G Middleware und verteilte Anwendungen: CORBA

1 Überblick

- Motivation
- Object Management Architecture (OMA)
- Anwendungsobjekte und IDL
- Object Request Broker (ORB)
- Portable Object Adaptor (POA)
- CORBA Services

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Literatur, URLs

HeVi99. Michi Henning, Steve Vinosk: *Advanced CORBA Programming with C++*. Addison-Wesley, 1999.

BrVD01. Gerald Brose, Andreas Vogel, Keith Duddy: Java Programming with CORBA. 3rd Edition, Wiley, 2001.

OMG99. Object Management Group, OMG: *The Common Object Request Broker: Architecture and Specification.* Rev. 2.3.1, OMG Doc. formal/99-10-07, Oct. 1999.

Pope98. A. Pope: *The CORBA Reference Guide*. Addison-Wesley. 1998.

Linn98. C. Linnhoff-Popien: *CORBA, Kommunikation und Management*. Springer, 1998.

TaBu01. Zahir Taki, Omran Bukhres: *Fundamentals of Distributed Object Systems*. Wiley, 2001.

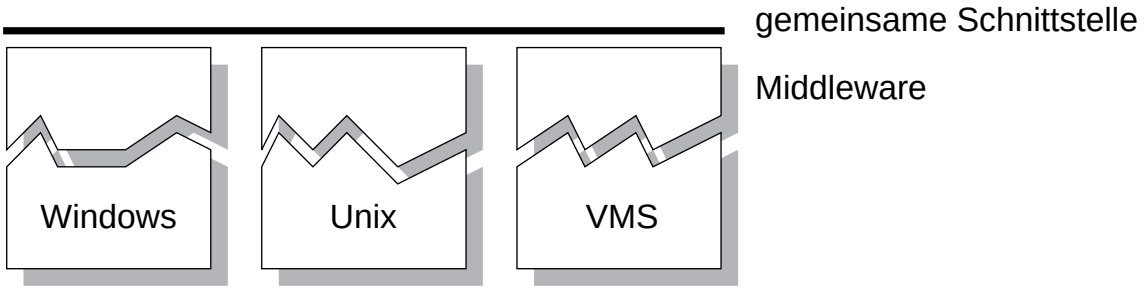
Daneben vor allem online-Informationen

- OMG
  `http://www.omg.org/gettingstarted/`
  offizielle Seiten der Object Management Group
- Douglas C. Schmidt
  `http://www.cs.wustl.edu/~schmidt/corba.html`
  sehr gute Informationssammlung

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.1 Middleware für verteiltes Programmieren

- Middleware – Software zwischen Betriebssystem und Anwendung



  - ◆ Bereitstellung von Diensten für verteilte Anwendungen
  - ◆ **gesucht:** Middleware für verteiltes objektorientiertes Programmieren

    - Middleware ist klassisch eine Softwarekomponente, die Dienste für verteilte Anwendungen bereitstellt. Dabei abstrahiert Middleware von der eingesetzten Hardware- und Softwarearchitektur und bietet eine gemeinsame Schnittstelle an.

- CORBA – Common Object Request Broker Architecture
  Standard der OMG

    - Gemeinsame Architektur für einen Object Request Broker (ORB). Ein ORB ist eine Vermittlungseinheit für den Aufruf von verteilten Objekten. Das „gemeinsam“ bezieht sich auf die gemeinsame Anstrengung der wichtigsten IT Firmen in diesem Bereich, zusammengeschlossen in der OMG (Object Management Group), einen Standard für verteiltes Programmieren mit Objekten zu entwickeln.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.2 CORBA-Überblick

1 Standard

- CORBA = Common Object Request Broker Architecture
  - plattformunabhängige Middleware-Architektur für verteilte Objekte
- OMG = Object Management Group
  - Standardisierungsorganisation mit etwa 1000 Mitgliedern
  - Teilbereiche der Standardisierung
    - CORBA
    - UML
    - XMI
    - ...
- Formaler Standardisierungprozess
  - Standard-Dokumente
  - Definition von CORBA -Kompatibilität

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Motivation

- Verteilte objektbasierte Programmierung
  - ◆ verteilte Objekte mit definierter Schnittstelle
- Heterogenität in Verteilten Systemen
  - ◆ verschiedene Hardware-Architekturen
  - ◆ verschiedene Betriebssysteme und Betriebssystem-Architekturen
  - ◆ verschiedene Programmiersprachen
    - ➤ Transparenz der Heterogenität
- Dienste im verteilten System
  - ◆ Namensdienst
  - ◆ Zeitdienst
  - ◆ Transaktionsdienst
  - ◆ …

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Entwurfsziele

- CORBA ist ein Standard
  - Standard-Dokumente
  - Definition von "CORBA compliance"
    - ➤ CORBA schreibt keine Implementierung vor sondern Funktionalität
    - ➤ CORBA fordert Interoperabilität verschiedener Implementierungen
  - Hersteller realisieren Implementierungen des Standards (z. B., VisiBroker, Orbix, Orbacus, MICO, etc.)
- CORBA ist eine Middleware zur Abstraktion von
  - Hardware,
  - Betriebssystem und
  - Programmiersprache

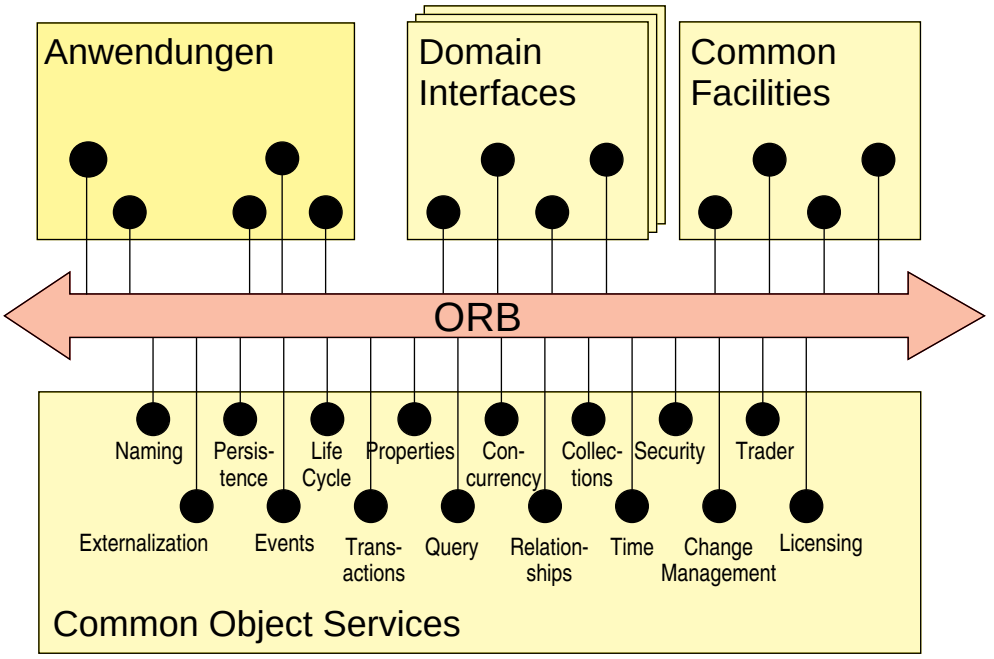
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Entwurfsziele (2)

- Interoperabilität
  - Eine Anwendung soll ohne Änderungen auf verschiedenen CORBA-Implementierungen ablaufen können
  - Anwendungen auf verschiedenen CORBA-Implementierungen sollen miteinander kommunizieren können
    - Mit Hilfe von CORBA soll nicht nur von Hardware, Betriebssystem und Programmiersprache abstrahiert werden, sondern auch von der Implementierung des CORBA Systems selbst.
- Einbindung von Altapplikationen ("Legacy-Anwendungen")
  - Kapselung von Altanwendungen in CORBA Objekte
    - Ein CORBA Objekt verdeckt die Altapplikation. Aufrufe an dem Objekt werden in entsprechende Interaktionen mit der Altapplikation umgesetzt.
- Vision der Business Objects / Components
  - alle Geschäftsdaten und -vorfälle sind CORBA Objekte
    - Damit würde die gesamte Geschäftswelt die „gleiche Sprache“ sprechen und mit Hilfe von CORBA-Objektinteraktionen miteinander in Kontakt stehen können.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 OMA – Object Management Architecture



- Die CORBA Architektur besteht aus dem ORB (Object Request Broker). Dieser vermittelt Aufrufe zwischen Objekten. Der ORB weiß wo welches Objekt liegt. Die Anwendungsobjekte können untereinander mit Hilfe des ORB kommunizieren. Die Anwendungsobjekte stehen untereinander in Client/Server-Beziehung.
- Die Common Object Services sind allgemeine, anwendungsunabhängige, standardisierte Systemdienste, die eine CORBA Implementierung anbieten kann bzw. muss, z.B. den Naming Service zum Auffinden von Objekten anhand eines Namens oder einen Transaktionsdienst zur Unterstützung von Transaktionen im verteilten System. CORBA-Services verhalten sich wie normale Objekte.
- Common Facilities sind ähnlich wie Object Services unabhängig von einem bestimmten Anwendungsbereich. Im Gegensatz zu Services legen sie aber nicht Schnittstellen für Systemdienste, sondern Schnittstellen zu Benutzeranwendungen (z. B. für Dokumentenbearbeitung oder Grafikschnittstellen) fest.
- Domain Interfaces legen ähnlich wie Facilities Anwendungsschnittstellen fest, die sich allerdings an bestimmten Anwendungsbereichen oder Branchen orientieren. Beispielsweise Product Data Management (PDM), Telekommunikation, Dienste des Gesundheitswesens oder Finanzanwendungen.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 CORBA-Implementierungen

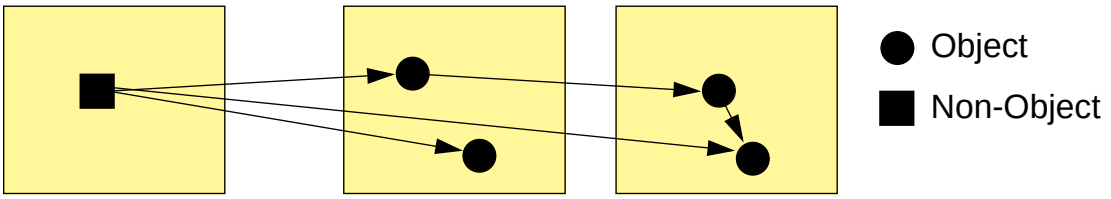
- Eine CORBA-Implementierung muss enthalten:
  - die Implementierung der Kern-Architektur
  - eine Sprachabbildung (z. B. für C++)
- Eine CORBA-Implementierung kann ausserdem enthalten:
  - eine beliebige Zahl von Services
  - Funktionalität für die Interoperabilität mit anderen CORBA-Implementierungen (GIOP, IIOP)
- **Wie** die Implementierung die Anforderungen des Standards realisiert bleibt freigestellt
  - unterschiedliche Implementierungen sind möglich
    - ➤ als Daemon
    - ➤ als Bibliothek
    - ➤ ...

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.3 CORBA-Anwendungsobjekte

1 Verteilte CORBA-Objekte

- haben Identität, Zustand, Methoden
- bilden eine Anwendung
- Kommunikation zwischen den Objekten über ORB
- CORBA-Objekte können aufgerufen werden (realisieren Server-Seite)
- CORBA-Objekte können als Aufrufer (Client) agieren



➤ Aufrufer müssen nicht unbedingt CORBA-Objekte sein

- Da Clients keine Objekte sein müssen, muss man kein CORBA-Objekt erzeugen, um mit einem anderen Objekt in Interaktion zu treten.
- Auch Prozesse, etc. konnen über entsprechende Schnittstellenfunktionen CORBA-Objekte aufrufen.

Achtung:
Server-Objekt darf nicht mit einem CORBA-Server verwechselt werden.

- Server-Objekte sind Server im Sinne der Client/Server-Interaktion. Sie werden über Methodenaufrufe angesprochen.
- CORBA-Server sind Einheiten des darunterliegenden Betriebssystems, die CORBA-Objekte beherbergen, z.B. UNIX-Prozesse.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

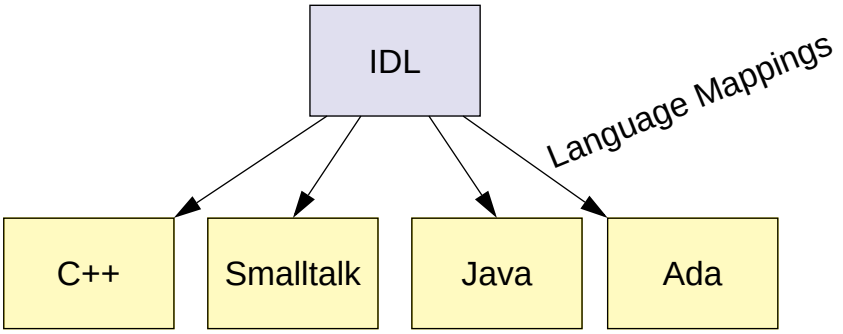
1 Verteilte CORBA-Objekte (2)

- Verteilte Objekte bilden eine Anwendung
  - ➤ Anwendungsobjekte liegen auf verschiedenen Rechnern
    - Die Anwendungsobjekte sind die Einheiten der Verteilung in CORBA. Objekte auf einem Rechner können auf eine oder mehrere Systemeinheiten verteilt sein, z. B. auf mehrere Prozesse eines UNIX Systems.
  - ➤ Sie finden sich und kommunizieren miteinander
    - CORBA bietet den Objekten eine transparente Kommunikation, d.h. die Objekte müssen nicht wissen, wo der jeweilige Kommunikationspartner liegt.
- ★ Beispiel Druckmanagement-System
  - ➤ Client-,
  - ➤ Spooler- und
  - ➤ Druckerobjekte
    - Verschiedene Anwendungsobjekte liegen verteilt im System. Sie sind für den Zugriff auf das Drucksystem (Clients), für das Puffern von Druckaufträgen (Spooler) und für das eigentliche Drucken zuständig.
- implizite, nicht-orthogonale Interaktion
  - ◆ (Remote-)Methodenaufrufe
  - ◆ Client-Stub vermittelt Aufruf an Server-Objekt
  - ◆ **At-most-once** / **exactly-once** Semantik

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Interface Definition Language (IDL)

- Sprache zur Beschreibung von Objekt-Schnittstellen
  - unabhängig von der Implementierungssprache des Objekts
  - Sprachabbildung (Language-Mapping) definiert, wie IDL-Konstrukte in die Konzepte einer bestimmten Programmiersprache abgebildet werden
  - Language-Mapping ist Teil des CORBA standards
  - Language mappings sind festgelegt für:
    - Ada, C, C++, Java, Lisp, PL/I, Python, Smalltalk
    - weitere inoffizielle Language-Mappings existieren,z.B. für Perl



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Interface-Definition-Language (2)

- IDL angelehnt an C++
  - ◆ geringer Lernaufwand
  - ◆ eigene Datentypen
    - ➤ Basistypen
    - ➤ Aufzählungstyp
    - ➤ zusammengesetzte Typen
  - ◆ Module
    - ➤ definieren hierarchischen Namensraum für Anwendungsschnittstellen und -typen
  - ◆ Schnittstellen
    - ➤ beschreiben einen von außen wahrnehmbaren Objekttyp
  - ◆ Exceptions
    - ➤ beschreiben Ausnahmebedingungen

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Interface-Definition-Language (3)

- Beispiel: Kontoschnittstelle

```
exception WrongAccountNumber { string errorMsg; };

interface Account
{
    readonly attribute double balance;

    double withdraw( in double amount )
        raises WrongAccountNumber;
    double deposit( in double amount );
        raises WrongAccountNumber;
};
```

  - ◆ verteilte Objekte können **`Account`**-Schnittstelle implementieren

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Interface-Definition-Language (4)

- Umsetzung von IDL in Sprachkonstrukte (z.B. C++)

IDL

```
module MyModule
{
    interface MyInterface
    {
        attribute long lines;
        void printLine( in string toPrint );
    };
};
```

C++

```
namespace MyModule {
    class MyInterface : ... {
        public:
         virtual CORBA::Long lines();
         virtual void lines( CORBA::Long _val );
         void printLine( const char *toPrint );
        ...
    };
};
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3.2Interface-Definition-Language (5)

- Umsetzung von IDL in Sprachkonstrukte (z.B. Java)

IDL

```
module MyModule
{
    interface MyInterface
    {
        attribute long lines;
        void printLine( in string toPrint );
    };
};
```

Java

```
package MyModule;

public interface MyInterface extends ... {
    public int lines();
    public void lines( int lines );
    public void printLine(
        java.lang.String toPrint );
    ...
};
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Interface-Definition-Language (6)

★ Vorteile

- ◆ Schnittstellenbeschreibung unabhängig von Implementierungssprache
- ◆ ermöglicht Unterstützung vieler Programmiersprachen

▲ Nachteile

- ◆ Objektschnittstelle muss in IDL und in der Implementierungssprache beschrieben werden (letzteres kann teilweise automatisiert werden)
- ◆ IDL ist sehr ausdrucksstark (z. B. **sequence**)
  - Sprachabbildung für Konstrukte, die in einer Programmiersprache nicht direkt vorhanden sind, ist kompliziert
- ◆ Fähigkeiten einer Sprache, die nicht in IDL beschrieben werden können, können nicht genutzt werden

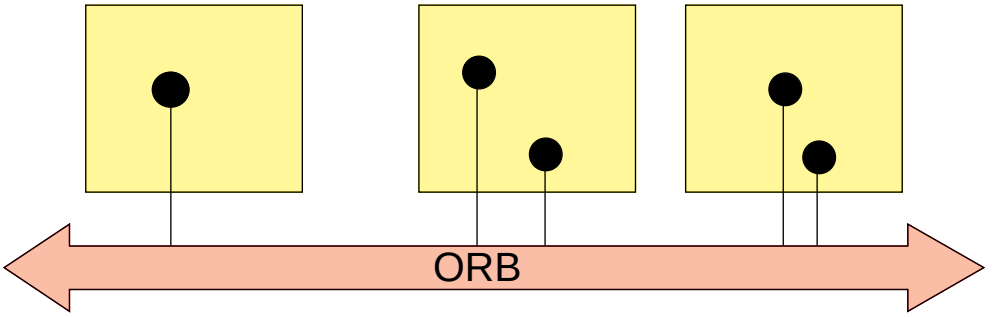
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Objekte Erzeugen und Binden

- Erzeugen eines Server-Objekts
  - Beschreibung der Objektschnittstelle in IDL
  - Programmierung des Server-Objekts in der Implementierungssprache
  - Registrierung des Objekts am ORB (bzw. dem Object Adaptor)
    - ➤ der ORB erzeugt eine "Interoperable Object Reference" (IOR)
- Binden des Clients an das Server-Objekt
  - Referenz auf Server-Objekt besorgen
    - ➤ Ergebnis einer Namensdienst-Anfrage
    - ➤ Rückgabewert eines Methodenaufrufs
    - ➤ von "ausserhalb" des Systsems: Benutzer kennt Referenz als String (IORs können in Strings konvertiert werden und umgekehrt)
  - Erzeugung des Client-Stub
  - Methodenaufruf über Client-Stub

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.4 Object Request Broker – ORB

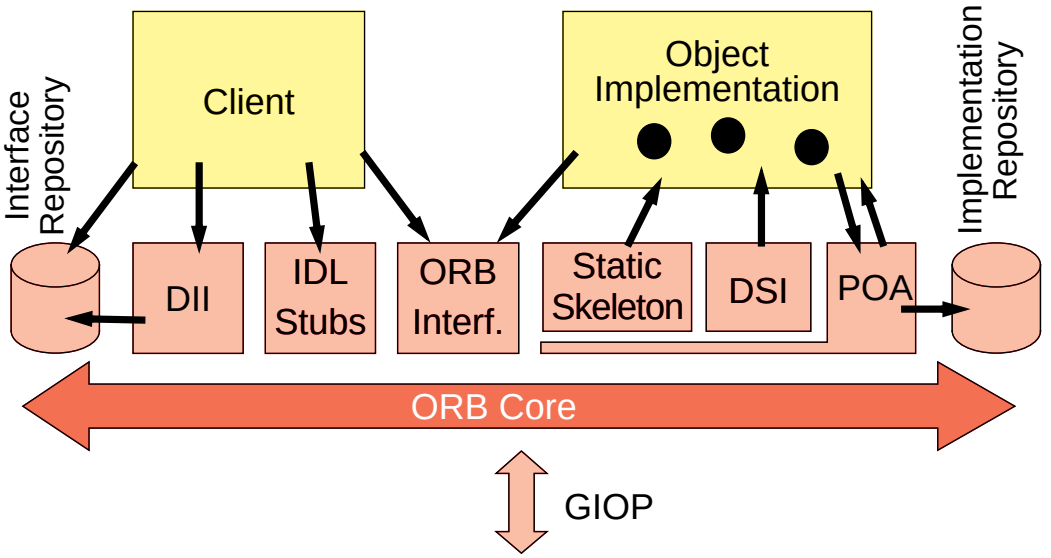


- Der ORB ist das Rückgrat einer CORBA-Implementierung
- ORB vermittelt Methodenaufrufe von einem zum anderen Objekt
  - ... innerhalb eines Adressraums / Prozesses
  - ... zwischen Adressräumen / Prozessen
  - ... zwischen Adressräumen / Prozessen von verschiedenen ORBs
- Der ORB implementiert Ortstransparenz

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Architektur

- Zentrale Komponenten eines ORB



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.