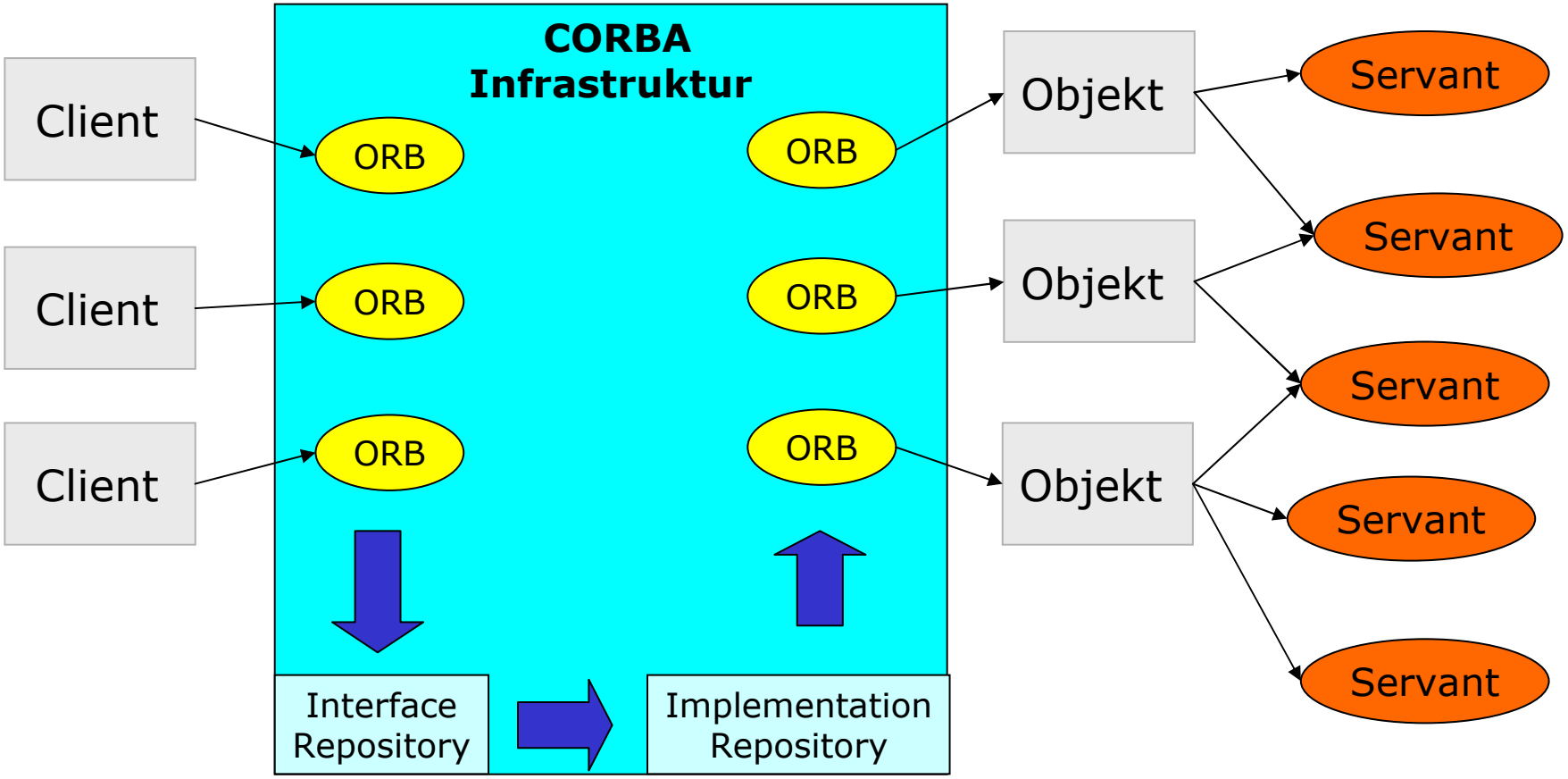
Vorlesung
Software aus Komponenten

3. Komponenten-Modelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2005/06

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

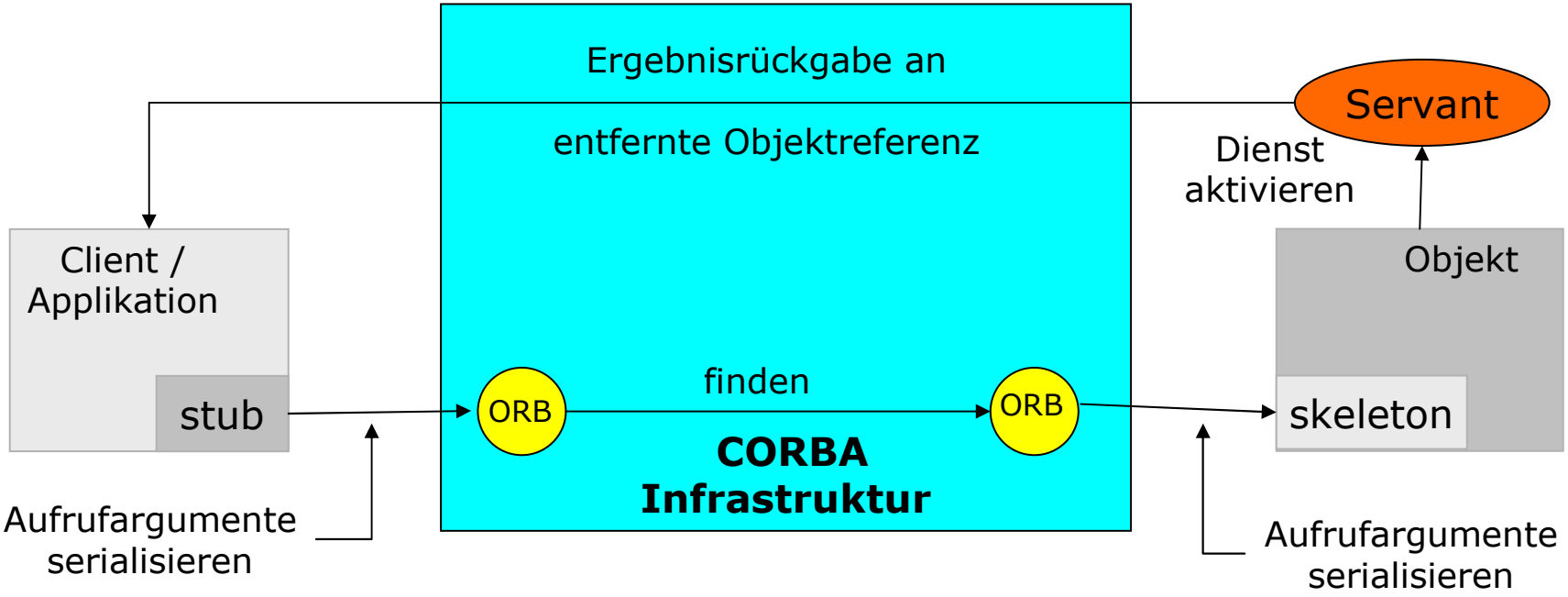
Architektur im Überblick



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Stummel (stubs) und Skelette (skeletons)

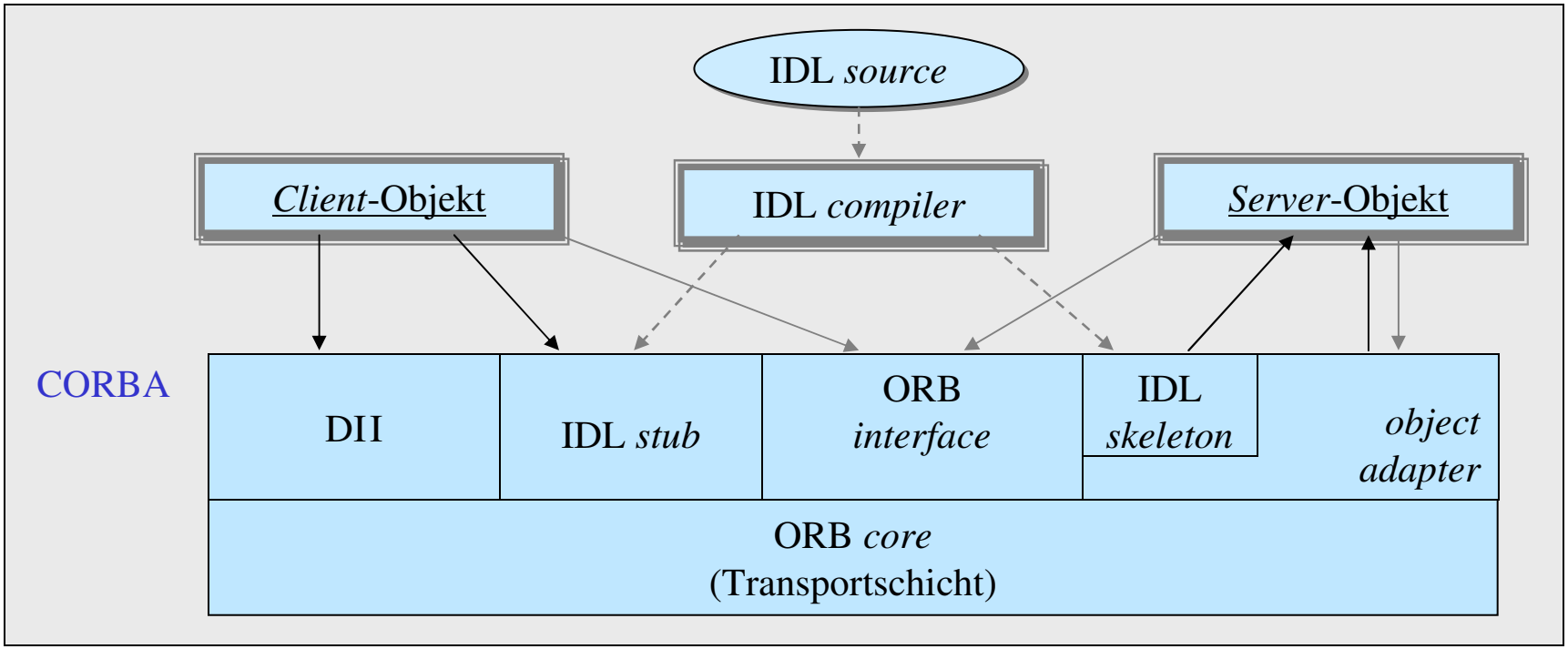
- Methodenaufrufe erfolgen über Stummel-Skelett-Prinzip des RPC
  - mit OMG IDL Compiler aus Schnittstellenbeschreibung generierbar
- direkt nur für statische Methodenaufrufe einsetzbar (static invocation interface SII / static skeleton interface SSI)



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 Objektadapter, wie der Servant aktiviert wird
  - 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
- vermittelt Methoden-Aufrufe zwischen Applikationen und Servanten
  - arbeitet nur mit den Schnittstellen (stub, skeleton)
  - Schnittstellen-Definition über OMG IDL
- Verwendet dazu Informationen aus dem **Schnittstellen-Repository** (interface repository - IR)
- Begriffe: **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 – Objekt - Servant:
  - Suche und Aktivieren / Deaktivieren von Objekten, über deren Methoden der jeweilige Dienst erbracht wird
  - Verwendet dazu Informationen aus dem **Repository der Implementierungen** (implementation repository)

Aufgaben des ORB (Fortsetzung)

- 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

- verwaltet die Schnittstellen-Definitionen
- es sind auch Methodenaufrufe möglich, deren Schnittstelle zur Übersetzungszeit des Clients unbekannt war (dynamische Erweiterbarkeit)

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 und Diensten
  - 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

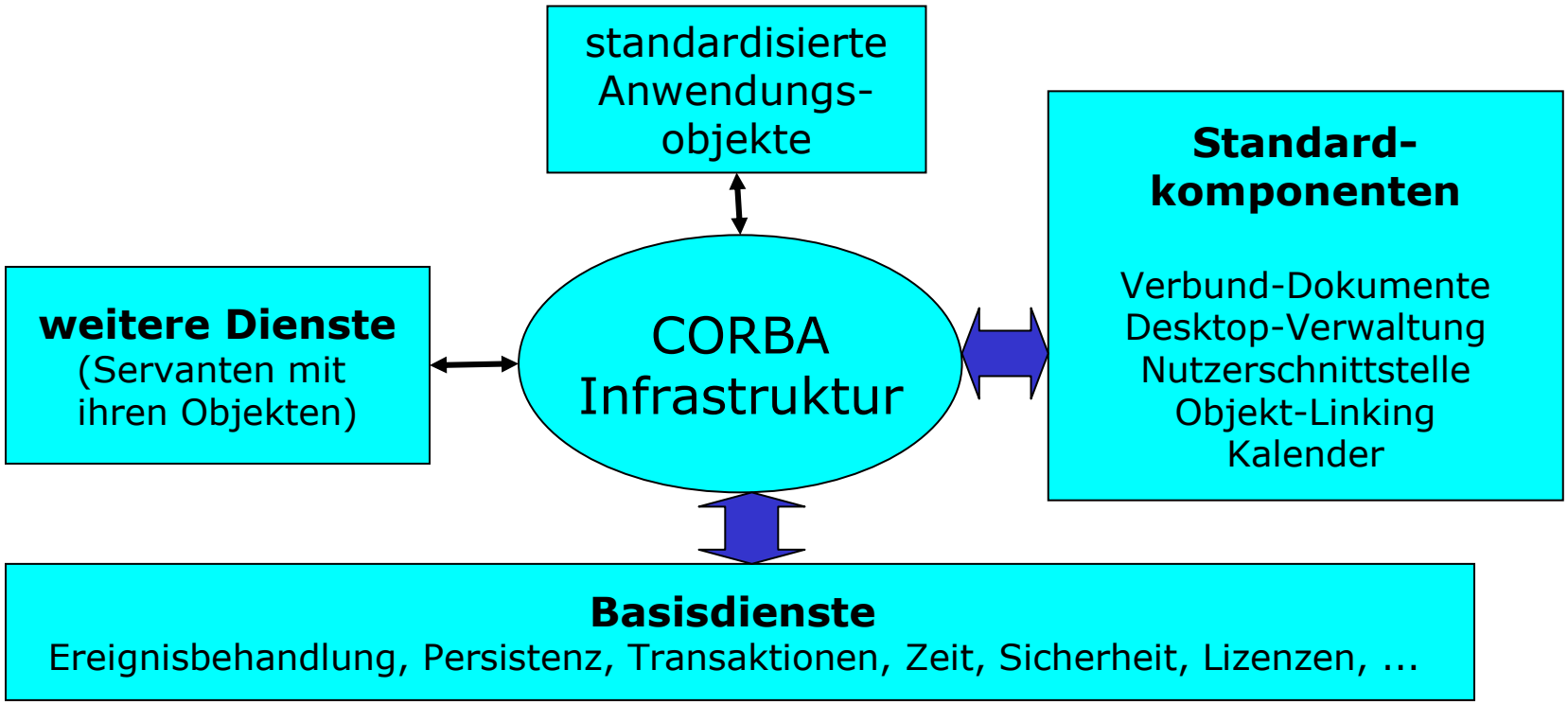
- Hauptanwendungsfeld: Ersetzen von Sockets und RPC in Anwendungen, die über mehrere Server verteilt sind
- höhere Abstraktionskonzepte vor CORBA 3 kaum bedient
  - Kooperation beim Entwickeln verteilter Anwendungen über Teamgrenzen bisher kaum unterstützt
- Weiter gehende Konzepte hat OMG schon lange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

Idee: Unterteilung der Dienste in mehr oder weniger wichtige

- Bereitstellung von Dienst-Servanten für wichtige Funktionalitäten, 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

Basisdienste (CORBAservices)

- Fokus auf fundamentale Bausteine **jeder** Implementierung, welche die Komponenten-Infrastruktur zur Verfügung stellen sollte
  - z.B. Ereignisbehandlung, Transaktionsabwicklung, Namensverwaltung
  - derzeit 16 Basisdienste spezifiziert
- 2 Kategorien:
  - Dienste zur Unterstützung unternehmensweiter verteilter Anwendungen
    - nutzen typischerweise CORBA-Objekte als Moduln und CORBA als Kommunikations-Middleware
    - grob granulare Ebene, CORBA Kommunikations-Infrastruktur als „Objektbus"
  - Dienste zur Unterstützung fein granularer verteilter Anwendungen
    - Bedeutung nimmt ab, da zu hoher Komplexitätsgrad
    - Ausnahme: persistent state service (PSS) als einer der Pfeiler des CCM
- große CORBA-Anwendungen nutzen oft nur wenige Basisdienste
  - verfügbare CORBA-Plattformen bieten deshalb oft nur Implementierungen einiger Basisdienste an

Dienste zur Unterstützung grob granularer verteilter Anwendungen

Namensdienst (name service)

- Abbildung intern verwendeter UUID auf externe Bezeichner
- Namens**-Kontexte** und Kontext-**Hierarchien**
  - vergleichbar zu Verzeichnisstrukturen

Händlerdienst (trader service)

- Verfeinerung des Namensdiensts (white vs. yellow pages)
- Anbieter veröffentlichen Dienstangebote per **Registrierung**
- Nutzer finden Angebote über Händlerdienst per **Beschreibung**
- Händlerdienste organisieren Angebote in Handels**-Kontexten**
- Standardisierte Methoden zur Suche in den Angeboten

Ereignisdienst (event service)

- Verteilung der E.-**Objekte** von E.-**Erzeugern** (event supplier) an E.-**Konsumenten** (event consumer)
- E.-Objekte sind unveränderbar, wenn einmal erzeugt
  - strikt unidirektionaler Informationsfluss
- E.-**Kanäle** (event channel) entkoppeln Erzeuger und Konsument
- E. können getypt sein (OMG IDL)
- Kanäle können Ereignisse nach ihrem Typ filtern
- Push- und Pull-Methoden werden unterstützt

Benachrichtigungsdienst (notification service)

- Erweiterung des Ereignisdiensts um einige kritische Merkmale
  - Dienstqualität, Administration
  - dynamische E.-Filterung, Filterung auf verschiedenen Ebenen
- technisch kein Basisdienst, sondern Standardkomponente
  - gemeinsamer Standard mit Telecomm. Domain Task Force

Transaktionsdienst (object transaction service, OTS)

- einer der wichtigsten Bausteine für verteilte Anwendungen
- standardisiert seit 1994
- wird von den meisten ORB-Produkten und J2EE-Servern unterstützt
- **Eingebettete Transaktionen** nur optional
  - Transaktionshülle um Folge von Operationen
  - erforderlich zur unabhängigen Entwicklung auf verschiedenen Hierarchie-Ebenen
  - (noch) nicht standardisiert, weil kaum eines der heute ex. Transaktionssysteme so etwas vorsieht
- Verwaltung eines (objektspez.) aktuellen Tr.-**kontexts** durch OTS
  - Objekte müssen dazu die Schnittstelle *TransactionalObject* implementieren
  - Methoden *begin*, *commit*, *rollback* operieren auf dem Kontext
- Objekte unter Transaktionskontrolle registrieren sich beim OTS-Koordinator-Objekt