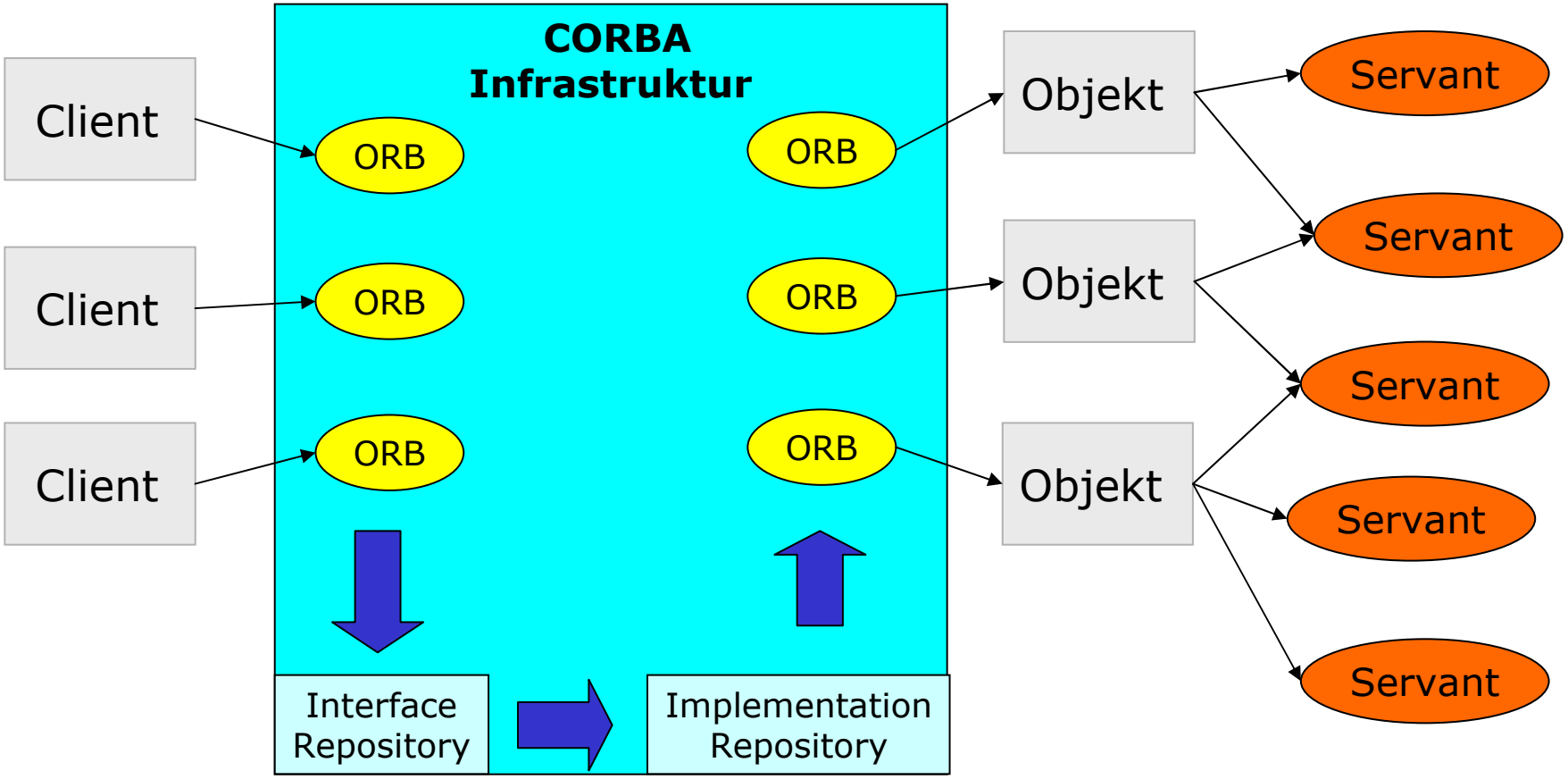
Vorlesung
Software aus Komponenten

3. Komponenten-Modelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2006/07

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

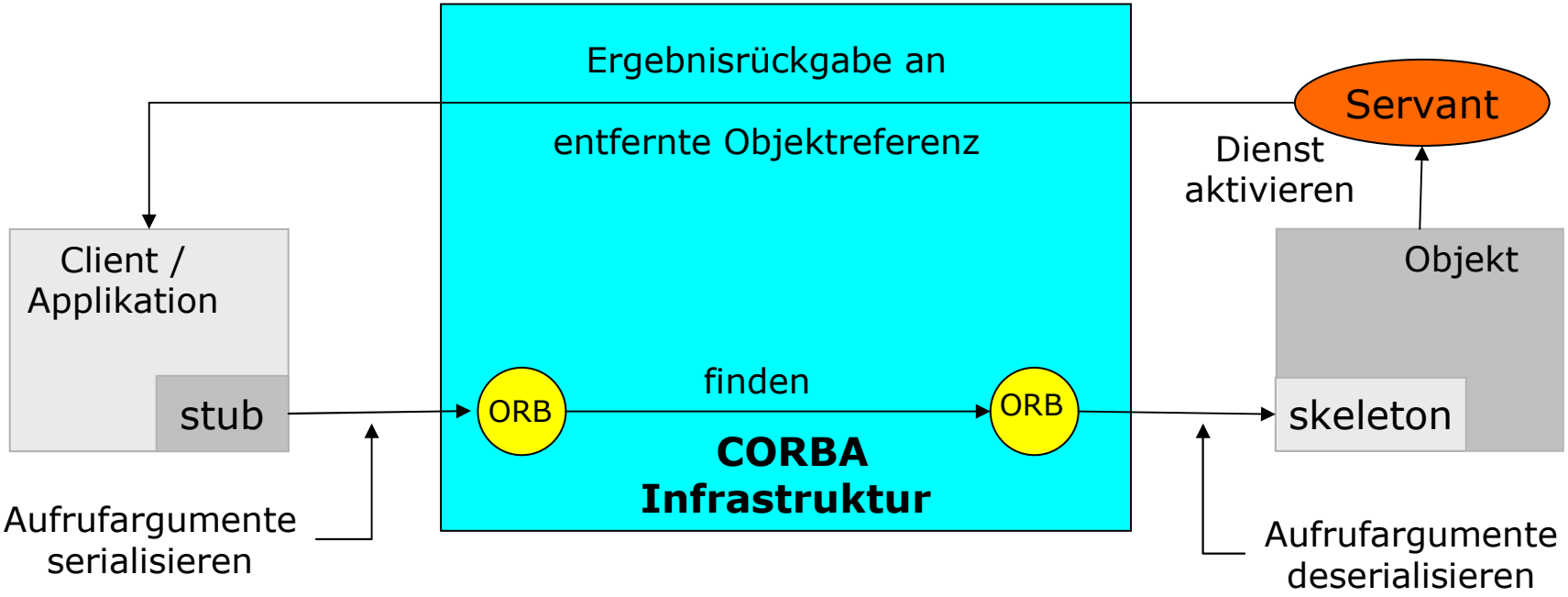
Architektur im Überblick



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Stummel (stubs) und Skelette (skeletons)

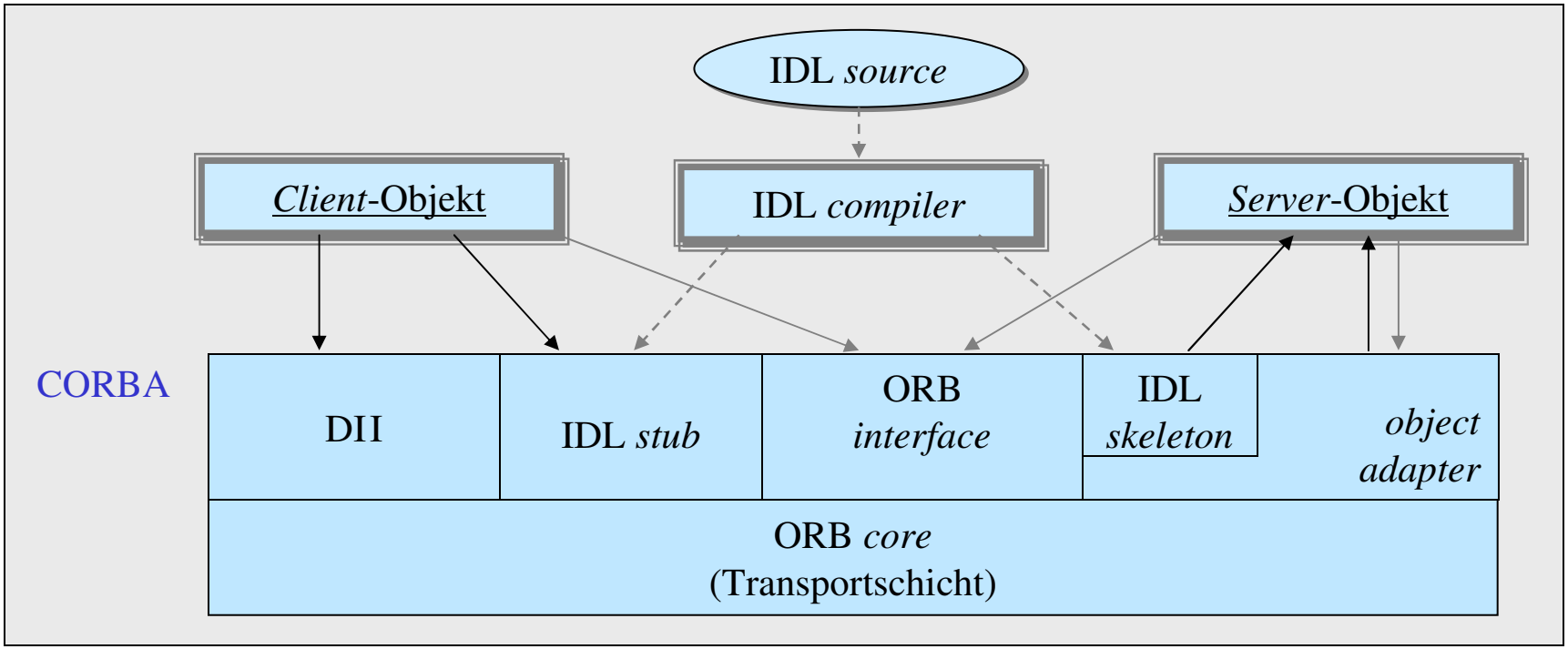
- Methodenaufrufe erfolgen über Stummel-Skelett-Prinzip des RPC
  - mit OMG IDL Compiler aus Schnittstellenbeschreibung generierbar
- direkt nur für statische Methodenaufrufe einsetzbar (static invocation interface SII / static skeleton interface SSI)



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.



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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 einen 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 (und nur dieser), wie der Servant aktiviert wird
  - jedes Objekt hat eine „Hausmaschine", auch wenn ein Servant über mehrere Maschinen „verfügen" kann
  - Reine Applikationen, die keine Dienste zur Verfügung stellen, sondern nur welche nutzen, werden auch nicht registriert. Können damit auch nicht durch CORBA gestartet werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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)
- CORBA 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)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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.

- Unterscheide zwischen ORB als Klasse und ORB-Instanzen.
  - ORB als Klasse entspricht unserem Komponentenbegriff
- Aufbau und Organisation des ORB als Klasse abhängig von Anbieter und Einsatzgebiet als einzelner Prozess oder verteilte Anwendung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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
- CORBA-Objekte sind damit Zwischending zwischen Komponenten und Objekten im Sinne der Vorlesung
- 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

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

CORBA Objektreferenzen

- Statt Objekten werden normalerweise nur Referenzen übergeben
  - Seit CORBA 2.3 können Objekte auch als Wertparameter übergeben werden
  - 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 muss nicht mehr existieren

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
  - Damit kann Sprache verwendet werden, die nur einen Teil des Standards implementiert, wenn auch nur dieser Teil verwendet wird.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Java und CORBA

- CORBA als Interoperations-Standard muss an konkrete Programmiersprachen **gebunden** werden
- Seit CORBA 2.2 (1998) gibt es OMG IDL to Java binding sowie Java to OMG IDL reverse binding
  - Java als die wichtigste CORBA-Referenzimplementierung
- Reverse binding interessant für Anbindung von Nicht-Java-Systemen an Java-Systeme über IIOP-Standard von CORBA
- Heute Koexistenz in fast allen Applikationsserver-Produkten

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Entwicklung einer CORBA – Anwendung unter JAVA
(aus "Lehrbuch der Softwaretechnik" von Helmut Balzert)

1) Spezifikation der Schnittstelle in CORBA – IDL
2) Übersetzung mittels IDL – Compiler
   - Stummel- und Skelettklassen werden erzeugt
3) Implementierung der Operationen der Schnittstelle
4) Rahmenanwendung entwickeln
   - Erzeugt Objekt der Klasse
   - Objekt wird für Clients zugreifbar gemacht
5) Entwickeln des Clients

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Definition der Schnittstelle mit OMG IDL

```
module SemOrg { // Schnittstelle der Klasse Firma
    interface Firma {
        attribute string Name;
        attribute float Umsatz;
        // Operationssignatur
        float berechneGewinn( in float Kosten );
    };
};
```

SemOrg.idl

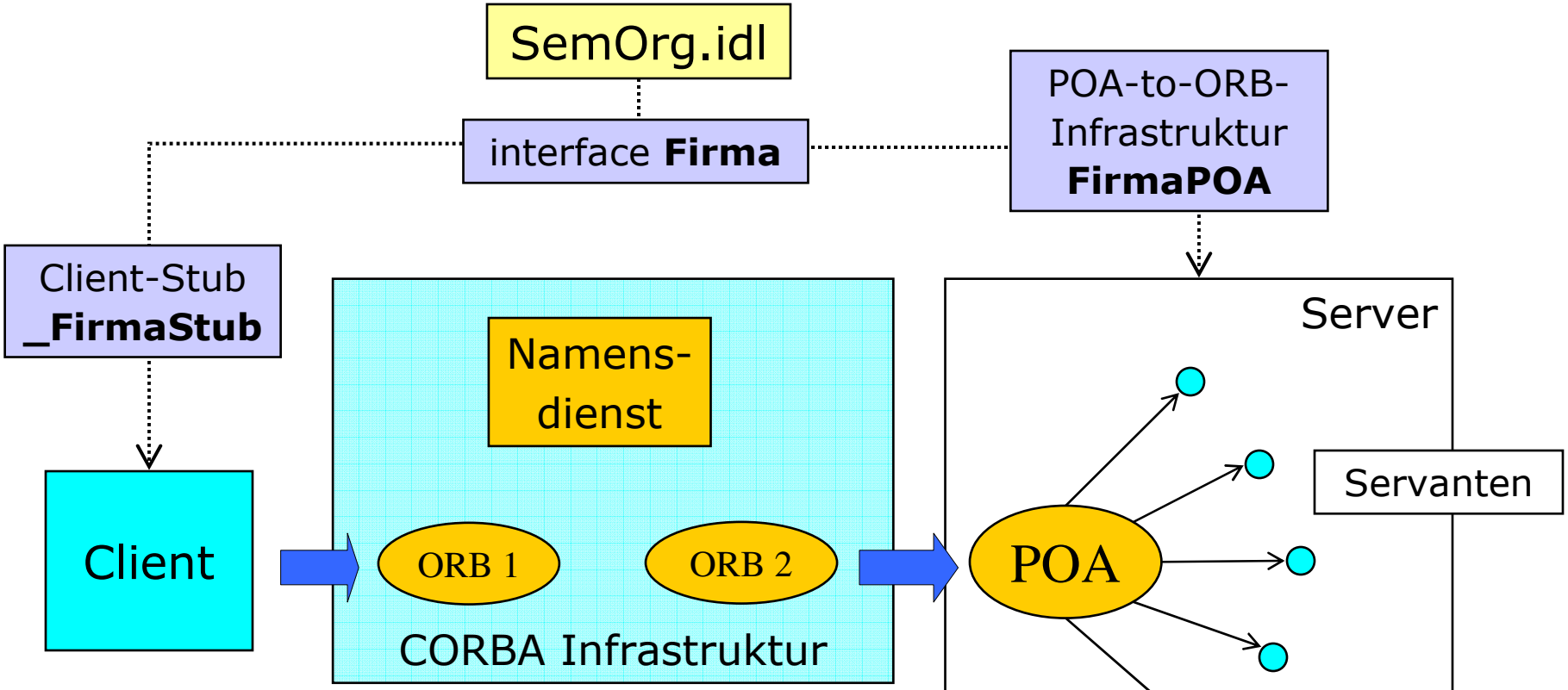
`idlj –f all SemOrg.idl` erzeugt daraus Java-Package **SemOrg**

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vom IDL – Compiler erzeugte Klassen

- interface **FirmaOperations**
  - interface Firma als Java-Interface
- interface **Firma** extends FirmaOperations, ...
  - Firma-Operations + CORBA.Object ...
- class **_FirmaStub** extends CORBA.portable.ObjectImpl
  implements SemOrg.Firma
  - voll generierte Stummel – Klasse
- final class **FirmaHolder** implements CORBA.portable.Streamable
- abstract class **FirmaHelper**
  - „Cast" von CORBA.Object zu Firma
- abstract class **FirmaPOA** extends PortableServer.Servant ....
  - portabler Objekt-Adapter
  - generierte Implementierung der ORB-seitigen Kommunikation

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



Client:
Hält eine CORBA-Referenz, in welcher der Objekt-Key steht.

ORB 2:
Kennt POA, der das Objekt mit dem Objekt-Key bedient

POA:
Kennt Zuordnung Objekt-Key ↔ Objekt

Dienst nutzen ⇦ Dienst finden ⇦ Dienst bekannt machen ⇦ Dienst als Subinstanz des POA generieren

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Implementierung des Servanten

```
package SemOrg;
public class FirmaImpl extends FirmaPOA {
    private String derName;
    private float derUmsatz;
    public FirmaImpl() {} // Konstruktor
    public float berechneGewinn(float Kosten)
        { return this.Umsatz – Kosten; }
    // get-/set-Operation für Attribut Name
    public String Name() { return this.derName }
    public void Name(String neuerName)
        { this.derName = neuerName; }
    //analog für Umsatz ...
}
```