Vorlesung
Component Ware und Web-Services

- Komponentenkonzepte -

5. CORBA

Dr. Hans-Gert Gräbe, F. Schumacher
Wintersemester 2003/2004

Inhalt

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

Geschichte

Zur Geschichte der OMG (Object Management Group)

- **Ausgangspunkt 1989:** Wie kommuniziert man in einem verteilten OO-System über Sprach- und Plattformgrenzen hinweg?
  - selbst auf derselben Plattform lieferten C++-Compiler inkompatiblen Bytecode
  - verschiedene Objektmodelle in verschiedenen Programmiersprachen
  - Plattformunterschiede bei Socket-Kopplung
  - „Deep gaps everywhere"
- im April 1989 von 11 Firmen gegründet
- heute mit ca. 800 Mitgliedern eines der größten Konsortien der Computer-Industrie
  - vor allem Systemanbieter und Anwender objektorientierter Techniken

Zielstellung: „Standardisierung, koste es, was es wolle", um Interoperabilität auf allen Ebenen in einem offenen Markt für „Objekte" zu erreichen

Geschichte

Zielstellungen der OMG

- Offene Interoperabilität zwischen einer Vielzahl von Sprachen, Implementierungen und Plattformen
- mehr standardisieren als „binäre" Standards
- Flexibilität statt Binärkompatibilität
  - „teure" Hochsprachenprotokolle

- **Nichtkommerzielle Vereinigung** zur Entwicklung von technisch exakten und in der Praxis realisierbaren Spezifikationen
- **Vereinbarung von Standards und Spezifikationen** der Infrastruktur für verteilte, objektorientierte Anwendungen
- **Aufstellung von Richtlinien** (guidelines) zur Entwicklung von Umgebungen, in denen heterogene Systemen (verschiedene Plattformen, Betriebssysteme u.ä.) zusammenarbeiten können
- Durch standardisierte, objektorientierte Softwarekonzepte die **Entstehung eines Marktes für Komponentensoftware forcieren**

Geschichte

Etappen der Entwicklung von CORBA

- CORBA 1 (seit 1991) : **Standardisierung des ORB**
  - erste Lösungen, um das Wirrwar zu entflechten
  - Ansatz: Vermittlung zwischen Anfragen und Diensten durch einen Object Request Broker (ORB)
  - CORBA = Common Object Request Broker Architecture
  - **Meilenstein**: Schnittstellen-Definitionssprache (OMG IDL)

- CORBA 2 (seit 1995 – 96) : **Interoperationsstandards zwischen ORB's**
  - **Meilenstein**: Internet Inter-ORB Protokoll (IIOP)
  - muss von jeder ORB-Implementierung unterstützt werden
  - Für CORBA 2 existiert Vielzahl von Realisierungen verschiedener Anbieter und für verschiedene Plattformen

Geschichte

Etappen der Entwicklung von CORBA (Fortsetzung)

- CORBA 3 (12/2002) : **Komponenten- und Systemintegration**
  - Höhere Abstraktionsebene
  - neue Sprachebenen zur Beschreibung von Komponenten-Eigenschaften
    - persistent state definition language (PSDL)
      - Beschreibung von Aggregationen getypter Komponenten (Records)
    - component implementation definition language (CIDL)
      - Beschreibung von Elementen des Lebenszyklus von Komponenten
  - seit 1998 in der Entwicklung, aber als Ganzes erst Ende 2002 freigegeben
    - CORBA 2.3 ... 2.6 (2001) : Freigabe verschiedener Standards, auf die man sich auf dem Weg zu CORBA 3 zwischenzeitlich geeinigt hatte
  - **Meilenstein**: CORBA Komponentenmodel (CCM)
  - aktuelle Version CORBA 3.0.2
  - bisher kaum Implementierungen, die CORBA 3 voll unterstützen

Geschichte

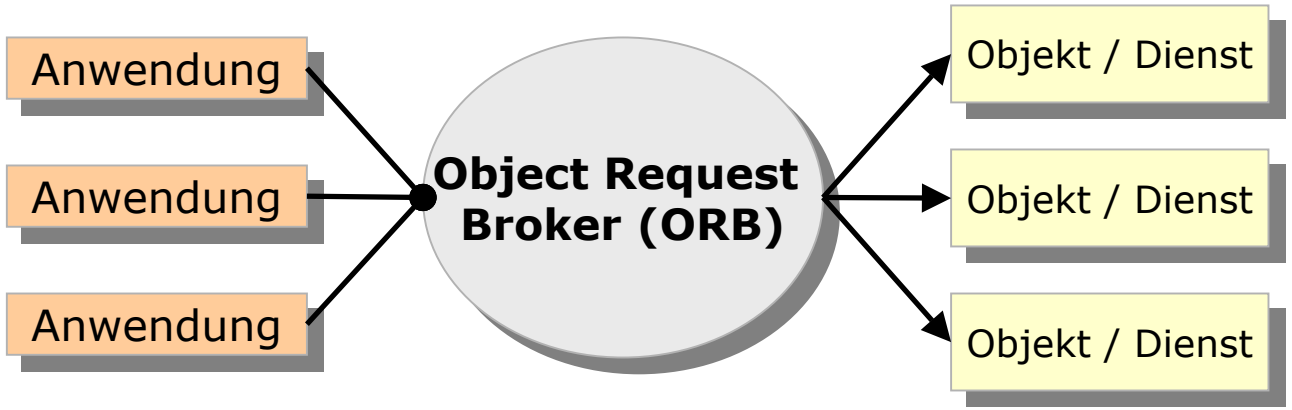
Beispiele für entwickelte Spezifikationen

- CORBA (Common Object Request Broker Architecture)
  - Spezifikation, kein Produkt
  - CORBA-Plattform für alle wichtigen Betriebssysteme
  - Hersteller unabhängige Architektur
  - Zusammenarbeit CORBA-basierter Anwendungen auf beliebigem anderen Computer, Betriebssystem, Programmiersprache, Netzwerk
- UML (Unified Modeling Language)
  - umfassende Modellierungssprache für die Softwareentwicklung
  - unterstützt Daten-, Prozess- und viele weitere Modelltypen
  - findet u.a. Anwendung in vielen CASE-Werkzeugen (Rational Rose...)
- CWM (Common Warehouse Metamodel)
- MDA (Model Driven Architecture)
- XMI (XML Metadata Interchange)

Architektur

CORBA besteht im Grunde aus drei wichtigen Teilen:

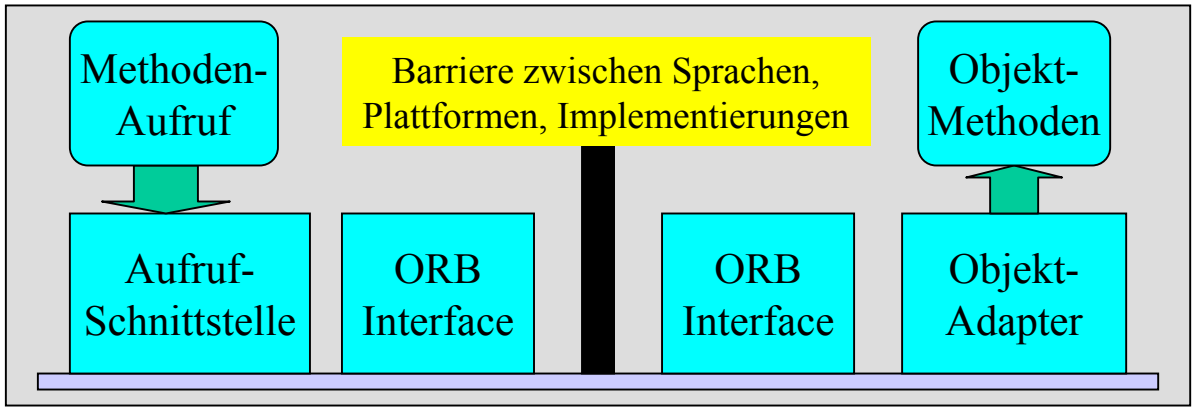
- einer Menge von Aufrufschnittstellen (Invocation Interfaces)
- dem Objekt-Anfrage-Vermittler (Object Request Broker - ORB)
- einer Menge von Objekt-Adaptern



Architektur

Laufzeitbindung von Methodenaufrufen

- Aufrufschnittstelle serialisiert Aufrufargumente
- ORB-Kern sucht Zielobjekt, -methode, organisiert Transport der Argumente
- Objektadapter deserialisiert Argumente und ruft entsprechende Methode des Zielobjekts auf.



Architektur

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 attrubute unsigned long ID;
        readonly attrubute string Name;
        readonly attrubute Date FirstContact;
        unsigned long Contacts ();
        void RegisterContact (Date dateOfContact);
        }
}
```

Architektur

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

Architektur

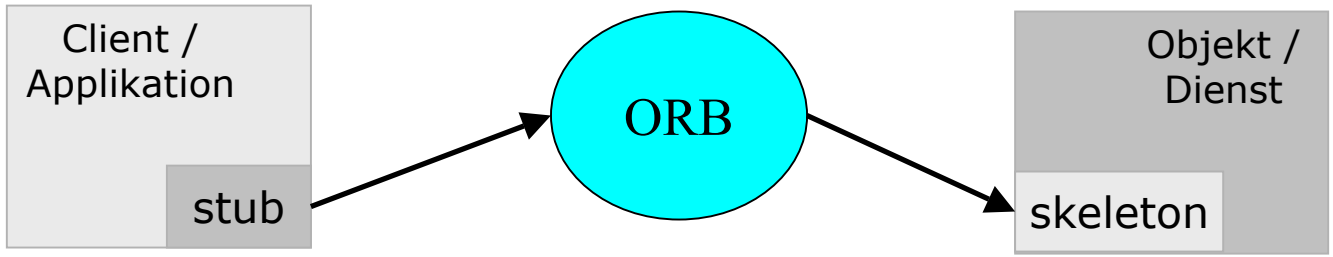
Wichtige Voraussetzungen (Fortsetzung)

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

Architektur

Stummel (stubs) und Skelette (skeletons)

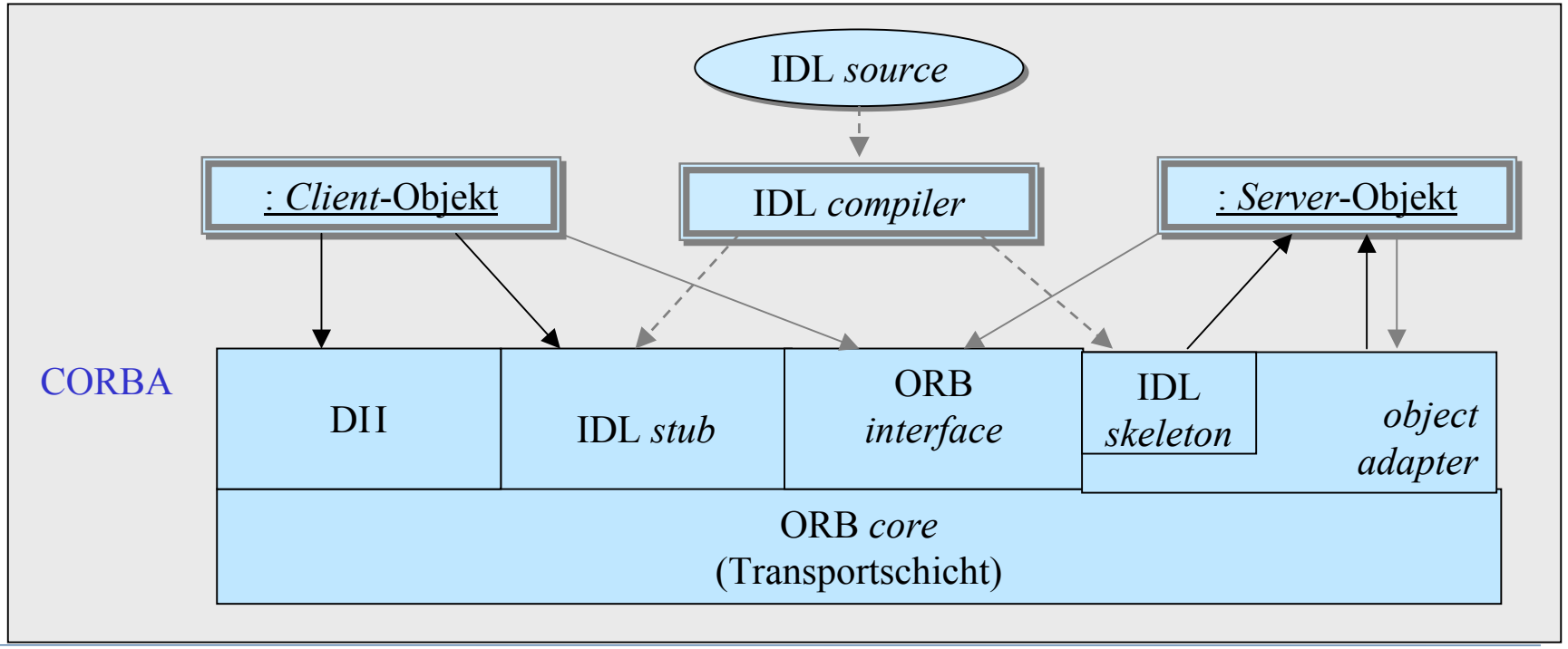
- Methodenaufrufe erfolgen über Stummel-Skelett-Prinzip des RPC
- **Stummel** sieht lokal wie ein Objekt aus
  - heißen deshalb auch (client-seitiges) Proxy-Objekt
- **Skelett** nimmt Methodenaufruf auf, deserialisiert Argumente und ruft dann die Zielmethode auf
- **Skelett** übernimmt Return-Wert, serialisiert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



Architektur

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.



Architektur

Asymmetrie 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)
    - war unterspezifiziert, deshalb Wildwuchs von Erweiterungen
    - seit 1998 überholt
  - aktuell: portable object adapter (POA)
  - 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, werden auch nicht registriert.
    - können damit auch nicht durch den ORB gestartet werden.

Object Request Broker (ORB)

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

Object Request Broker (ORB)

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

Object Request Broker (ORB)

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

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

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

Datentypen

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