0 übernommen
- Basisklassen-Bibliothek deckt alle wichtigen Grundfunktionalitäten ab
  - z.B. Textformatierung, Email-Versand, Codegenerierung
  - Basisklassen zur Anbindung an den Web Service Standard als Grundlage für verteilte Anwendungen
- Integration der Laufzeitumgebung in Windows Server seit 2003
  - explizit in Konkurrenz zu J2EE
  - Kernbestandteil von Windows Vista
  - abgespeckte Version der Laufzeitumgebung für Kompaktgeräte

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Weitere Features

- Bindung verschiedener Programmiersprachen an dieselbe Plattform erlaubt Erstellung einer Applikationen unter Verwendung verschiedener Hochsprachen
  - wird durch sprachunabhängige Editoren wie Eclipse oder Visual Studio .NET
  - Gemeinsame Verwendung von Bibliotheken
  - Wartbarkeit derartiger Projekte ist deutlich komplexer als solcher, die nur in einer Hochsprache geschrieben sind.
  - Ändert sich mglw. mit dem Einsatz generativer Techniken und modellgetriebener Software-Entwicklungsansätze
- Hoher Anklang der Plattform vor allem in marktengen und akademischen Bereichen, wo Programmiermöglichkeiten voll ausgereizt werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bestandteile des Frameworks

- Common Intermediate Language / Common Language Runtime
  - Zwischensprachencode wird in Maschinencode übersetzt und ausgeführt
  - kleinste Schnittmenge für Sprachen, die von CLR unterstützt werden
- Base Class Library
  - Bibliotheken mit wichtigen Basisfunktionen werden erst auf der Ebene der CIL eingebunden und sind so hochsprachenunabhängig verfügbar
- Common Language Specification / Common Type System
  - Vereinigung der Sprachkonzepte, die im .NET-Framework unterstützt werden, um Datenkompatibilität auf Hochsprachenebene zu sichern
- Assemblies
  - Komponentenkonzept zum Kapseln von Software-Bausteinen
  - mit Metadaten, Integritätscheck, Versionierung
  - Konzept des Managements von Abhängigkeiten zu anderen Assemblies
  - Global einheitliches Namenssystem (GAC)
  - werkzeuggestütztes Verwaltungssystem für Assemblies
  - Administration über Konfigurationsdateien (XML)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Allgemeine Konzepte

- Genauere Trennung und Spezifikation von
  - Entwicklungsmodell
  - Interaktionsmodell
  - Plattformmodell
  - Packaging and Deployment
- Fokus auf serviceorientierte Architekturen
  - stärkere Trennung von **Geschäftslogik** innerhalb der Komponenten und **Interaktionslogik** im Framework
- Fokus auf eng gekoppeltes Zusammenspiel der Interaktionslogik auf einer gemeinsamen BS-Basis (Applikationsserver, Client-PC)
  - Spring-Framework
  - OSGi – Open Service Gateway Initiative
  - CCM – Corba Component Mode
- Im Vergleich zu J2EE leicht-gewichtigere Container und Entwicklungsstrukturen (Lean Software Movement)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Separation of Concerns

- **Concerns** = Spezifische Anforderung oder Gesichtspunkte, die in einem Software-System behandelt werden müssen, um die übergreifenden Systemziele zu erreichen (Laddad, 2003).
- Problem: Mehrdimensionalität dieser Anforderungen lässt sich in modularem Entwurfsparadigma nur bedingt abbilden
- Trennung bereits auf der Ebene der Anforderungserhebung in
  - **Fachanforderungen** (*core concerns*), die als Komponenten ausgebildet und früh in die finale Applikation integriert werden
  - **Querschnittsanforderungen** (Logging, Sicherheit, Transaktionskonzepte, *cross cutting concerns*), deren Integration möglichst lange hinausgezögert wird
- In modernen Komponentenmodellen werden Querschnittsanforde-rungen der Fachapplikation als Basisdienste aus der Plattform über einen klar ausgeprägten **Kontext** (Laufzeitumgebung) injiziert.
  - Ansätze: Dependency Injection, Inversion of Control
  - Verallgemeinerung: Aspektorientierte Programmierung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Aspekte

Dependency Injection

- Klassisch: Objekt ist selbst zuständig, seine Abhängigkeiten (Zuordnung von benötigten Objekten und Ressourcen) zu erzeugen und zu verwalten.
- EJB 2: Steuerung liegt ausschließlich beim Framework, Anforderungen werden über Deployment-Deskriptoren spezifiziert
- Neu: Objekte werden vom Framework (im Kontext) erzeugt und verwaltet und von dort auf der Ebene der Anwendung (POJO) verfügbar gemacht – Verallgemeinerung des Factory-Pattern
  - Dezentralisierung durch Annotationen – Verschiebung von der Werkzeug- auf die Sprachebene

Inversion of Control

- Anwendung (POJO) verhält sich gegenüber dem Kontext wie ein Client gegenüber dem Server

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Aspekte

Aspektorientierte Programmierung

- Kontext wird nicht nur als Datenhintergrund und Lebenszyklus-Verwaltung verstanden, sondern als Laufzeitumgebung, in der die Fachlogik des Anwendungscodes abläuft
- Definition von Joinpoints im Anwendungscode, an denen Aspekte eingeflochten werden können
  - Joinpoints (Compilezeit) und Joinpoint shadows (Laufzeit)
- Definition von Regeln, nach denen an solchen Joinpoints wirliche Pointcuts ansetzen
- Advices – Art der möglichen Eingriffe: before, after, around
- Vorteil der Aspektorientierung ist die logische und physische Trennung der Semantik von Querschnittsanforderungen in den Komponenten von deren fachlicher Realisierung (Aspekt)
- Entwicklung eines Sprachstandards für derartige Funktionalität.
- Steckt noch in den Kinderschuhen – heute meist nur in Form expliziter Sprachkonstrukte wie in EJB 3 verfügbar.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Das Spring Framework

- Leichtgewichtiges Open Source Applikationsframework für die Java-Plattform, um Entwicklungen innerhalb der J2EE zu vereinfachen
  - Alternative zu den Sun J2EE Blueprints als Architekturempfehlung für J2EE-Anwendungen, dem ein strenges Schichtenkonzept (Web-Schicht, Business-Schicht auf EJB-Basis, Daten-Schicht) zu Grunde liegt.
- Verwirklicht beispielhaft das Zusammenspiel zwischen POJO's auf der Anwendungsebene und verschieden konditionierbaren Framework-Kontexten (Entwicklung, Testumgebung, Produktivumgebung) durch weitere Einbettung der Kontexte
- Auf der Basis existieren weitere Projekte, welche die Anbindung von Spring an verschiedene häufig anzutreffende Entwicklungsumge-bungen und -anforderungen realisieren.

Geschichte

- Oktober 2002: Grundlagen werden von Rod Johnson im Rahmen seiner Buchreihe „Expert One-On-One J2EE Design and Development" entwickelt, der auch den Sourcecode des Framework-Gerüsts als freien Download zur Verfügung stellt.
- Juni 2003: erstes Release, unter der Apache 2.0 Lizenz
- März 2004: Spring 1.0 – seitdem fand das Framework sehr schnelle Verbreitung. Entwicklung wird von SpringSource koordiniert
- Oktober 2006: Version 2.0
- November 2007: Version 2.5
- September 2009: SpringSource wird von VMWare übernommen
- Dezember 2009: Version 3.0
  - Neue Spring Expression Language, mit der Ausdrücke zur Laufzeit ausgewertet werden können.
  - Konfiguration mit Annotationen
  - REST-Unterstützung (Representational State Transfer) im MVC-Framework

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)
- 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

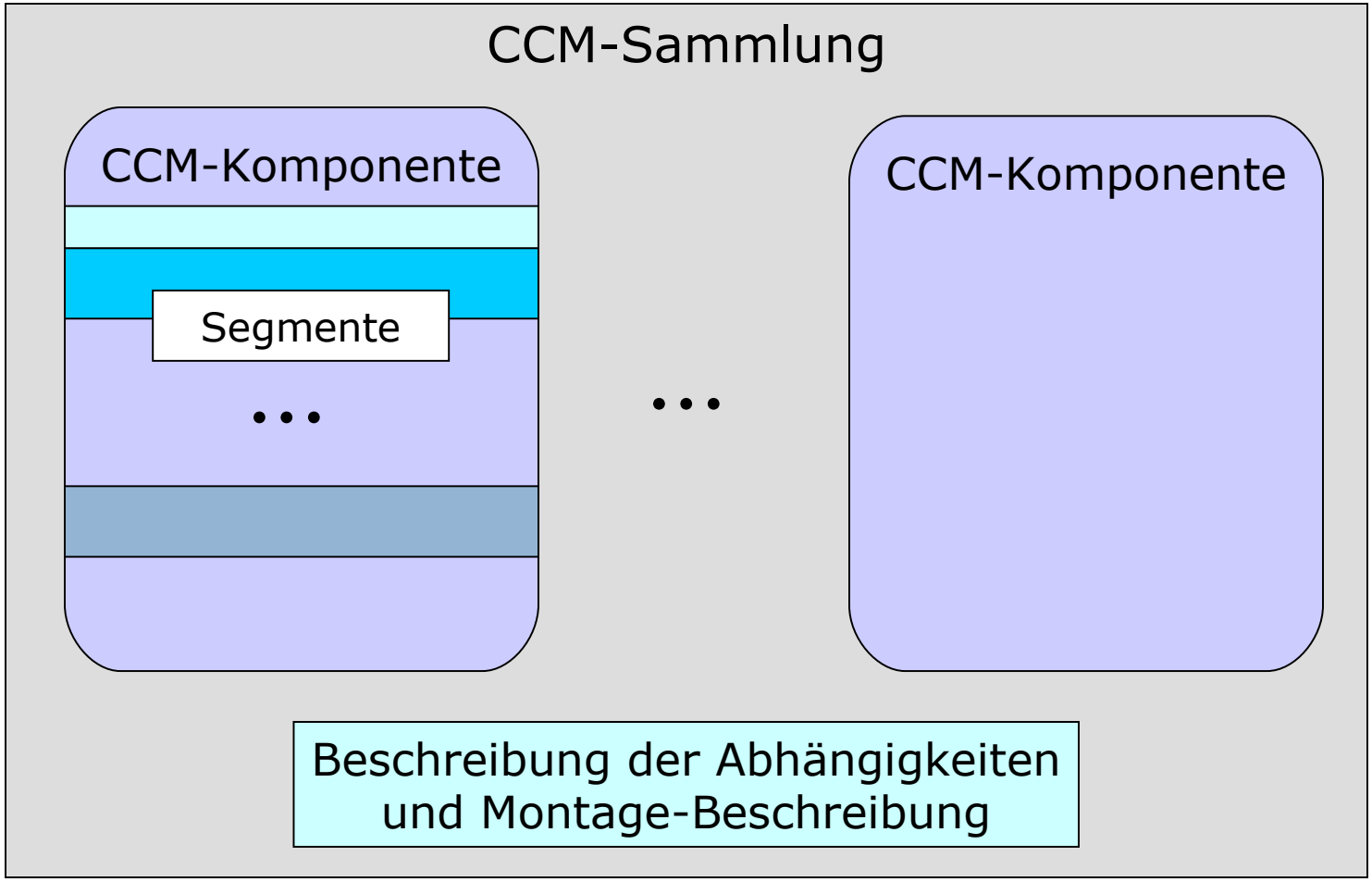
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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