Vorlesung
Software aus Komponenten

3. Komponentenmodelle

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

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

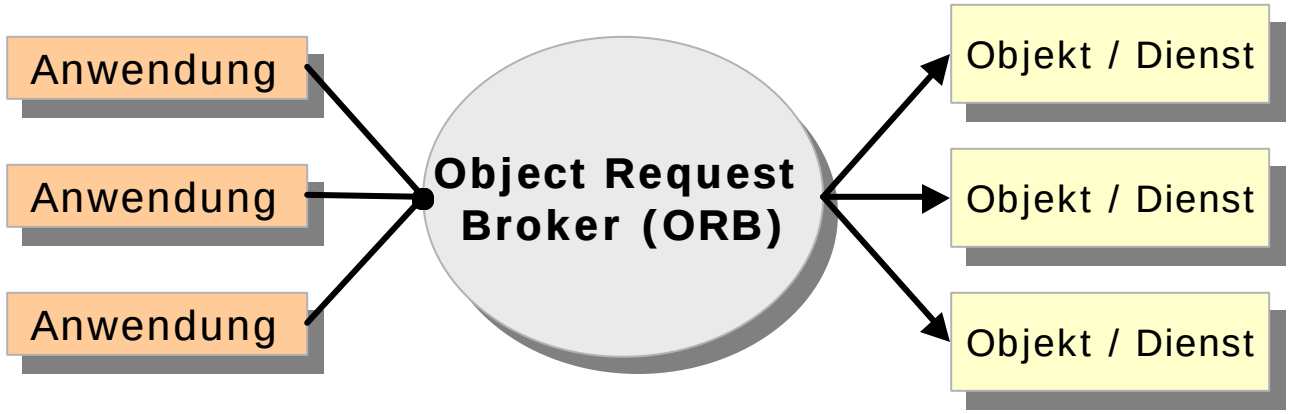
Die OMG und CORBA

- Geschichte, Zielstellungen, Entwicklungsetappen
- Architektur
  - Objekte, Servanten, Anwendungen
  - Schnittstellensprache OMG IDL
  - Dynamische Methodenaufrufe (DII)
  - Symmetrie des CORBA-Modells
- Der Object Request Broker (ORB)
- CORBA-Objekte und Objektreferenzen
- CORBA IDL und Datentypen

- Literatur: CORBA Spezifikation 3.0.3
  - 1154 Seiten pdf-Dokument, siehe http://www.omg.org

CORBA besteht im Grunde aus drei wichtigen Teilen:

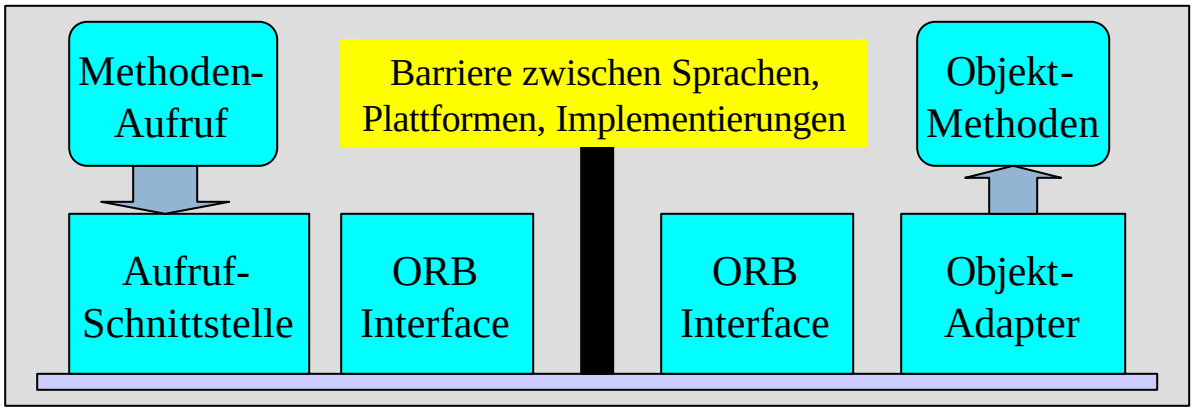
- einer Menge von Aufrufschnittstellen (Invocation Interfaces)
- den Objekt-Anfrage-Vermittlern (Object Request Brokers – ORB's) als den Schaltstellen der Kommunikation
  - mit einem spezifizierten Protokoll, dem internet inter-ORB protocol IIOP
- einer Menge von Objekt-Adaptern



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Laufzeitbindung von Methodenaufrufen

- Aufrufschnittstelle serialisiert Aufrufargumente
- ORB‘s suchen Zielobjekt, -methode, organisieren Transport der Argumente
- Objektadapter deserialisiert Argumente und ruft entsprechende Methode des Zielobjekts auf.



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Wichtige Voraussetzungen

1. Schnittstellen müssen in einer einheitlichen Sprache **definiert** werden (**Interface Definition Language - OMG IDL**)
   - wesentlicher Bestandteil des CORBA-Standards
   - ermöglicht generisches Serialisieren / Deserialisieren

```
module Example {
  struct Date {
            unsigned short Day;
            unsigned short Month;
            unsigned short Year;
    }
interface Ufo {
            readonly attribute unsigned long ID;
            readonly attribute string Name;
            readonly attribute Date FirstContact;
            unsigned long Contacts ();
            void RegisterContact (Date dateOfContact);
    }
}
```

Wichtige Voraussetzungen (Fortsetzung)

2. alle Programmiersprachen, die den CORBA-Standard unterstützen, müssen **an OMG IDL gebunden** werden.
   - Mapping von Datentypen,
   - Übersetzung des OMG IDL Operationsformats in das sprachspezifische Aufruf-Format
   - Fehlerbehandlung
   - existieren Anbindungen für C, C++, SmallTalk, Cobol, Java, ...

In OMG IDL beschriebene Schnittstellen werden dann

- mit einem **OMG IDL Compiler** übersetzt
- im **Schnittstellen-Repository des ORB** abgelegt
- durch **Methoden der ORB-Schnittstelle** angesprochen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

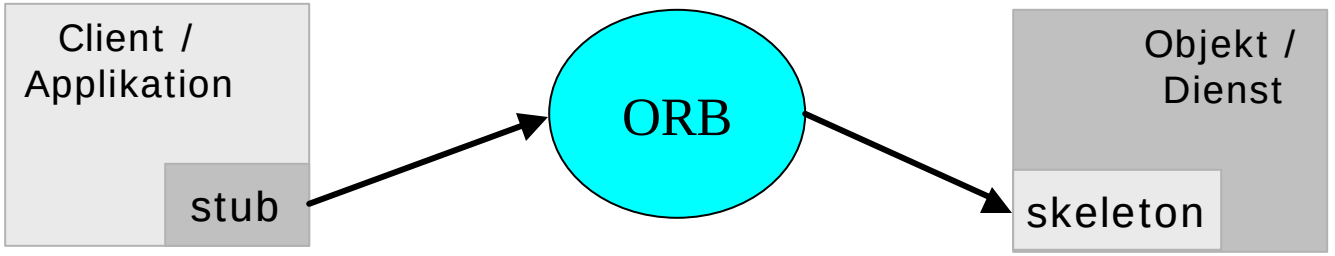
Wichtige Voraussetzungen (Fortsetzung)

3. Programmfragmente stellen **Implementierungen** für solche Schnittstellen (oder Teile davon) bereit
   - heißen **Objekt-Servanten** (object servant)
   - werden im **ORB-Implementations-Repository** registriert
   - Servanten werden vom ORB bei Bedarf geladen und/oder gestartet
   - Objektadapter teilen dem ORB mit, welche Objekte von welchen Servanten bedient werden.
   - Eine Serverumgebung (typ. Prozess) kann mehrere Servanten bedienen.
   - *:* - Beziehung zwischen Objekten und Servanten

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Stummel (stubs) und Skelette (skeletons)

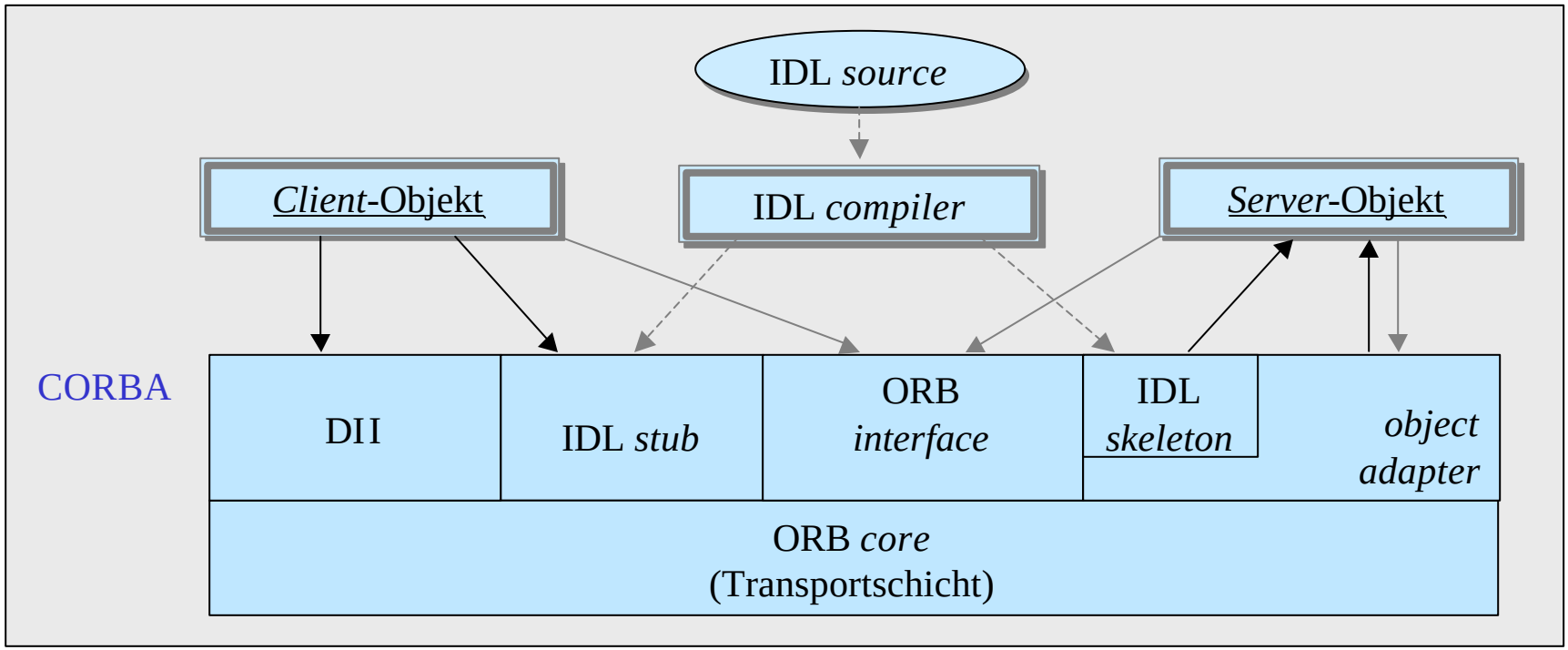
- Methodenaufrufe erfolgen über Stummel-Skelett-Prinzip des RPC
  - mit OMG IDL Compiler aus Schnittstellenbeschreibung generierbar
- **Stummel** sieht lokal wie ein Objekt aus
  - heißen deshalb auch (client-seitiges) Proxy-Objekt
- **Skelett** nimmt Methodenaufruf auf, deserialisiert Argumente und ruft dann die Zielmethode auf
- **Skelett** übernimmt Return-Wert, serialisiert diesen und gibt ihn weiter
  - heißen deshalb auch (serverseitiger) Stummel
- direkt nur für statische Methodenaufrufe einsetzbar (static invocation interface SII / static skeleton interface SSI)



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Dynamische Methodenaufrufe (dynamic invocation interface - DII)

- Erforderlich, um Methoden zur Laufzeit binden zu können
- seit CORBA 2.0 auch Dynamik auf der Serverseite (dynamic skeleton interface – DSI)
- verwenden eine **universelle Datenstruktur für Argumente**, um Methoden mit (statisch) unbestimmter Signatur zu behandeln
- Aus Geschwindigkeits-Gründen wird SII / SSI zusätzlich bereitgestellt.



Symmetrie des CORBA-Modells

- Keine Asymmetrie wie im Client-Server-Modell.
  - Jeder Prozess kann sowohl Methodenaufrufe absetzen als auch empfangen.
- Einzige Asymmetrie kommt durch den **Objektadapter.**
  - Programme, die als Servant eingesetzt werden, müssen sich beim ORB durch den Objektadapter registrieren
  - erster Standard: basic object adapter (BOA), deprecated seit 1998
    - war unterspezifiziert, deshalb Wildwuchs von Erweiterungen
  - heute: portable object adapter (POA)
    - aktueller Standard ist Teil von CORBA 3.0.3, März 2004
  - Mit der Registrierung „weiß“ der ORB, wie der Servant aktiviert wird
    - geschieht ebenfalls über den Objektadapter eines Objekts
    - jedes Objekt hat eine „Hausmaschine“, auch wenn ein Servant über mehrere Maschinen „verfügt“
  - Reine Applikationen, die keine Dienste zur Verfügung stellen, sondern nur welche nutzen, werden auch nicht registriert.
    - können damit auch nicht durch den ORB gestartet werden.

Aufgaben des ORB

- Schlüsselkomponente und Kommunikationszentrale der Architektur
- organisiert Methoden-Aufrufe zwischen Applikationen und Servanten
  - arbeitet nur mit den Schnittstellen (stub, skeleton)
  - Schnittstellen-Definition über OMG IDL
- verwaltet **Schnittstellen-Repository** (interface repository - IR)
- **Dienste** werden über **Objekte** angesprochen, deren Methoden mit den entsprechenden **Servanten** verbunden sind.
- Stummel – ORB:
  - liefert Interfacedefinitionen aus dem IR
  - dynamische Bindung von Aufrufen (DII)
  - Auflösung von Objektreferenzen
- ORB – Servant:
  - Suche und Aktivieren / Deaktivieren von Objekten, über deren Methoden der jeweilige Dienst erbracht wird
  - ORB verwaltet dazu das **Repository der Implementierungen** (implementation repository)
  - Eine Reihe von Diensten sind als **Basisdienste** standardisiert

Aufgaben des ORB (Fortsetzung)

- Organisation des Zugriffs auf ORB Basis-Dienste
- Verwaltung der Objekte
  - Aktivierung und Deaktivierung
  - Policy-Operationen
  - Verwaltung zugeordneter leichtgewichtiger Prozesse (Threads)
- Verwaltung der Objekt-Referenzen
  - Konvertierungen, Duplizieren, Speicherfreigabe

Der ORB ist ebenfalls ein Objekt.

- Aufbau und Organisation des ORB sind abhängig vom Anbieter und vom Einsatzgebiet
  - einzelner Prozess oder verteilte Anwendung

Interface Repository (IR)

- verwaltet die Schnittstellen-Definitionen
- es sind auch Methodenaufrufe möglich, deren Schnittstelle zur Übersetzungszeit des Clients unbekannt war
- wird vom ORB zur Weiterleitung von Anforderungen benutzt

Implementation Repository

- verwaltet Informationen über die Servanten und die zugeordneten Objekte
- wird vom ORB zum Lokalisieren und Starten von Diensten verwendet
- spezifisch für eine Betriebssystem-Umgebung

CORBA – Objekte

- Objekte sind Programmfragmente mit Eigenleben, charakterisiert durch
  - Objektzustand (aktuelle Werte der Attribute)
  - Funktionalität (verfügbare Methoden)
- Dienste eines Objekts können von Applikationen nur über den ORB in Anspruch genommen werden.
- ORB organisiert Aktivierung und Deaktivierung von Objekten
  - Anforderung entsprechender Ressourcen (CPU, Speicher)
  - Sicherung der persistenten Bestandteile bei Deaktivierung
  - Aktivierungs- und Deaktivierungsmuster können durch entsprechende Regeln (policies) festgelegt sein
- OMG IDL kennt daneben noch (primitive) Datentypen
- jedes CORBA-Objekt hat einen Typnamen (= Klasse in Java)
- Typname entspricht dem Schnittstellennamen in der IDL Deklaration
- Typname steht für einen abstrakten Datentyp als Menge von
  - Methoden und deren Signaturen
  - Variablen (Attributen) und deren Typen

CORBA Objektreferenzen

- Statt Objekten werden normalerweise nur Referenzen übergeben
  - Seit CORBA 2.3 können Objekte auch als Wertparameter übergeben werden
  - werden dazu in ein XML-Konstrukt umgewandelt
  - verwendet eine **standardisierte Serialisierung**
- Referenz über Programmgrenzen ≠ Referenz innerhalb eines Programms
  - kann Objektänderungen durch die Evolution des Programmstatus im Ursprungsprogramm nicht verfolgen
  - Referenzen = Klone, die nach Erzeugung ein Eigenleben entwickeln
  - teurer in der Handhabung als physische Referenzen
  - eher vergleichbar mit einer URL
    - seit CORBA 2.3 existiert Standard zur Darstellung von Objektreferenzen als URL
  - ORB Schnittstelle enthält **Methoden zum Umwandeln** zwischen physischen und CORBA Objektreferenzen
- Lebensdauer per Definition **unbestimmt**
  - Wiederverwendung einer Referenz kann einen Fehler auslösen
  - referenziertes Objekt kann inzwischen gelöscht sein

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

OMG IDL und Datentypen

- OMG IDL unterscheidet primitive Datentypen und CORBA Objektreferenzen
  - Basistypen (integer, float, char, string)
  - zusammengesetzte Datentypen
    - Strukturen, Sequenzen, Aufzählungstypen
    - multi-dimensionale Felder fester Größe
- Parameter eines primitiven Datentyps werden als Wertparameter übergeben
- Umfang der Unterstützung ist von eingesetzter Programmiersprache abhängig
  - CORBA Standard: Aufruf eines nicht unterstützten Typs erzeugt einen Fehler zur Übersetzungszeit

CORBA in der industriellen Anwendung

- Hauptanwendungsfeld: Ersetzen von Sockets und RPC in Anwendungen, die über mehrere Server verteilt sind
- höhere Abstraktionskonzepte vor CORBA 3 kaum bedient
  - Kooperation beim Entwickeln verteilter Anwendungen über Teamgrenzen bisher kaum unterstützt
- Weiter gehende Konzepte hat OMG schon lange im Visier

Object Management Architecture (OMA)

- erste Standardisierungen mit CORBA 2, seit 1997 im Focus der OMG
- mit CORBA 3 ins Zentrum gerückt
- 3 neue Standardisierungsfelder
  - Spezifikation von (grundlegenden) Objektdiensten (CORBAservices)
  - Spezifikation von (häufig benötigten) Anwendungsbestandteilen (CORBAfacilities)
  - Spezifikation von Anwendungsobjekten
- CORBA Komponentenmodell (CCM)

Basisdienste (CORBAservices)

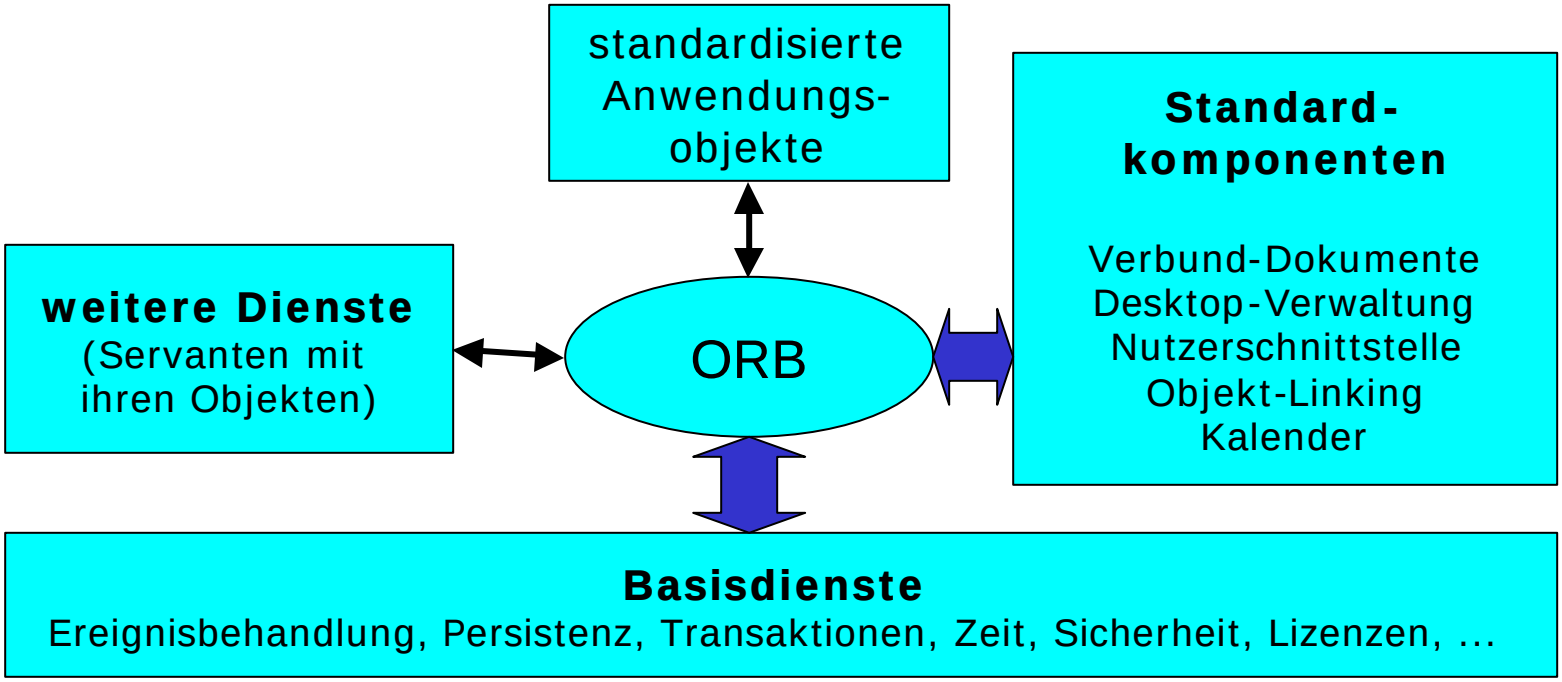
- Unterstützung von CORBA-basierten Programmen unabhängig vom Einsatzfeld oder Anwendungsmodell
- Fokus auf fundamentale Bausteine **jeder** Implementierung
  - z.B. Ereignisbehandlung, Transaktionsabwicklung, Namensverwaltung

Standardkomponenten (CORBAfacilities)

- Standardisierung von häufig benötigten Anwendungsbestandteilen
  - Komponentenrahmen zur einfachen Integration von Anbieterlösungen
- Abgrenzung von Bereichen horizontal oder vertikal
  - horizontal: Fokus auf generellem Anwendungsmodell
    - Standards für Nutzerschnittstellen, System- und Aufgabenverwaltung
    - verliert mit CORBA 3 an Bedeutung
  - vertikal: Focus auf bereichsspezifischen Einsatzfeldern
    - gewinnt mit CORBA 3 an Bedeutung
    - im Rahmen von OMG SIG‘s oder Domain Task Forces

Anwendungsobjekte

- Standardisierung von bereichsspezifischen Einheiten, die in einen **Komponentenrahmen** (component framework) „eingesteckt“ werden können
  - Bsp: Geschäftsfeld-Objekte (business objects) = Objekte, die direkte Geschäftsprozessabstraktionen repräsentieren
  - steht erst ganz am Anfang der Entwicklung



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Auf dem Weg zu CORBA 3

- Als Ganzes formal erst Ende 2002 freigegeben
- Teilstandards Stück für Stück bereits in CORBA 2.3 ... 2.6 integriert
- CORBA 2.3
  - Objekte als Wertparameter
  - XML-Abbildungen
  - Java-RMI als Schnittstellenmodell in CORBA
    - Java-CORBA-Koevolution
    - Java als wichtigste Plattform für Implementierung der Standards
- CORBA 2.4
  - Objektreferenzen als URL
  - asynchrone Botschaften
  - Minimal- und Realzeit-CORBA
- CORBA 2.5: Fehlertoleranz- und Abbruch-Standards
- CORBA 2.6: Sicherheitsstandards
- CORBA 3: **Meilenstein** ist Komponentenmodell (CCM)
  - im Wesentlichen fertig seit Ende 2001