Komponentenmodelle (CORBA)

Eine kurze Einführung in das Design
und die Anwendung von CORBA

Jürgen Doser

`<doser@freiburg.linux.de>`

18. Juni 1999

Entwicklung

Die anhaltende Softwarekrise führte zu einer Entwicklung der Programmierparadigmen:

	wesenliche Merkmale
Abstrakte Datentypen, Modularisierung	Datenkapselung, Abstraktion
Objektorientierung	Vererbung, Polymorphie
wiederverwendbare Komponenten	Client-Server Architektur, verteilte Systeme

Warum Komponenten ?

Komponentenmodelle haben eine Reihe von Vorteilen gegenüber klassischen Programmierparadigmen:

- **software reuse:** Komponenten sind wiederverwendbar, unabhängig von:
  - Ort,
  - Plattform,
  - Programmiersprache.
- **diststibuted computing:** Komponenten ermöglichen verteiltes Rechnen und verteilte Datenhaltung.
- **interoperability:** Komponenten verschiedener Hersteller können zusammenarbeiten, ohne den jeweiligen Source-Code zu kennen.

Welche Ansätze gibt es?

Eine Reihe von Software Herstellern und Organisationen haben Modelle entwickelt, die komponentenbasiertes Programmieren ermöglichen. Beispiele sind:

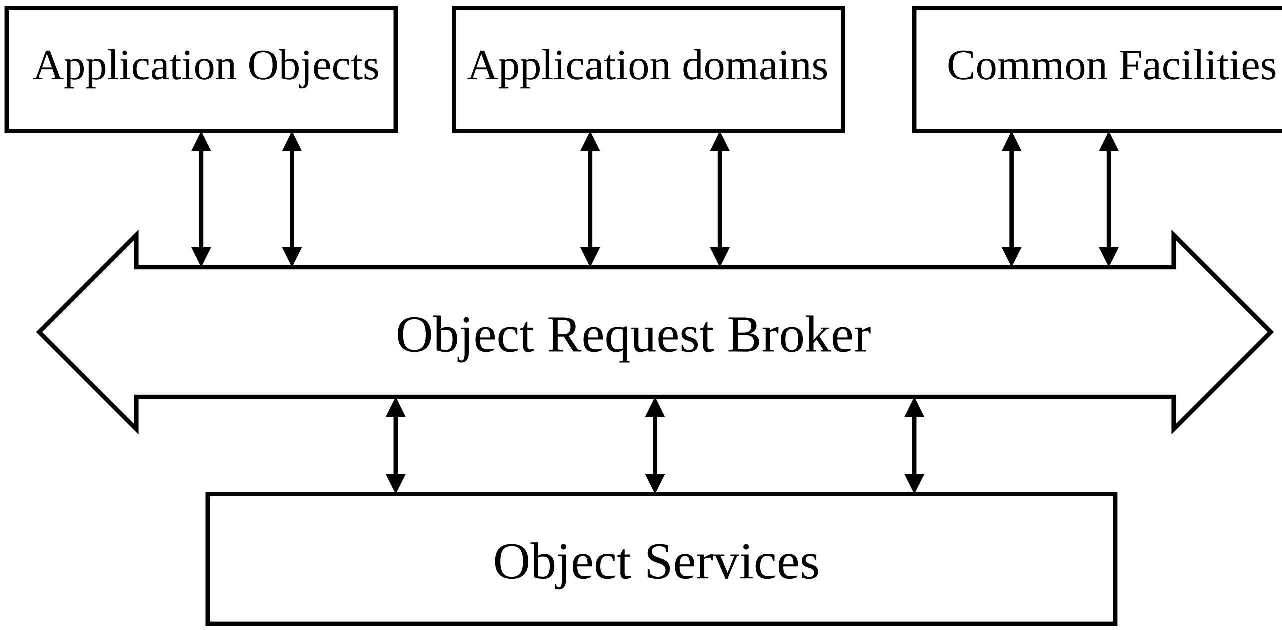
- **CORBA:** spezifiziert von der **Object Management Group (OMG)**
- **COM:** bzw. **COM+**, **DCOM**, **ActiveX**. Entwickelt von Microsoft für die Windows-Plattform.
- **OSF DCE:** das **Distributed Computing Environment** von der **Open Software Foundation**
- **RMI:** die **Remote Method Invocation** Schnittstelle in SUN's Java und darauf aufbauende **JavaBeans**.

Warum CORBA?

Diese Frage läßt sich wohl am einfachsten beantworten, indem man alle Nachteile der anderen Ansätze aufzeigt:

- **COM:** COM und seine Verwandten DCOM, ActiveX sind nur für die Windows-Plattform erhältich.
- **OSF DCE:** bietet keine Unterstützung für Objektorientierung..
- **RMI, JavaBeans:** sind spezifisch für die Programmiersprache Java.

Common Object Request Broker Architecture

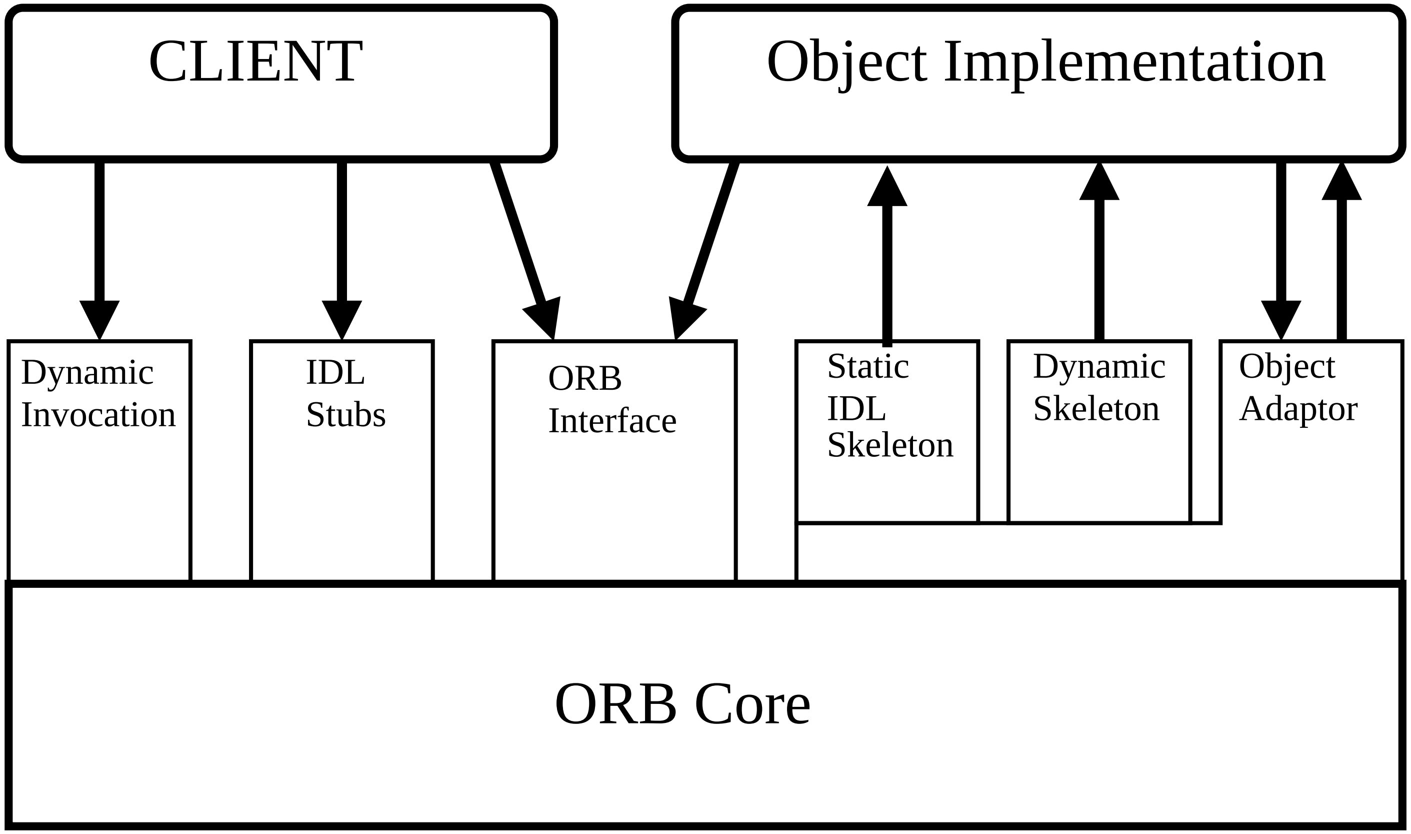


Die **Common Object Request Broker Architexture** besteht aus dem eigentlichen **ORB** und den verschiedenen Objekten, die über diesen ORB kommunizieren. Die Objekte werden dabei in verschiedene Kategorien eingeteilt.

Objekt-Kategorien

- **Object Services:** grundlegende Funktionalitäten, die praktisch jede Implementation benötigt, z.B. das Erzeugen, Löschen, Kopieren und Speichern von Objekten.
- **Common Facilities:** Funktionalitäten, die zwar von jeder Implementation benutzt werden kann, aber nicht so grundlegend sind, wie die *Object Services*, z.B. eine Drucker-Schnittstelle.
- **Application Domains:** Funktionalitäten, die speziell auf bestimmte Marktbereiche zugeschnitten sind, wie etwa Finanz- oder Transportwesen.
- **Application Objects:** Komponenten, die Softwarehersteller zu Verfügung stellen.

Die Object Management Architecture



IDL Stubs

- Schnittstellen werden in der **Interface Definition Language** spezifiziert..
- Syntax ist nah an C++ angelehnt.
- IDL-Dateien werden kompiliert zu Headerfiles für Clients und zu Implementationsgerüsten für Implementationen.
- Benutzer dieser Komponenten können dann mit diesen umgehen, wie mit normalen Objekten (nur die Objekterzeugung geschieht anders).
- Das Mapping von IDL zu Sprachen wie C,C++ oder Java ist spezifiziert.

Beispiel IDL-File

```
interface Logger{

    enum Log_Priority{
        LOG_DEBUG, LOG_ERROR
    };

    exception Disconnected{};

    struct Log_Record{
        Log_Priority Type;
        long host_addr;
        long time;
        sequence<char> msg_data;
    };

    void log(in Log_Record log_rec)
        raises (Disconnected);

    attribute boolean verbose;
};
```

Beispiel Client

```
int main(){

    LoggerRef logger;
    Log_Record log_rec;

    logger = Logger::_bind();

    log_rec.type = LOG_DEBUG;
    log_rec.time = //...
    //...

    try{ logger->log(log_rec); }
    catch(Logger::Disconnected){
        cerr << "logger disconnected"
             << endl;
    }

    return 0;
}
```

ORB Interface

- spezifiziert Funktionen, die nicht von Komponenten ausgeführt werden, sondern vom ORB selbst.
- dazu gehören Funktionen wie:

```
string object_to_string(in Object obj);
Object string_to_object(in string str);
ImplementationDef get_implementation();
InterfaceDef get_interface();
boolean is_nil();
boolean is_equivalent(in Object other_obj);
```

- Schnittstelle ist unabhängig vom ORB und vom Object Adaptor.

Object Adaptor

- Zuständig für Dinge wie Erzeugen von Objektreferenzen, Aktivierung von Objekten, Registrierung von Implementationen.
- Jeder Methoden-Aufruf durchläuft zuerst den Objekt Adapter, der (falls nötig) entsprechende Objekte erzeugt, die diese Methode zur Verfügung stellen und schliesslich dafür sorgt, daß die richtige Implementation diese Methode ausführt.
- Er kann auch plattformübergreifend Dienste anbieten, die manche ORB-Implementationen oder Betriebssysteme nicht anbieten.
- Um Flexibilität und Performance für spezielle Aufgaben zu wahren, gibt es verschiedene Arten von Objekt Adaptern

Beispiele für Objekt Adapter

Basic Object Adaptor: Der "Standard"-Adapter. Wird üblicherweise benutzt, falls Komponenten als eigene Programme vorliegen (Client-Server). Kümmert sich auch um den Start des Servers, falls nötig.

Library Object Adaptor: Wird üblicherweise benutzt, falls die Implementationen als dynamische Bibliothek vorliegen. Braucht sich also unter anderem nicht um Objektaktivierung zu kümmern.

Object Oriented Database Adapter: Stellt eine direkte Verbindung zu einer objekt-orientierten Datenbank her.

KOM/OpenParts

- Das **KDE Objekt Model** benutzt CORBA um Eventhandling und den Signal-Slot Mechanismus von qt auf verteilte Objekte zu erweitern.
- Jedes KOM-Objekt ist in der Lage beliebige Events zu empfangen und zu filtern.
- Kommunikation geschieht dadurch, daß eine Applikation Events verschickt (auch an sich selbst), oder eben ein Signal gesendet wird.
- **OpenParts** baut auf KOM auf und stellt embedded controls in der GUI zur Verfügung.

Bibliographie

- OMG Homepage:
  `<http://www.omg.org/>`
- OMG CORBA-Seite:
  `<http://www.omg.org/CORBA/>`
- CORBA Success Stories:
  `<http://www.corba.org/>`
- CORBA-FAQ:
  `<http://www.cerfnet.com/~mpcline/corba-faq/>`
- Free CORBA Page:
  `<http://adams.patriot.net/%7etvalesky/freecorba.html>`
- gute Linksammlung:
  `<http://pent21.infosys.tuwien.ac.at/Research/Corba/>`