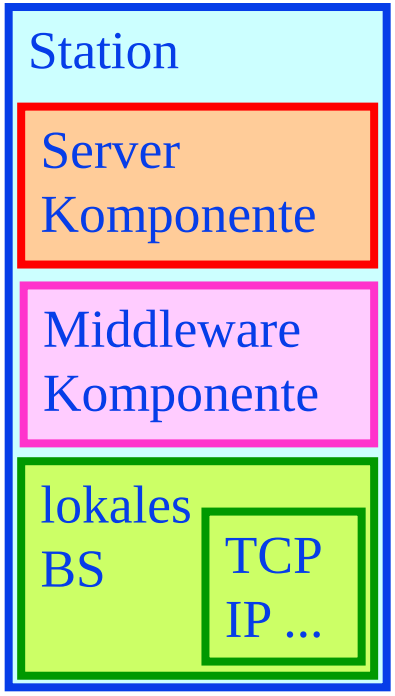
Rechnernetze und verteilte Systeme (BSRvS II)

Prof. Dr. Heiko Krumm

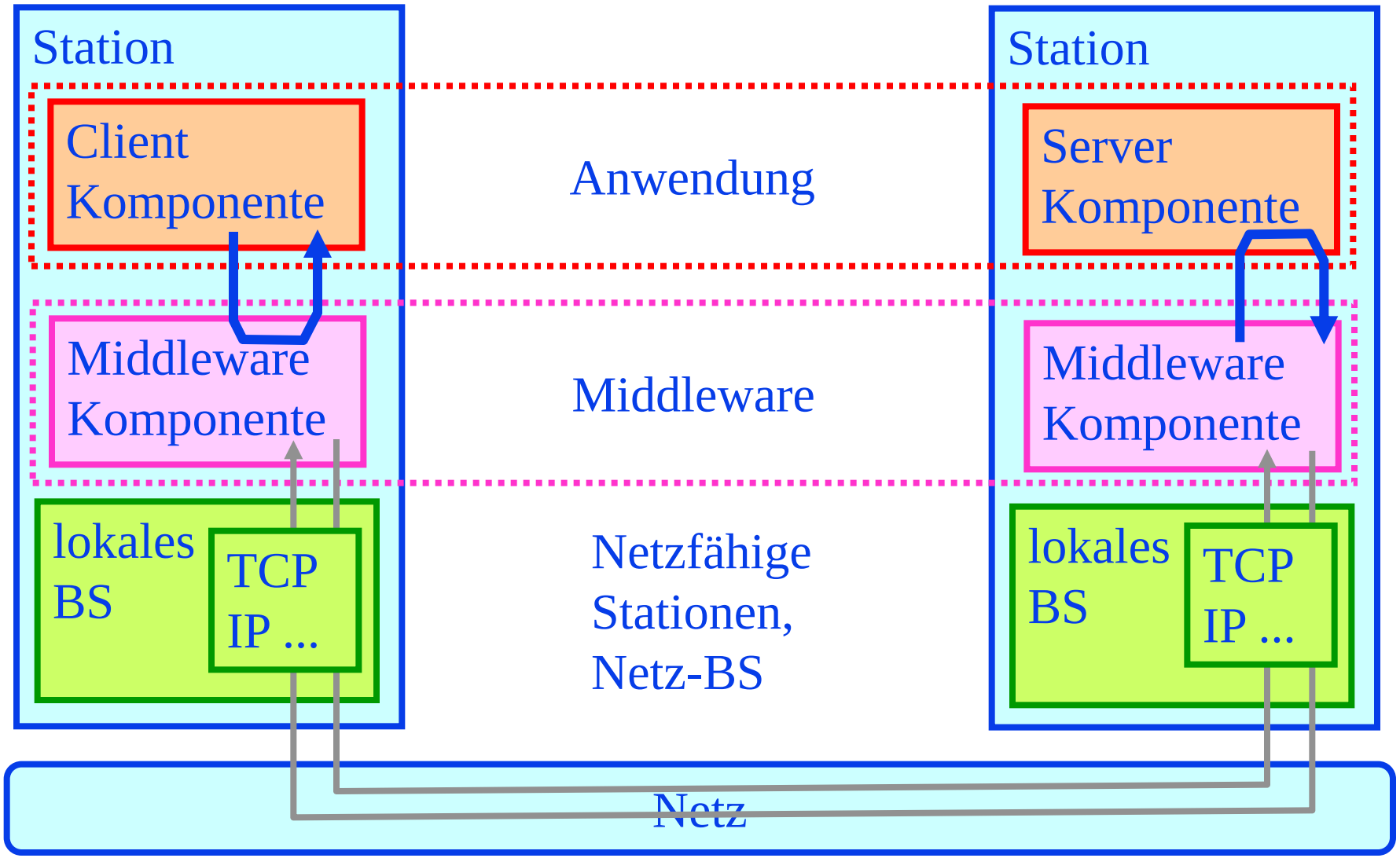
FB Informatik, LS IV, AG RvS
Universität Dortmund



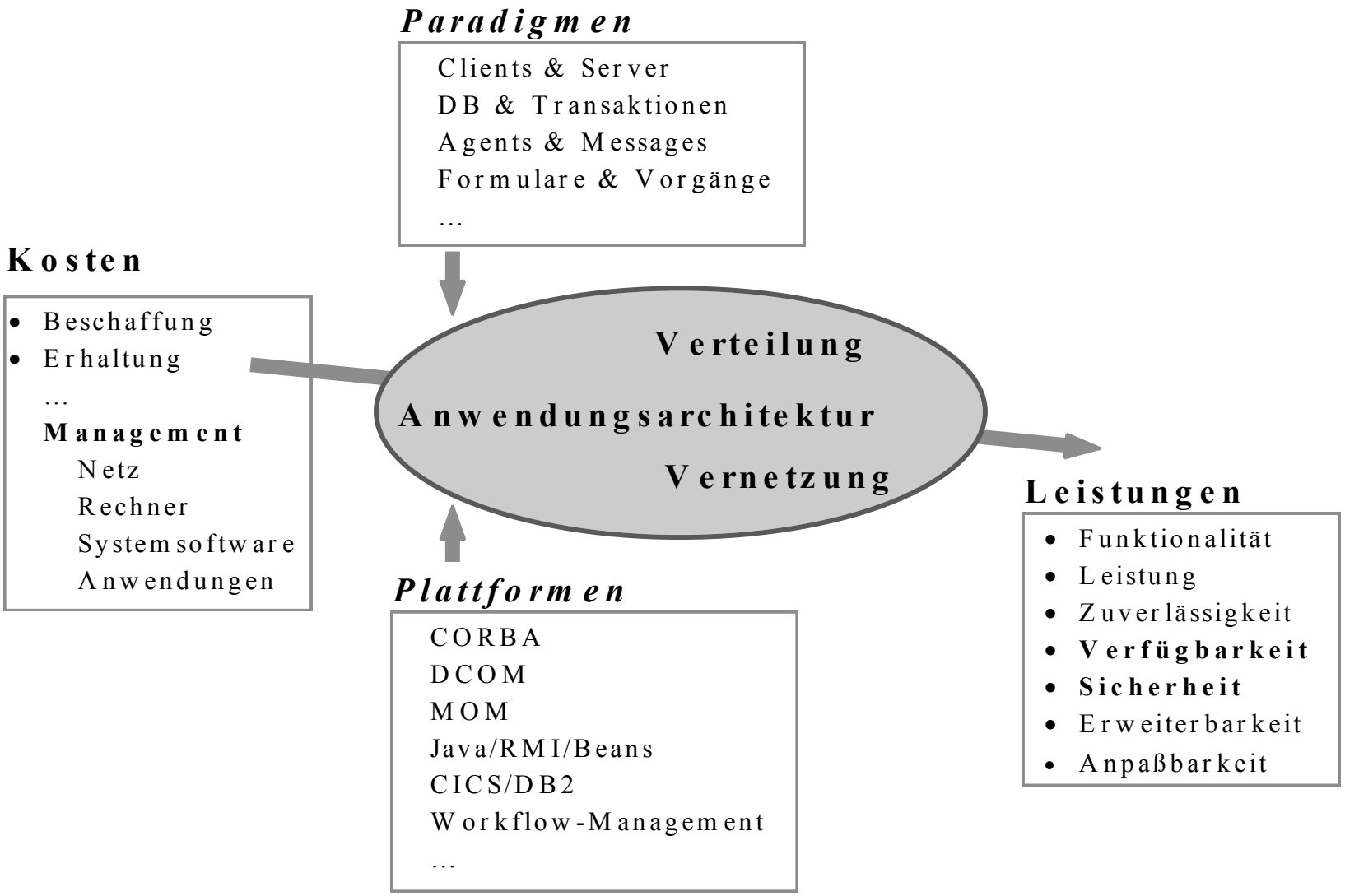
- Trends
- Übersicht
- Plattformtypen
- CORBA

- **Computernetze und das Internet**
- **Anwendung**
- **Transport**
- **Vermittlung**
- **Verbindung**
- **Multimedia**
- **Sicherheit**
- **Netzmanagement**
- **Middleware**
- **Verteilte Algorithmen**

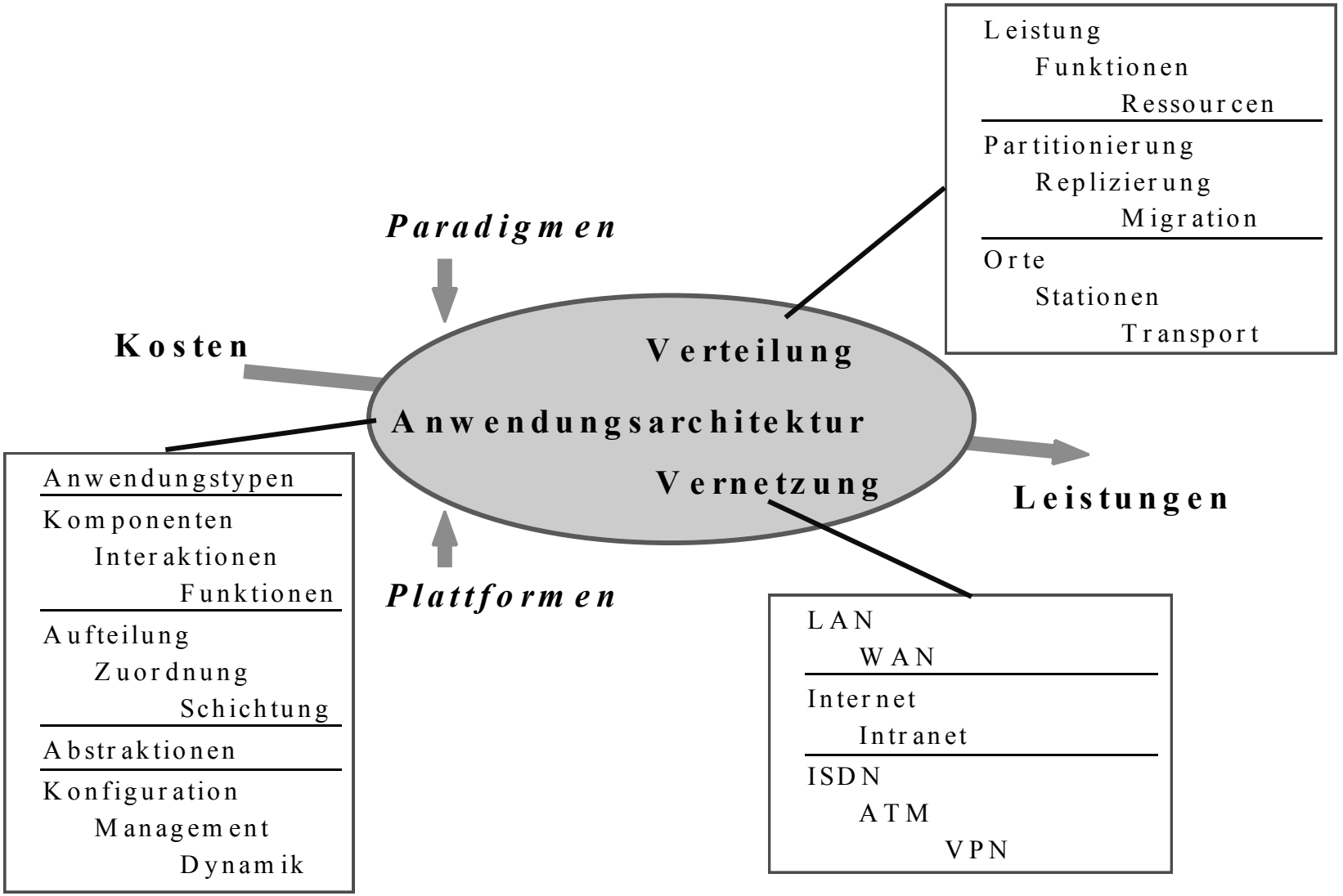
Middleware – Zwischen BS und Anwendung



Anwendungen – Entwicklungsaspekte



Anwendungen – Entwicklungsaspekte



Trends: Veränderte Anwenderanforderungen

Wachsende und vernetzte Problembereiche ——>
Menschen & rechnergestützte Services

Service - Verfügbarkeit: Immer und Überall

Service - Vielfalt:
– Speziell zugeschnittene Services
– Einheitliche Oberflächen
– Bedienung und Ergebnisverknüpfung

Internet - Einbindung
Hypermediale Browser - Oberflächen

Trends: Strukturierung der Anwendungen

„1-Tier-Monolith“ ——>
„2-Tier-Systeme“ ——>
„3-Tier-Systeme“ ——>
kommunizierende, kooperierende Komponenten

Presentation and MMI
Application (Client)
Application (Service)
Database and Resources

„Entzauberte Geheimnisse“, offene Schnittstellen, mündige Anwender
Modularität, Abstraktionen, Kapselung, Trennung von Belangen, Anpassbarkeit
Verschiedene Quellen zur freizügigen Integration von Expertise
Verschiedene Quellen zur verbesserten Marktsituation

Middleware-Plattformen: Übersicht

- **MOM (Message oriented Middleware)**
- **RPC (Remote Procedure Call)**
- **ORB (Object Request Broker)**
- **Componentware**
- ***SOA – Service Bus – Webservices***

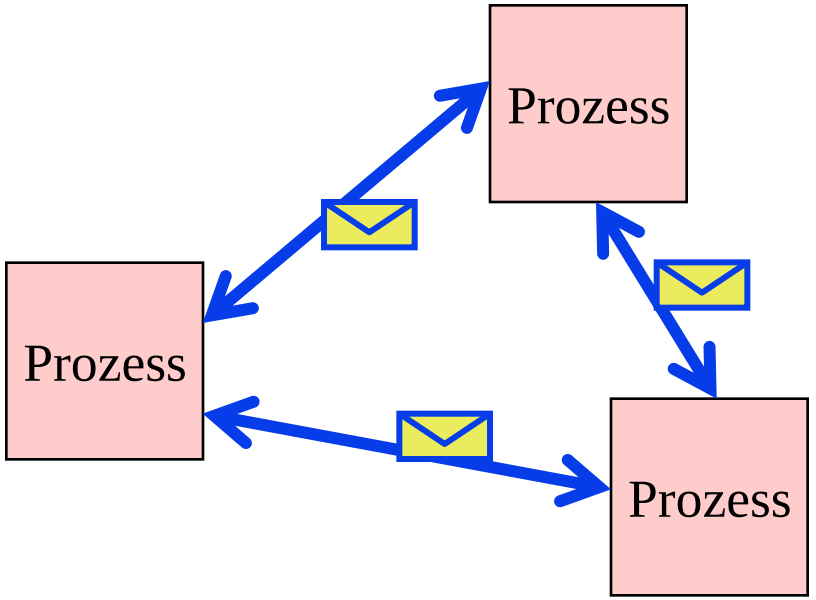
- **DBMS (Data Base Management Systems)**
- **Multi-DBMS**
- **TP (Transaction Processing)**
- **DTP (Distributed Transaction Processing)**

- **Workflow-Systeme**
- **Groupware-Systeme**

Plattformen: Message oriented Middleware

„Prozesse und Nachrichten“
z.B. netzfähige Unix-Stationen mit Socket-Schnittstelle zu TCP / UDP-Transportdiensten

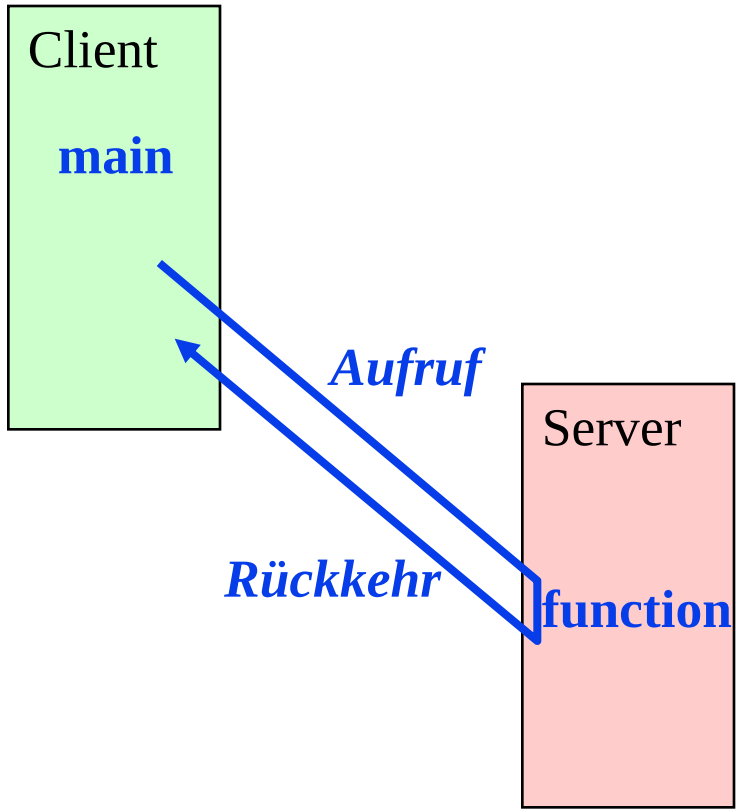
- **Adressierung**
- **Nachrichtenkonstruktion und Analyse**
- **Nachrichtenversand und Empfang**
- **Nachrichten-interpretation**
- **Schnittstellen-definition: Anwendungsprotokoll**



Plattformen: Remote Procedure Call

„Clients und Server“, z.B. Sun-RPC, Java-RMI

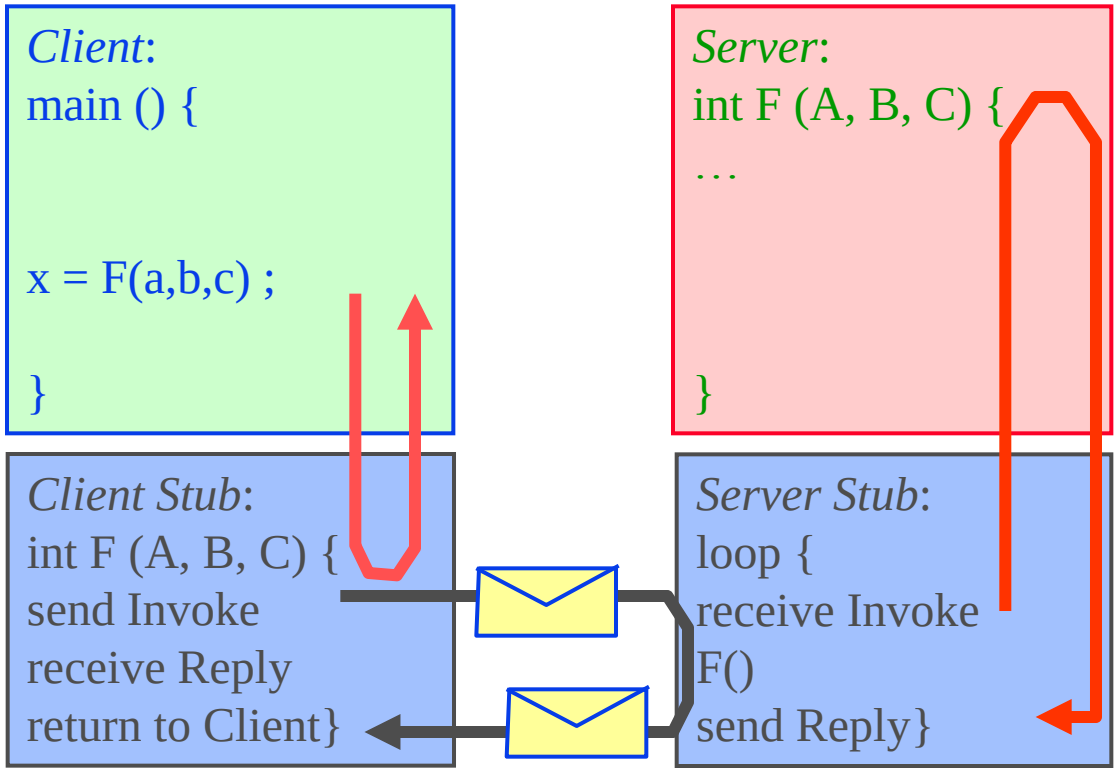
- **Adressierung**
- **Prozeduraufruf und Ausführung**
- **generierte Stubs**
- **IDL-Dokument: Schnittstellen-Definition (Prozedurköpfe)**



Sun RPC

Schnittstellen-
und
Kommunikations-
Implementierung
per
automatisch mittels
XDL-Compiler *rpcgen*
aus XDL-Datei
generierten
Stubs

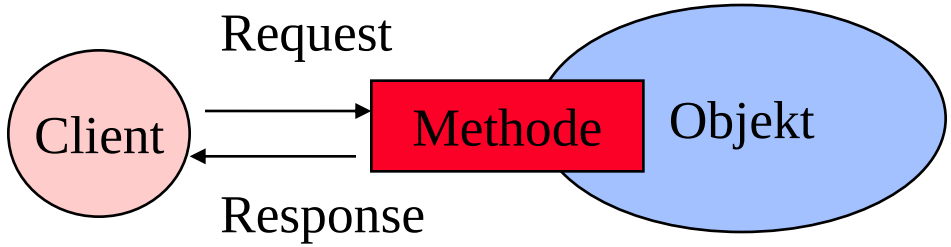
XDL
Interface Definition
Language

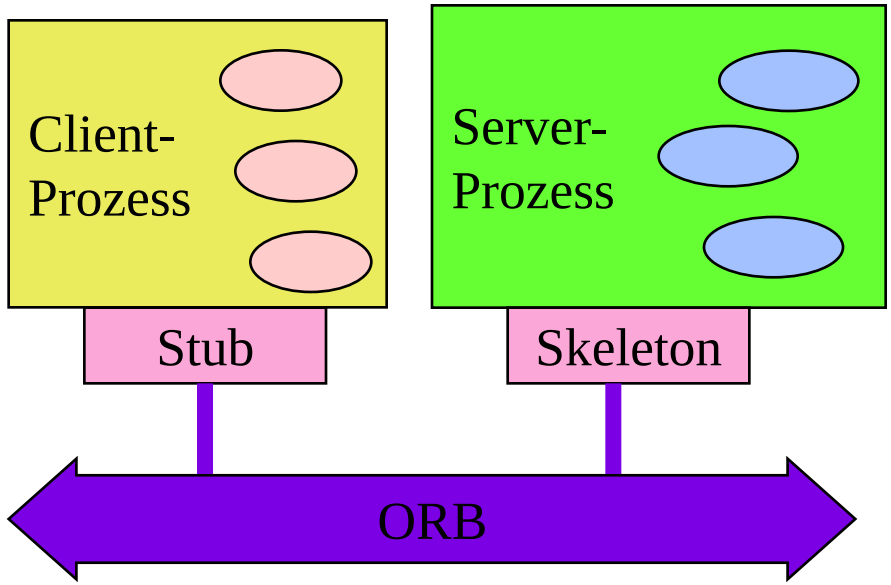


Plattformen: Object Request Broker

„Clients und gekapselte Objekte“, z.B. CORBA-ORB

- **Broking von Diensten**
- **Operationsaufruf und Ausführung: Objektmethoden**
- **generierte Stubs (statisch / dynamisch)**
- **IDL-Dokument (Operationsköpfe)**



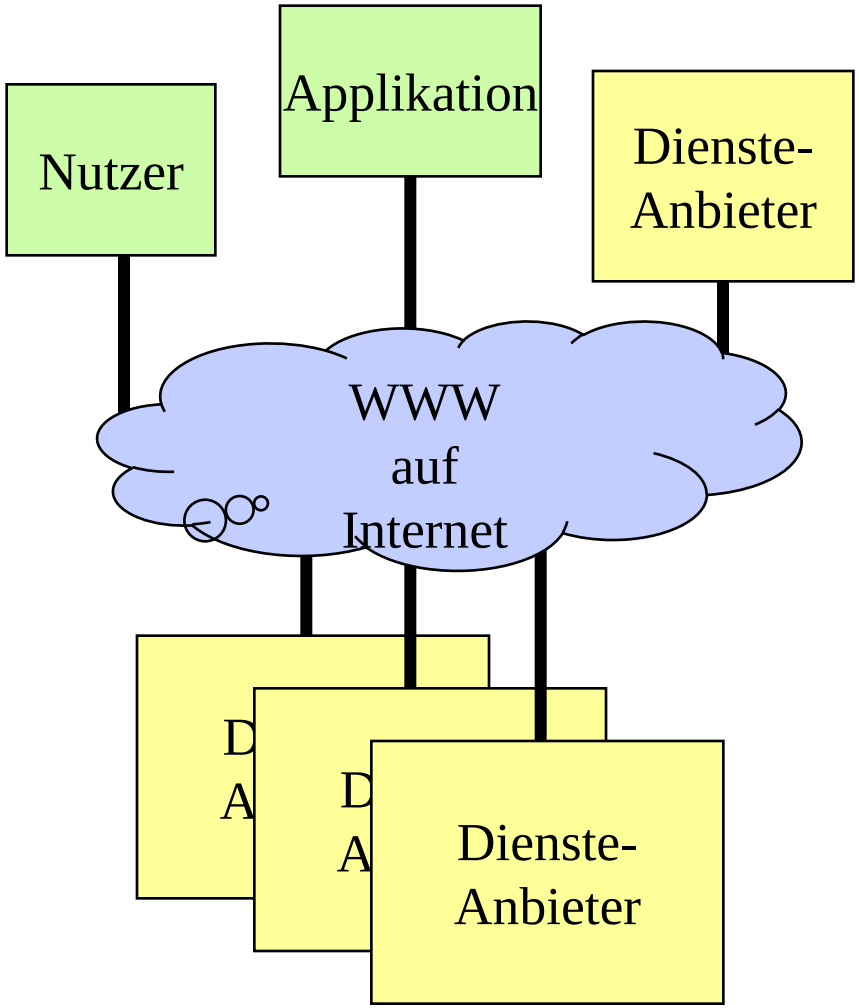


Plattformen: Service-Systems, SOA

„Dienstleister und Dienstleistungsbeziehungen“ , z.B. Web-Services

- **Broking und Vermittlung**
- **Temporäre Dienstleistungsbeziehungen**
- **Dienstanforderung und Ausführung**

- **Maschinen- und Menschen-Schnittstellen**

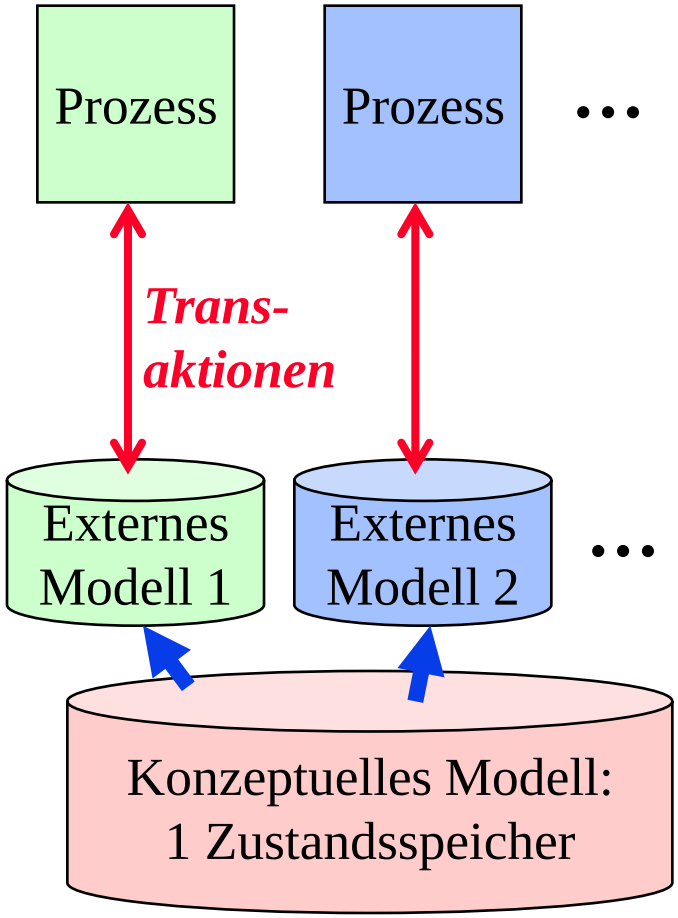


Plattformen: Database Management Systems

„Datenbasis und Transaktionen“

- **Zustandsraum: Datenmodell**
- **Transitionen: Aktionen**
- **Konsistenzbedingungen**
- **Transaktionen: *ACID***
  - **Logik: *AC***
  - **Modularer Entwurf: *I***
  - **Zuverlässigkeit: *D***

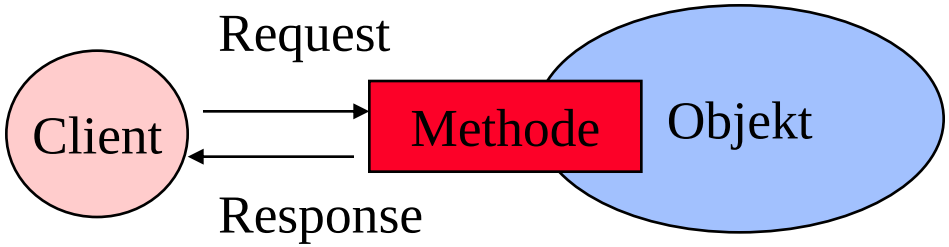
- **SQL-Schnittstellen**
- **Interface-Builder**

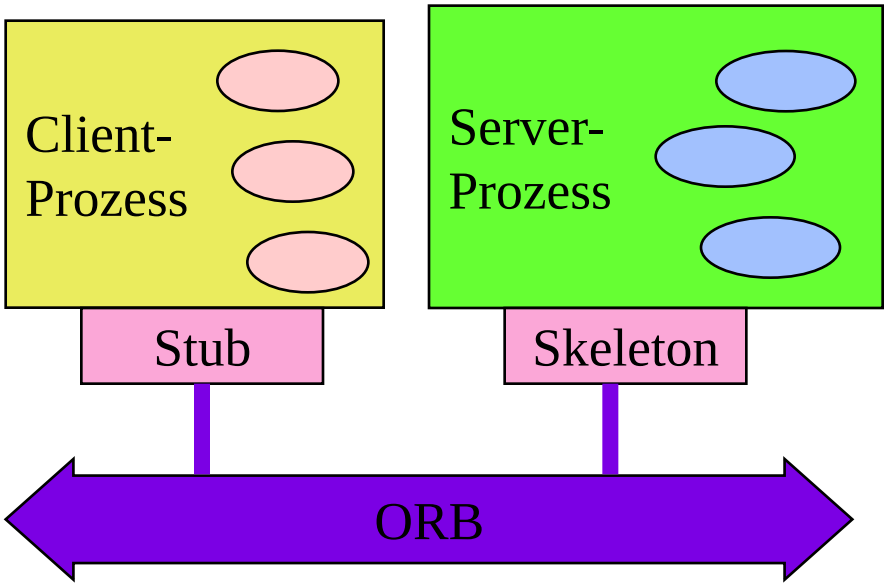


Plattformen: Common Object Request Broker (CORBA)

„Vermittlung des Zugangs zu und Zugriff auf entfernte Objekte sowie spezielle Services und Hilfsdienste“

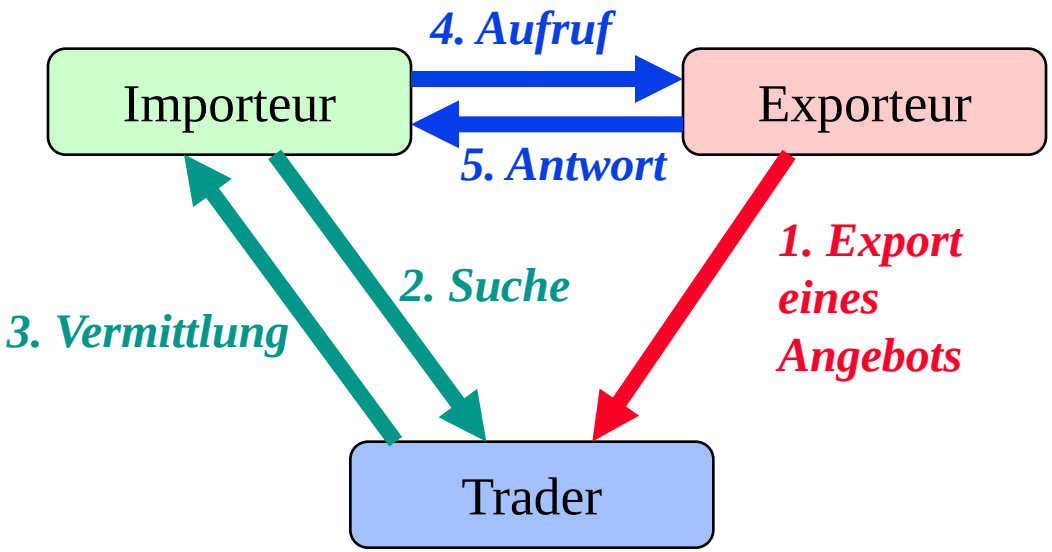
- **Spezialitäten**
  - **Erblast – Integration**
  - **Interoperabilität mit anderen Plattformen**
  - **problemlose Netzeinbindung (IIOP)**
  - **vereinfachtes Management**





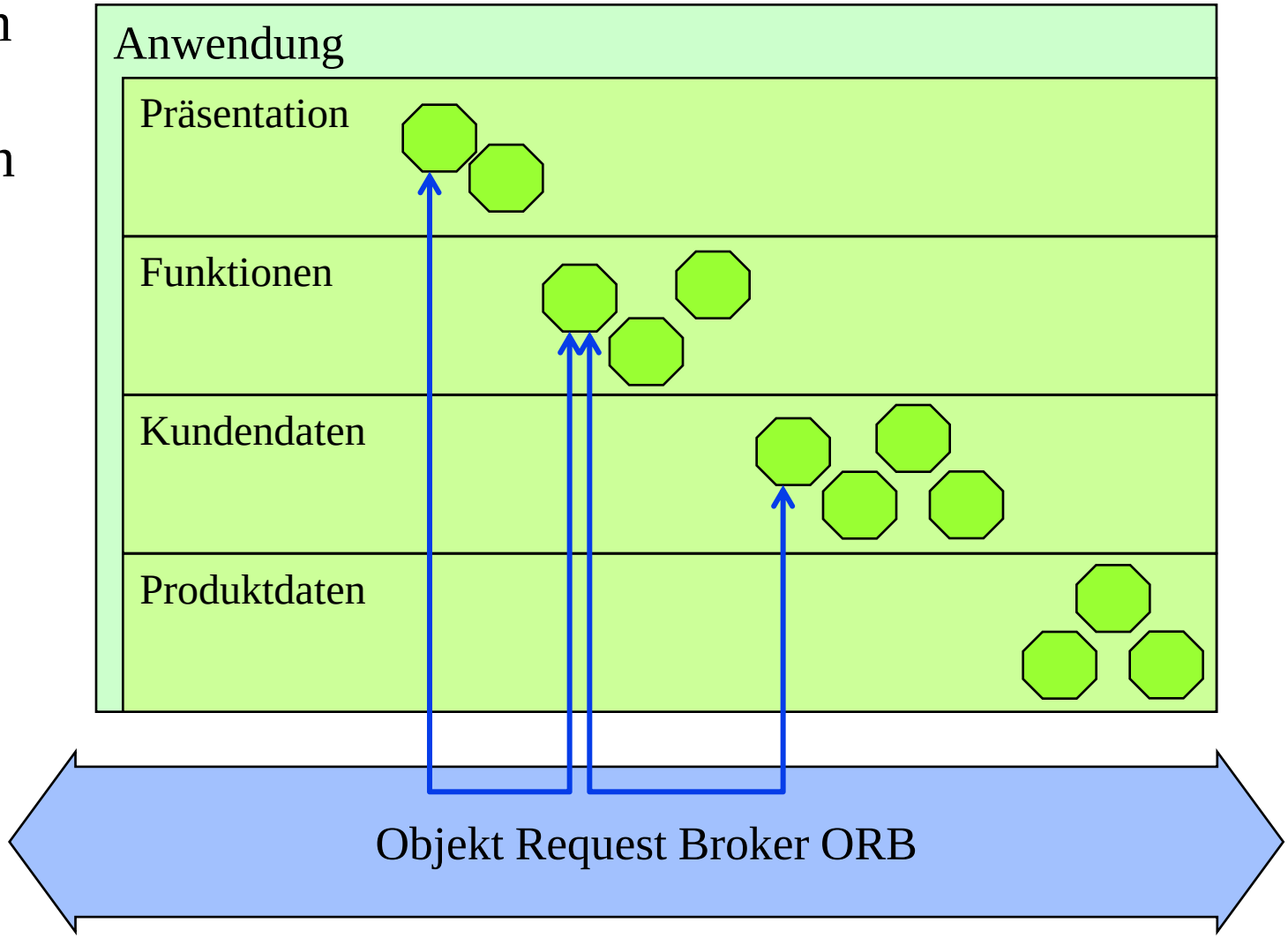
CORBA: Dienstevermittlung

- Dienst:
  Objekt-Implementierung
  - Anforderungen / Dienstleistungen: Methoden-Aufrufe
- Schnittstellen-Verzeichnis:
  Typen
- Implementierungs-Verzeichnis:
  Konkrete Dienstleister



CORBA: Struktur einer Anwendung

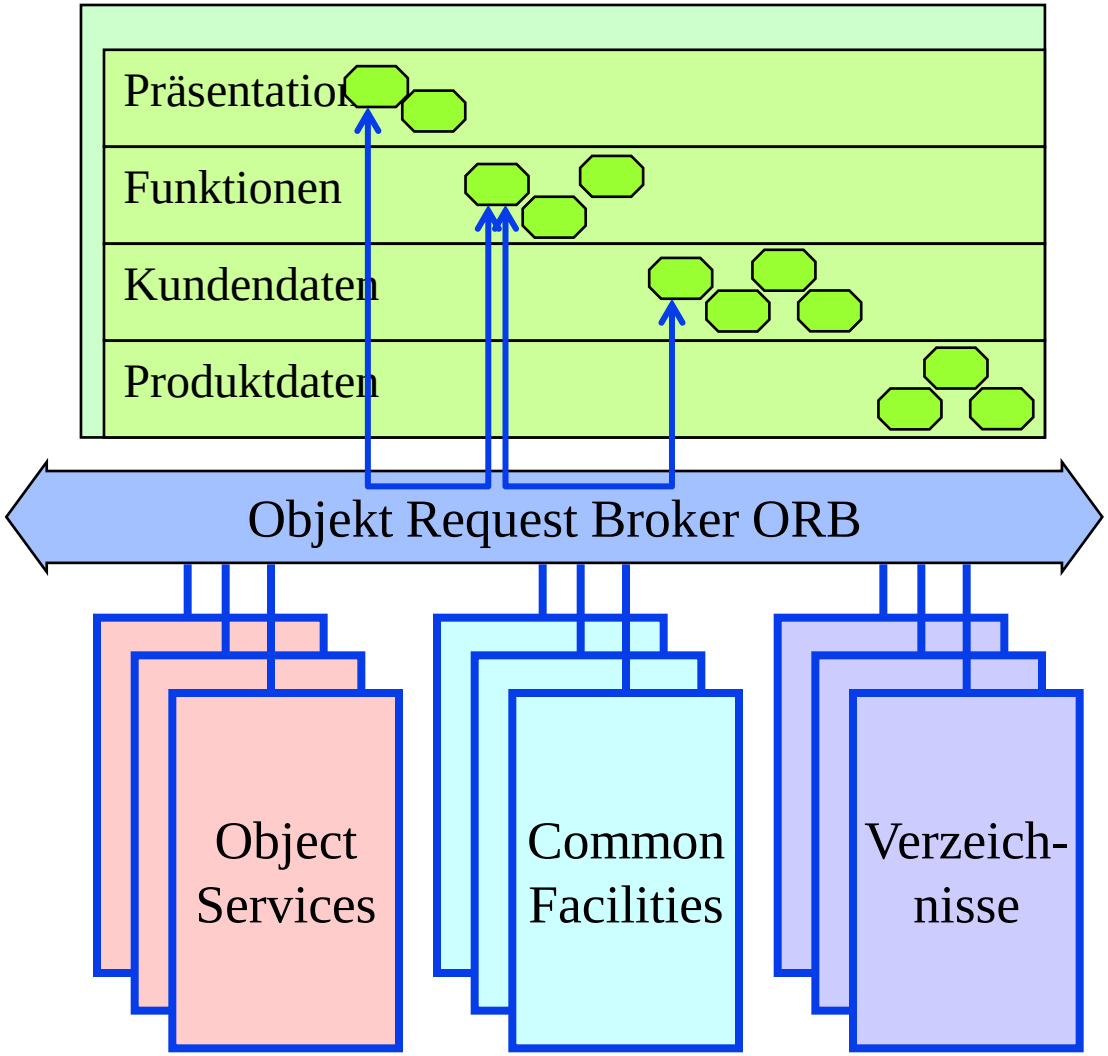
- Anwendungen erscheinen als Objektmengen



CORBA: Elemente

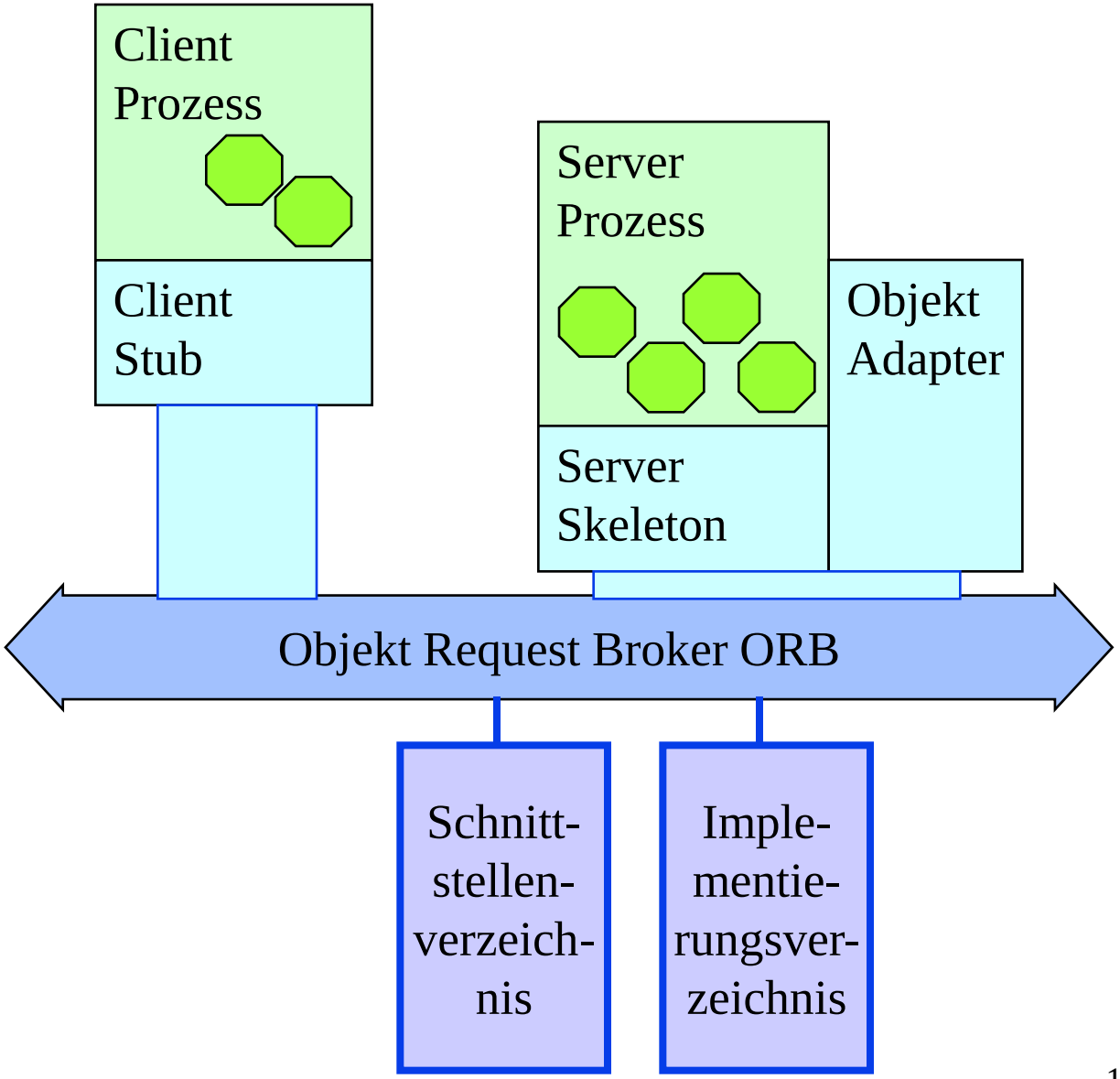
Object Services:
Unterstützung des verteilten Objektsystems
- Sicherheitsdienste
- Ereignisaustausch
- Messages
- Abrechnung
- Persistenz
- Objekt-Lifecycle
- …

Common Facilities:
Sammlung allgemein nützlicher Objekte
- Drucken
- Fehlerbehandlung
- Oberfläche
- …



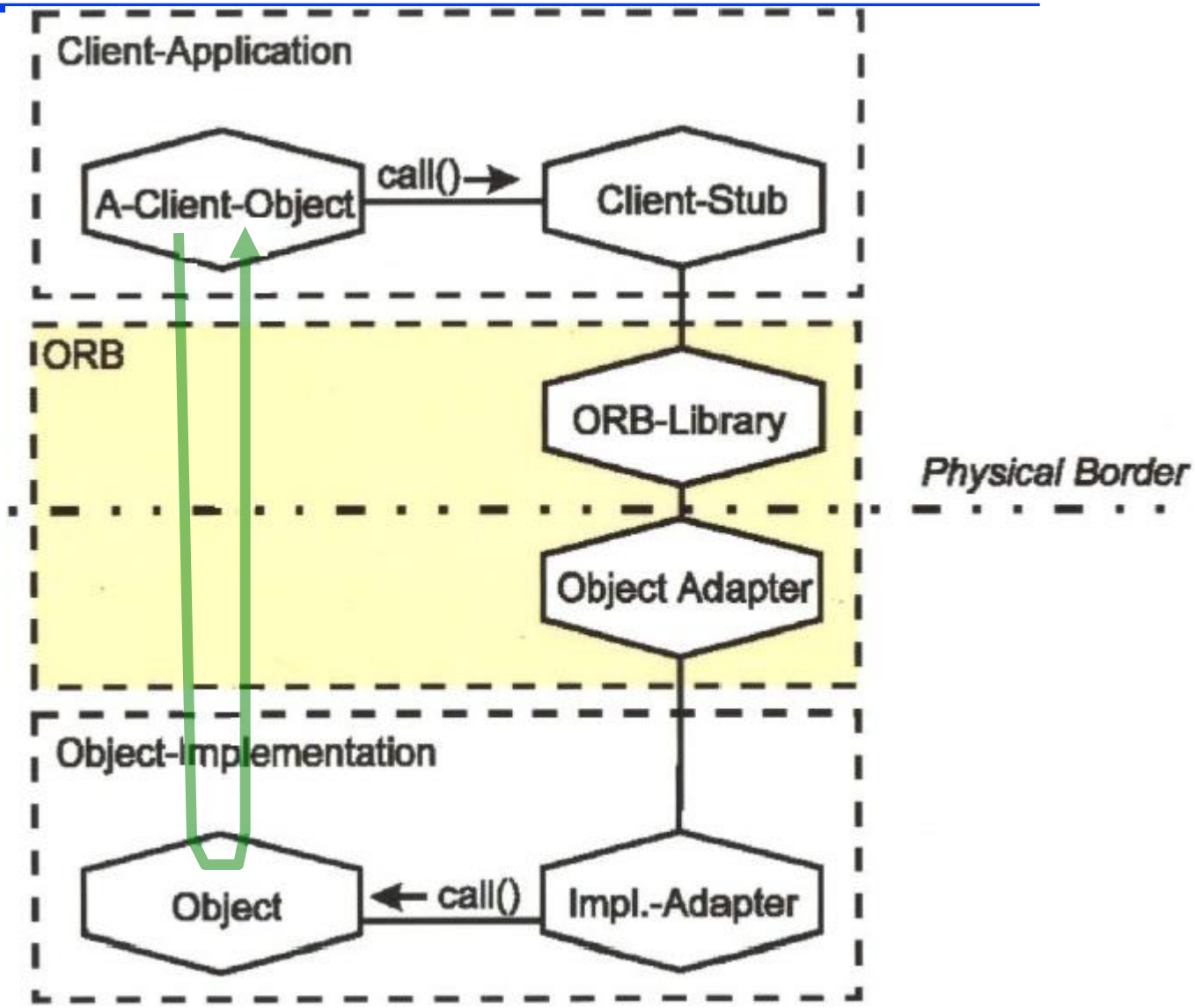
CORBA: SW-Architektur

- Software-System besteht aus:
  - Client-Prozessen
  - Server-Prozessen
  - ORB
  - Verzeichnissen
- Adapter (generiert) zur Anbindung der Prozesse und Objekt-Implementierungen an ORB

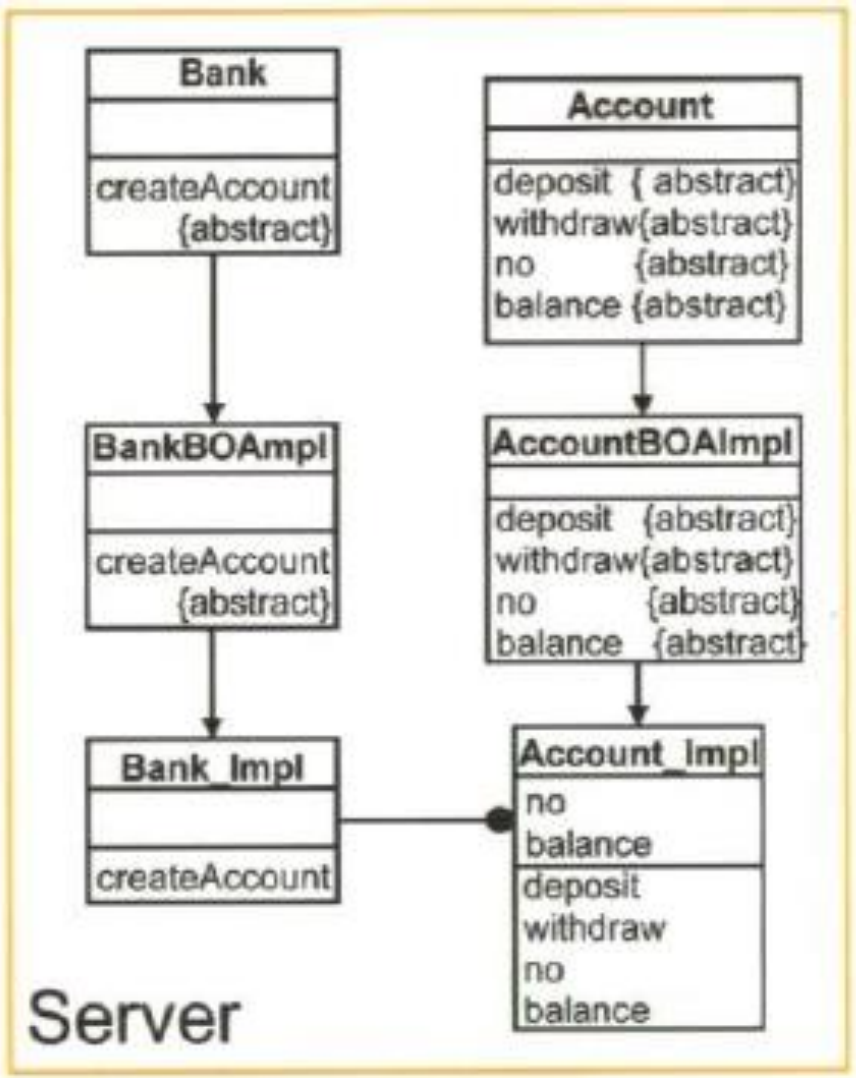


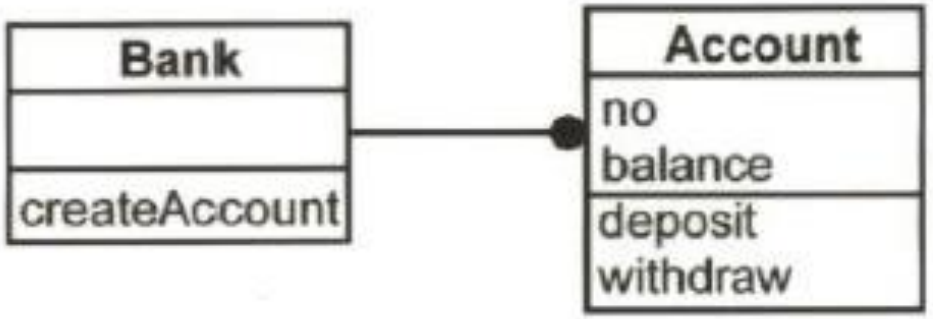
CORBA: SW-Architektur

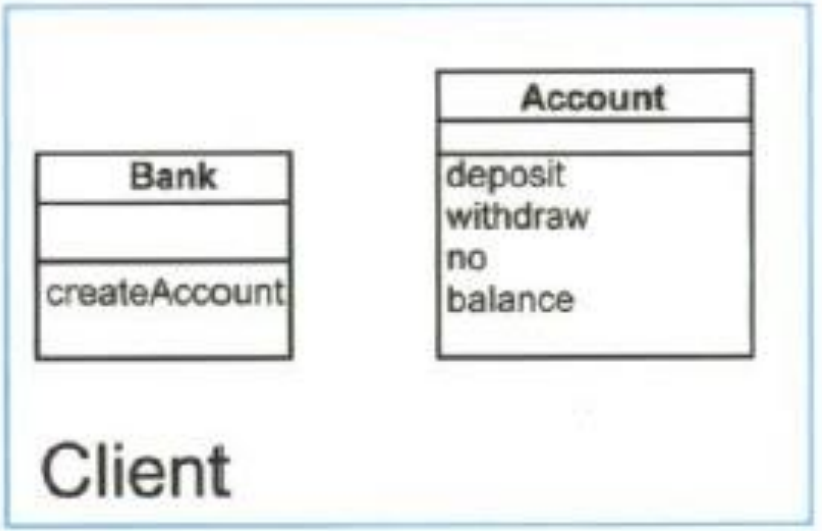
- Software-System besteht aus:
  - Client-Prozessen
  - Server-Prozessen
  - ORB
  - Verzeichnissen
- Adapter (generiert) zur Anbindung der Prozesse und Objekt-Implementierungen an ORB



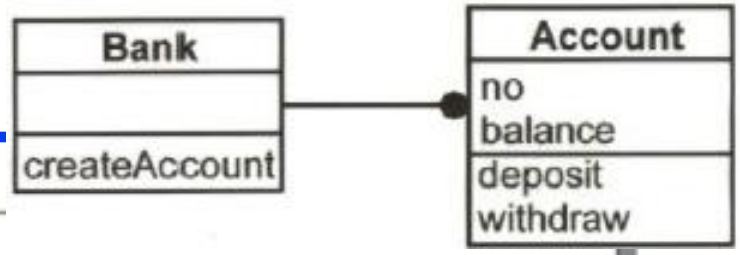
CORBA: Schnittstellendefinition-Beispiel Bankkonten







CORBA: IDL-Beispiel Bankkonten




```
interface Account {
    readonly attribute double m_balance;
    readonly attribute long   m_number;

    exception TooMuch { };

    void withdraw(in double amount) raises (TooMuch);
    void deposit(in double amount);
};


interface Bank {

        exception Unknown { };

        Account createAccount(in long no) raises (Unknown);
};
```

CORBA: Client - Kontoeröffnung

```
# include <iostream.h>
# include "bank.hh"

int
main()

{
    Bank_var bank = Bank::_bind(":BankSrv");

    try {
        Account_var acc = bank -> createAccount(1015);
        acc -> deposit(5000);
        acc -> withdraw(200);
        cout << "amount = " << acc -> m_balance() << ".\n";
    } catch (...) {
        cout << "Exception.\n";
    }

    return 0;
}
```

CORBA: Server: Bank-Implementierung

```
class Bank_Impl : public BankBOAImpl {
public:
  virtual Account_ptr createAccount (CORBA_Long no,
                      CORBA_Environment &IT_env=CORBA_IT_chooseDefaultEnv ());

};

Account_ptr
Bank_Impl::createAccount(CORBA_Long no, CORBA_Environment &IT_env)

{
    if (no>0) {
        Account_ptr result = new Account_Impl(no);
        Account::_duplicate(result);
        return result;
    } else {
        throw Bank_Unknown();
    }

}
```

CORBA: Server: Account-Implementierung

```
class Account_Impl : public AccountBOAImpl {
public:
    Account_Impl(CORBA_Long no) { m_no = no; m_bal = 0; }
    virtual CORBA_Double m_balance (CORBA_Environment &IT_env) { return m_bal; }
    virtual CORBA_Long m_number (CORBA_Environment &IT_env) { return m_no; }
    virtual void withdraw (CORBA_Double amount, CORBA_Environment &IT_env);
    virtual void deposit (CORBA_Double amount, CORBA_Environment &IT_env);
private:
    CORBA_Long      m_no;
    CORBA_Double    m_bal;
};

void
Account_Impl::withdraw (CORBA_Double amount, CORBA_Environment &IT_env)

{
    cout << "withdraw " << amount << ".\n" << flush;
    if (m_bal > amount)
        m_bal -= amount;
    else
        throw Account_TooMuch();
}
```

CORBA: Server: Hauptprogramm

```
# include <iostream.h>
# include "bank.hh"


int
main()

{
    Bank_Impl myBank;

    CORBA_Orbix.impl_is_ready();

    return 0;
}
```