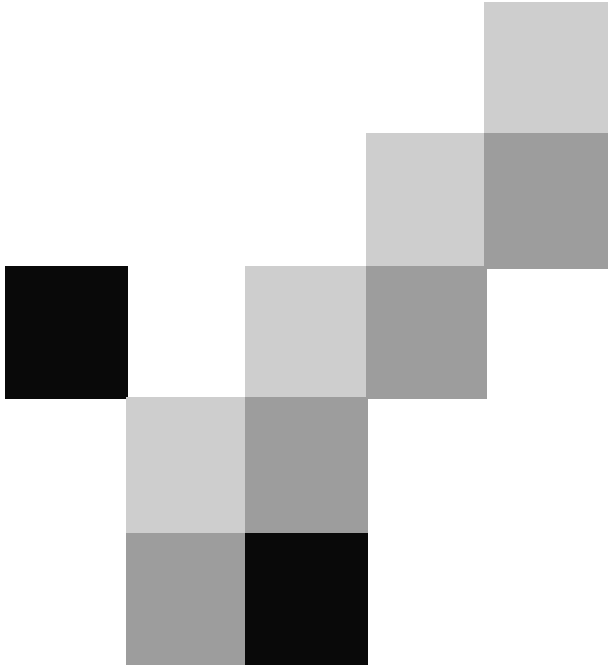
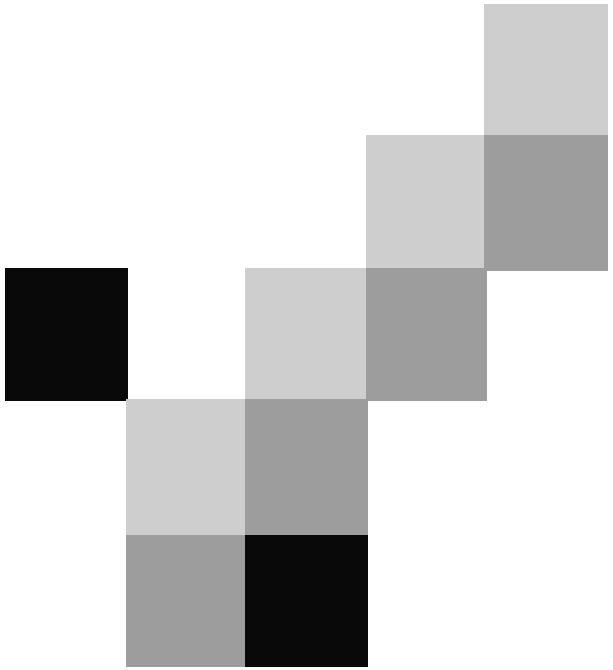


Seminar

Ausgewählte Komponenten von Betriebssystemen

IDL4 Compiler



IDL4 Compiler

Hristo Pentchev

Überblick

- CORBA
- IDL Allgemein
- IDL4 Compiler
- Beispiele

CORBA

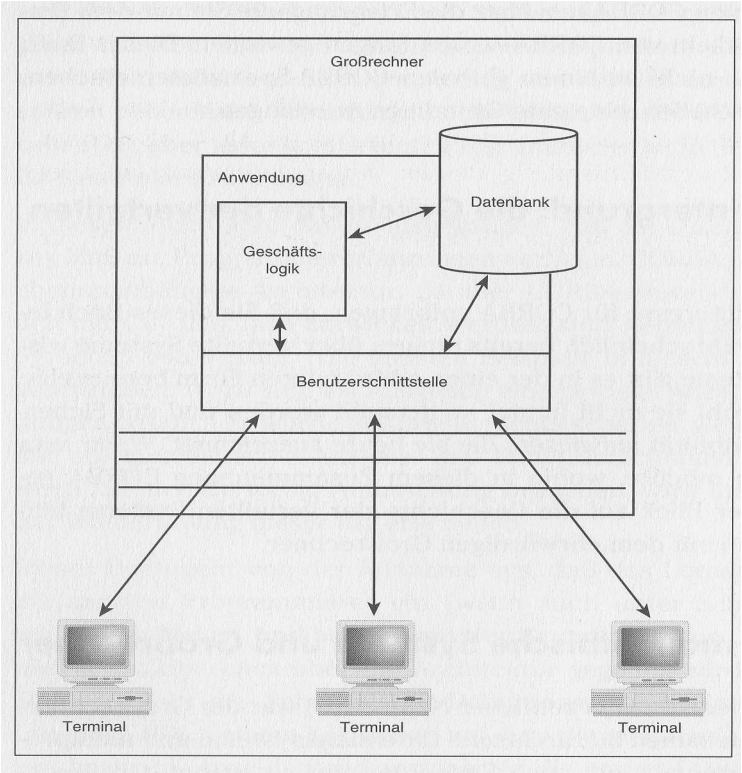
Common
Objekt
Request
Broker
Architecture

Gemeinsame
Architektur
für
Objektanforderungs-
vermittler

Entwickelt von OMG in
1991

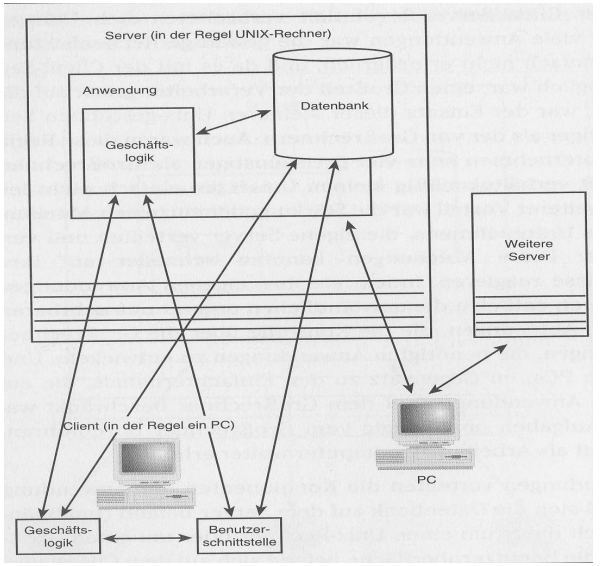
CORBA

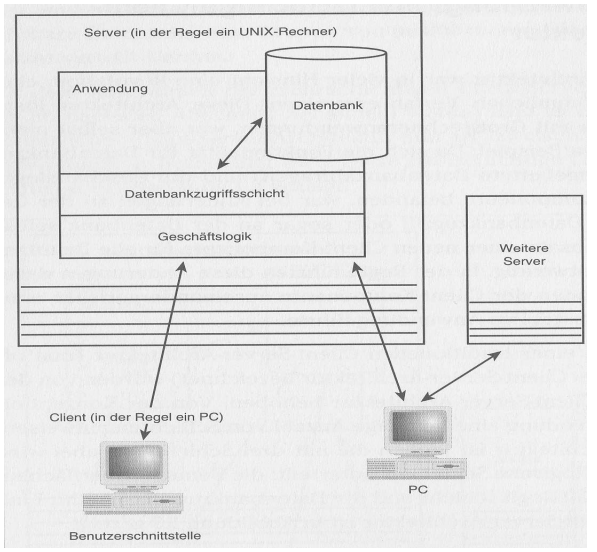
- Historische Entwicklung
  - Monolithische Systeme und Großrechner
    - Sehr teuer in der Wartung
    - Große Anzahl von Benutzer
    - Wurden Zentral verwaltet
    - Monolithische Architektur



CORBA

- Client-Server Architektur
  - Großrechner waren unnötig Leistungsfähig für viele Anwendungen
  - Übertragung von Verarbeitungslast auf die PCs
  - Preisgünstig
  - Unabhängiger und profilierter
  - Datenbank auf dem Server (Unix-Rechner oder Großrechner)
  - Benutzeroberfläche auf dem Client





CORBA

- Verteilte Systeme
  - Absolute Flexibilität durch
    - Konstante Komponentenschnittstellen
- Die Rolle von CORBA
- Sprachenunabhängige Architektur (hauptsächlich in C++ und Java)
- Plattformunabhängig

CORBA

- Alternativen zu CORBA
  - Die Socket-Programmierung

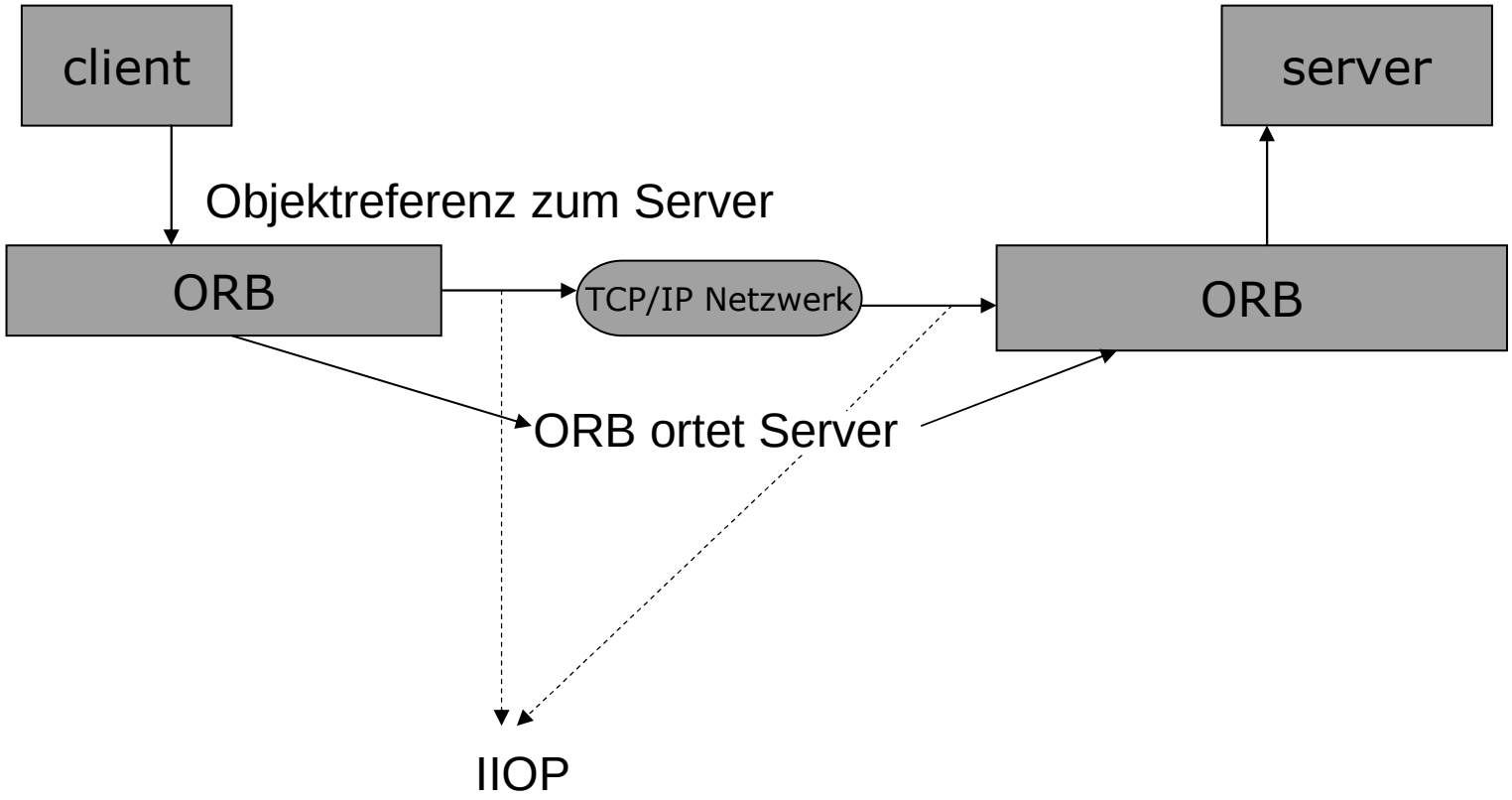
    Kommunikation zwischen Rechnern und Prozessen über sockets
  - Remote-Prozedurenaufruf (RPC)

    Funktionsorientierte Schnittstelle zur Kommunikation auf Socket-Ebene
  - Distributed Computing Environment(DCE,verteilte Rechnerumgebung) von Open Software Fondation(OSF)
  - Distributed Component Object Model (DCOM) von Microsoft
  - Remote Methode Invocation(RMI) in Java Standard

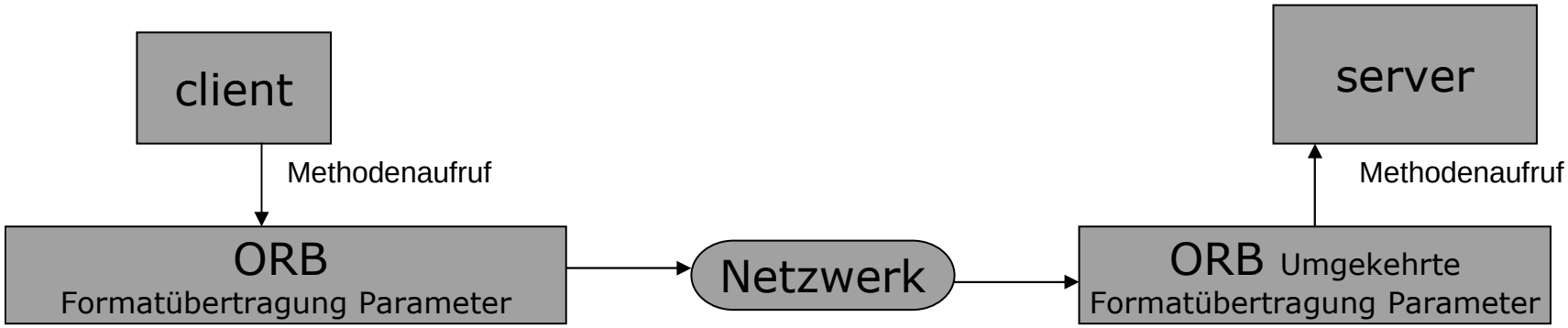
CORBA

- Die CORBA Architektur
  - Object Request Broker(Objektanforderunsvermittler)
    - Eine Software Komponente, die die Kommunikation zwischen Objekten erleichtern soll.
  - Interface Definition Language
  - Kommunikationsmodell
    - Interoperable Object References (IOR) für Zugriff auf CORBA-Objekte
    - Internet Inter-ORB Protokoll (IIOP)- Kommunikation zwischen ORBs

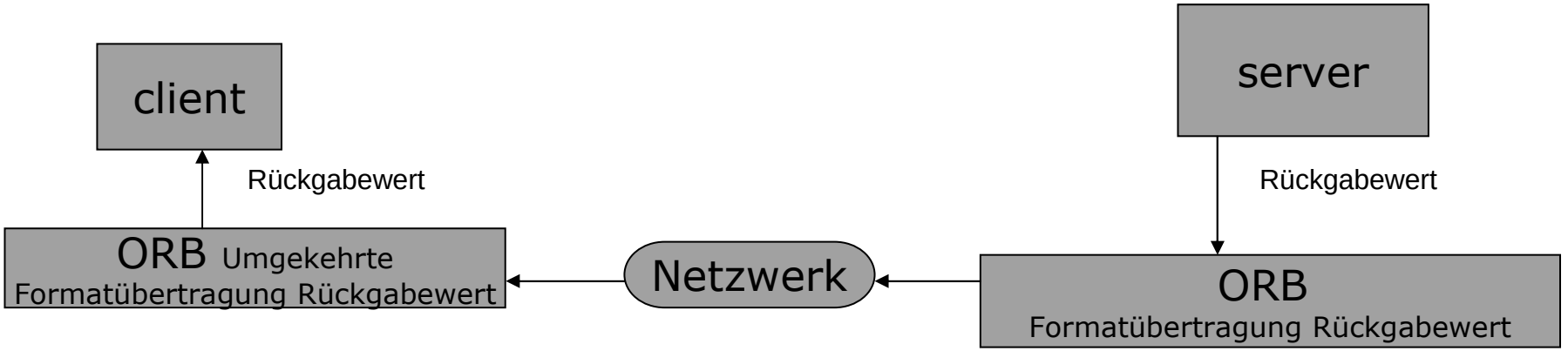
ORB-Auflösung von Objektanfragen



ORB-Auflösung von Objektanfragen

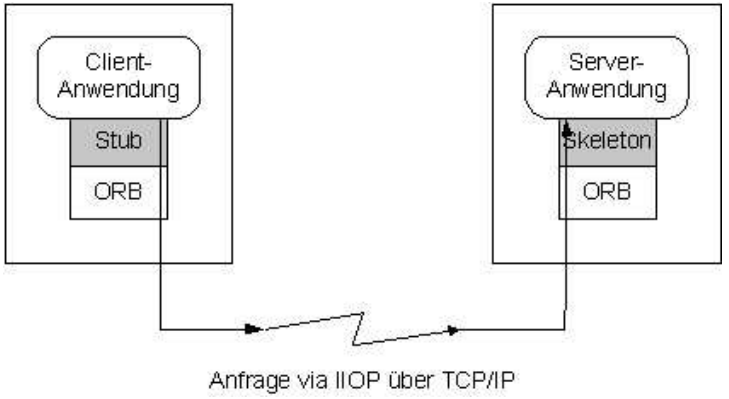


ORB-Auflösung von Objektanfragen



CORBA

- Objektmodell
  - Pass by Reference
- Clients und Server
- Stubs und Skeletons – werden beim Kompilieren von Schnittstellendefinitionen generiert



IDL

Interface

Definition

Language

Schnittstellendefinitionssprache

IDL

- Sprachenunabhängig durch Sprachenabbildung (Language Mapping)
- Grundregeln
  - Definitionssyntax
    - ;
    - {}
  - Kommentare: // oder /*…*/
  - C - Präprozessor

IDL

- Einfache Typen
  - void
  - boolean
  - char(8Bit), wchar(normal 16Bit)
  - string
- Gleitkommatypen
  - float:IEEE-Gleitkommawert mit einfacher Genauigkeit
  - double: doppelte Genauigkeit
  - long double: erweiterte doppelte Genauigkeit

IDL

- Integer-Typen
  - short(16Bit)
  - long(32 Bit)
  - long long(64 Bit)
  - unsigned short
  - unsigned long
  - unsigned long long
  - octet(8 Bit)

IDL

- Zusammengesetze Typen
  - Enum
  - Struct
  - Union

```
enum WochenTage{ Sonntag,
   Montag};

struct DatumStruktur{ short
   Jahr; short Monat;
   WochenTage Tag;};

union Multi switch(long){
   case 1:
        short s,
   case 2:
        double d,
   case 3:
   default:
        string str;
};
```

IDL

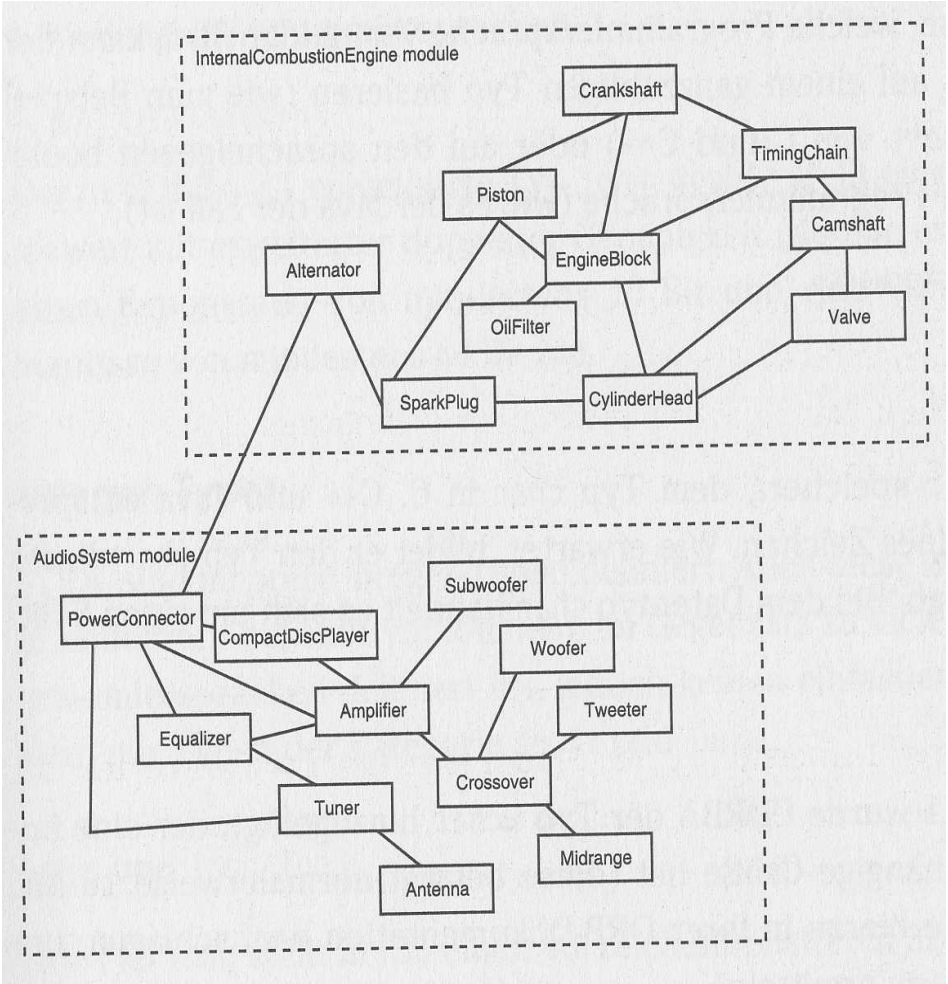
- Const
- Typedef
- Container-typen
  - Sequence
  - Array
- Exceptions
  - Standard-Exceptions
  - Userdefined
- Der Typ any

```
const float pi=3.14;

typedef boolean in_der_liste;


typedef sequence<long> Bestand;
typedef sequence<long,3>
   DreiWerte;

typedef string WochenTag[3];


exception unbekanterTyp{string
   meldung};
void foo(in any x) raises
   (unbekanterTyp);
```

IDL

Sprachkonstrukte

- Module
  - Kopplung
  - Bindung



IDL

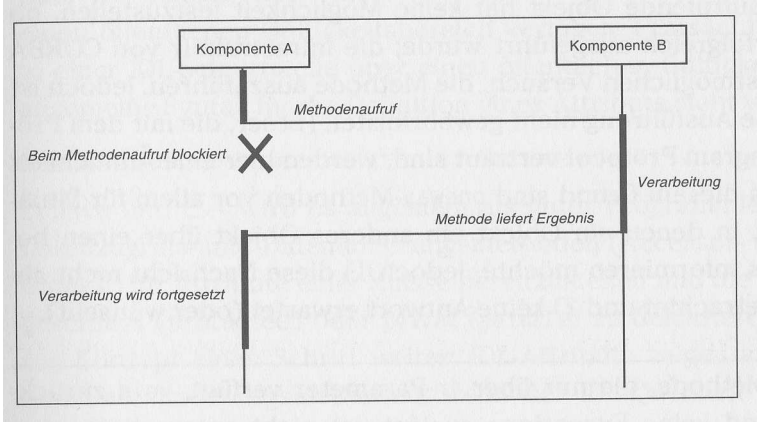
- interface
  - Methoden
    - Blockieren
    - oneway :void, in, no exceptions
  - Parameter
    - in
    - out
    - inout

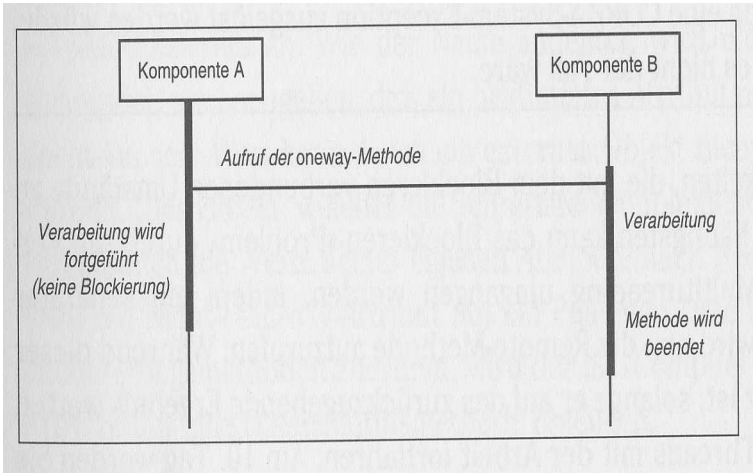
```
interface bank {
        void deposit(in string owner, in long
    account_no, in float amount);
        float balance(in long account_no);
      };
```

  - Attribute

```
attribute short s; ->  short meinWert;
                       void meinWert( short wert);
```

  - Vererbung von Schnittstellen





IDL4

- IDL4
  - Ist ein stub-code Generator
  - Unterstütz 2 Sprachen
    - CORBA IDL
    - DCE IDL
  - Unterstütz die folgende Typen
    - einfache

short	signed 16 bit integer	supported
long	signed 32 bit integer	supported
long long	signed 64 bit integer	supported
unsigned short	unsigned 16 bit integer	supported
unsigned long	unsigned 32 bit integer	supported
unsigned long long	unsigned 64 bit integer	supported
float	floating point (32 bits)	supported
double	floating point (64 bits)	supported
long double	floating point (80 bits)	supported
char	8 bit character	supported
wchar	16 bit character	supported
boolean	boolean value	supported
any	wildcard type	-
octet	unsigned 8 bit integer	supported
enum	enumeration	supported
ref	pointer	supported
fpage	memory flexpage	supported
fixed	fixed-point decimal	-

IDL4

- Zusammengesetze

string	unbounded string	supported
string<n>	bounded string	supported
wstring	unbounded wide string	supported
sequence<type>	type sequence	supported
sequence<type,n>	bounded type sequence	supported
struct	structure	flat members supported
union	union	DCE style supported
Object	object reference	supported
array	array	supported
typedef	alias type	supported
exception	exception	memberless exceptions supported

IDL4

- Generiert
  - Client stubs: -c
  - Server stubs: -s
  - Server templates: -t
- Wird wie folgt benutzt:

  idl4 [Optionen] schnittstelledefinition.idl

IDL4

Allgemeine Optionen:

- -c
- -s
- -t

- -o filename.c
- -h filename.h
- -v

Warnungsoptionen:

- -Wprealloc
- -Wall

Test.idl

```
module storage{
    interface textfile{
      void readln(inout short pos, out string line);
    };
  };
```

Test_client.h

…

```
static inline void
    storage_textfile_readln(storage_textfile _service,
    CORBA_short *pos, CORBA_char **line,
    CORBA_Environment *_env)

    {
```

…

Test_template.c

```
#include "test3_template_server.h"
...
IDL4_INLINE void storage_textfile_readln_implementation(CORBA_Object _caller, CORBA_short *pos, CORBA_char
      **line, idl4_server_environment *_env)

      {
        /* implementation of storage::textfile::readln */

        return;
      }

...
void storage_textfile_server(void)

      {
      …
while (1)
        {
          buffer.size_dope=STORAGE_TEXTFILE_RCV_DOPE;
          buffer.rcv_window = idl4_nilpage;
          partner = idl4_nilthread;
          reply = idl4_nil;
          w0 = w1 = w2 = 0;

          while (1)
            {
              idl4_reply_and_wait(reply, buffer, partner, msgdope, fnr, w0, w1, w2, dummy);

              if (msgdope & 0xF0)
                break;

              idl4_process_request(storage_textfile_vtable, fnr & STORAGE_TEXTFILE_FID_MASK, reply, buffer,
      partner, w0, w1, w2, dummy);
            }

          enter_kdebug("message error");
        }
      }
      ...
```

Client.c

```
#include<test_client.h>
void testen(l4_threadid_t server) {
    CORBA_Environment env =idl4_default_environment;
    short pos = 100;
    char *buf;

    storage_textfile_readln(server, &pos, &buf, &env);

    switch (env._major) {
        case CORBA_SYSTEM_EXCEPTION:
            printf("IPC failed, code %d\n",
            CORBA_exception_id(&env));
            CORBA_exception_free(&env);
            return -1;
        case CORBA_USER_EXCEPTION:
            printf("User-defined exception");
            CORBA_exception_free(&env);
            return -1;
        case CORBA_NO_EXCEPTION:
            break;
    }

...
```