Software Reuse

6. CORBA und CCM

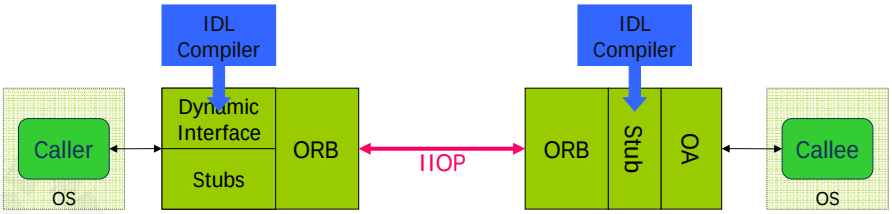
Peter Sturm
Universität Trier

Common Object Request Broker Architecture (CORBA)

- Standard der Object Management Group (OMG)
  - 1991 CORBA 1.1
  - 1994 CORBA 2.0
  - seit ca. 2001 CORBA 3.0
- OMG
  - Herstellerübergreifendes Konsortium
  - Gegründet 1989 von 11 Mitgliedern
  - Aktuell mehr als 600 Mitglieder
- Ziele
  - Verteilte Anwendungen mit objektorientierten Methoden
  - Objektorientierte Softwarebausteine (Komponenten)
  - Schnittstellenstandards für verteilte Büroanwendungen

Aufbau

- Eigene CORBA-IDL
  - Language-Bindings für alle gängigen Sprachen
    - Manche Lösungen diskussionswürdig
- ORB = Object Request Broker
  - „Open Source“-Lösungen vorhanden
  - Beispiele: MICO (Uni Frankfurt), ORBit in Gnome



IDL

- CORBA-IDL mit Abstand mächtigste Schnittstellenbeschreibung
- Basistypen
  - Zahlen, Strings, …
- Benutzerdefinierte Typen
  - Strukturen, Arrays, …
- Interfaces und Operationen
- Module
- …

Basistypen

Typ	Werteraum	Größe
short	-215 bis 215-1	≥ 16 Bit
long	-231 bis 231-1	≥ 32 Bit
unsigned short	0 bis 216-1	≥ 16 Bit
unsigned long	0 bis 232-1	≥ 32 Bit
float	IEEE single precision	≥ 32 Bit
double	IEEE double precision	≥ 64 Bit
char	ISO Latin-1	≥ 8 Bit
string	ISO Latin-1 (kein NUL)	Variabel
boolean	TRUE oder FALSE	./.
octet	0-255	≥ 8 Bit
any	Zur Laufzeit identifizierbarer Typ	Variabel

Benutzerdefinierte Typen

- Konstanten
- Eigene Namen
  - `typedef short Year;`
- Aufzählungen
  - `enum Farbe { rot, gelb, gruen };`
  - Zuweisung bestimmter Zahlenwerte nicht erlaubt
- Strukturen (wie überall)
- Unions
  -
```
union Mitarbeiter switch (Hackordnung) {
case Professor:
    unsigned short mtbn;
case Mitarbeiter:
    unsigend long sws;
};
```

Benutzerdefinierte Typen (contd.)

- Arrays
  - `typedef Pixel Farbe[3];`
  - Typedef muß deklariert werden
- Sequenzen
  - Variabel-langer Vektor
  - Unbegrenzt: `typedef sequence<Pixel> Zeile;`
  - Begrenzt: `typedef sequence<Pixel,100> Zeile;`
  - Sequenzen von Sequenzen sind möglich
- Rekursive Datenstrukturen

```
struct Knoten {
    long wert;
    sequence<Knoten> kinder;
}
```


Interface

- CORBA-Beschreibung einer Schnittstelle
- Bestandteile
  - Konstanten und Typdefinitionen
  - Exceptions
  - Attribute (Data Members)
  - Operationen (Function Members)
- Keine geschachtelte Interfaces möglich
- Directional Attribute
  - in
    - Parameter wird vom Client zum Server gesendet
  - out
    - Parameter wird vom Server zum Client gesendet
  - inout
    - Paramter wird vom Client zum Server gesendet. Der resultierende Wert wird nach Beendigung der Operation an den aufrufenden Client zurückgesendet

Operationen

- Operationsdefinition
  - Ergebnistyp
  - Operationsname
  - 0 oder mehr Parameterdeklarationen
- Overloading nicht erlaubt (keine gleichen Operationsnamen)
- Keine anonymen Typen als Parameter oder Ergebnis:
  - `sequence<long> get_longs();` // Fehler ☺
  - `typedef long Zahl;`
    `sequence<Zahl> get_zahlen();` // Erlaubt
- Einwegoperationen
  - typedef sequence<octet> bytestream;
    oneway void send (in bytestream data);
  - Keine out und inout Parameter, void Return

Exceptions

- Modellierung von Fehlern
- Beispiel
  - `exception Failed {};`
  - ```
    exception RangeError {
        unsigned long supplied_val;
        unsigned long min_permitted_val;
        unisgned long max_permitted_val;
    }
    ```
- Operationen verwenden raises-Ausdruck:
  - ```
    void set_value ( in unsigned long v )
        raises( RangeError, Failed);
    ```
- Keine Ableitung bei Exceptions erlaubt

Attribute

- Beispiel
  - ```
    interface Thermostat {
        readonly attribute short istwert;
        attribute short sollwert;
    };
    ```
- Attribute sind keine Data Member!
- Obiges ist äquivalent zu
  - ```
    interface Thermostat {
        short get_istwert ();
        short get_sollwert ();
        void set_sollwert ( in short t );
    }
    ```


Module

- Aufbau von Namensräumen
  - ```
    module X {
        typedef short t;
        …
    };
    ```
  - ```
    module Y {
        X::t v;
        …
    };
    ```
- Eigenschaften
  - Module gleichen Namens können wieder geöffnet werden

Inheritance

- Interface-Inheritance
  -
```
interface Base {
    …
};
```
  -
```
interface Child : Base {
    …
};
```
- Inkl. Polymorphie wie in C++ u.ä.
- Ggf. implizite Ableitung vom Basistyp Object
- Leere Interfaces sind erlaubt
- Abgeleitete Interfaces können Typen, Konstanten und Exceptions redefinieren
- Kein Overloading
- Multiple Inheritance erlaubt (ohne Mehrdeutigkeiten)

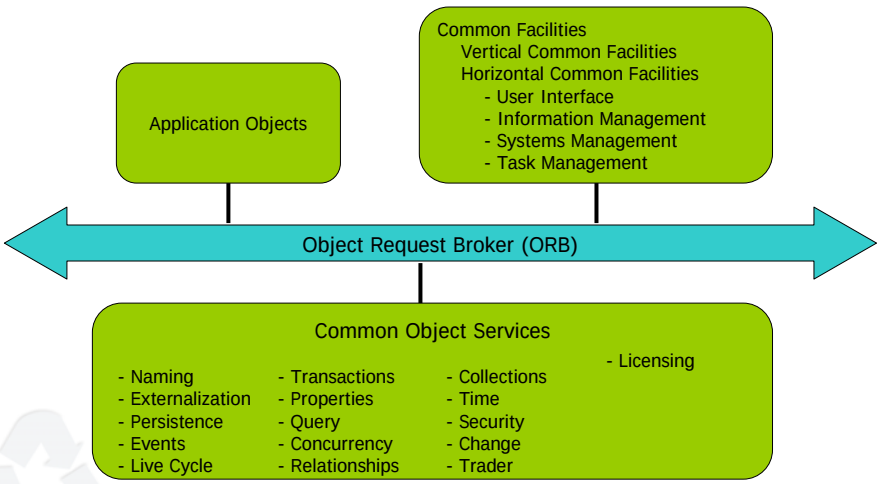
IDL ⇒ C++ Mapping

- Vergleichsweise direkt, da C++ mächtiger ist
  - Basic Types, Enums, Konstanten gleich (ggf. Namensergänzung)
  - Strings werden char *
  - Wrapperklassen z.B. für Strings (String_var)
  - Aus Sequences werden spezifische Vektorklassen
  - Module werden Namespaces
- Interface = abstrakte Klassen
  -
```
interface IF {
    long f ();
}
```
  -
```
class IF : public virtual CORBA::Object {
public:
    virtual CORBA::Long f () = 0;
    …
}
```

IDL ⇒ Java

- Grundfunktionalität über Package org.omg.*
  - siehe auch C:\java\j2sdk1.4.1_01\docs\index.html
- Ansonsten Generierung der Java-Mappings vergleichbar C++
- Aus einem IDL-Interface X wird
  - Java-Interface X für Client (Signature Interface)
  - Java-Interface X_Operations für Server (Operations Interface)

Object Management Architecture (OMA)



Common Object Service Specification (COSS)

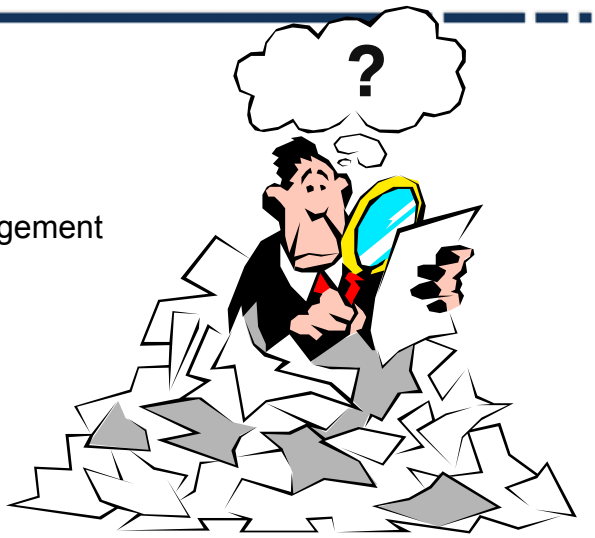
- Gültig für alle CORBA-konformen Plattformen
- Naming
- Externalisation
  - Export von Zuständen in „flache“ Dateien
- Persistenz
  - Dauerhafte Objektzustände
- Events
  - Weiterleitung asynchroner Ereignisse
  - Einrichtung von „Event Channels“
- Lifecycle
  - Unterstützung des Objekt-Lebenszyklus
  - Funktionen: Erzeugen, Löschen, Kopieren, Verlagern, ...

COSS contd.

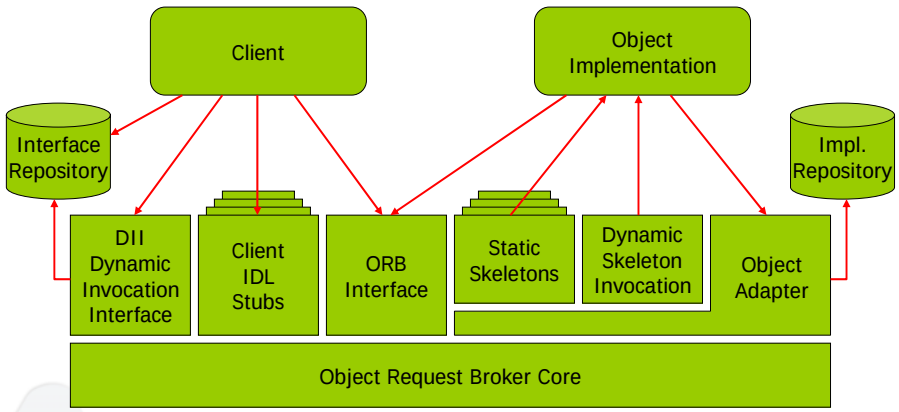
- Transactions
  - 2 Phase-Commit-Protokoll
  - Flache und geschachtelte Transaktionen
- Properties
  - Attribut/Wert-Paare für Objekte speichern und abfragen
- Query
  - SQL-Abfragen
- Concurrency
  - Verwaltung und Realisierung von Locks
  - Nutzung u.a. bei Transaction Services
- Relationships
  - Gruppierung von Objekten
  - Funktionen: Erzeugen, Traversieren, ...

COSS contd. (2)

- Collections
- Time
- Security
- Change Management
- Trader
- Licensing



CORBA ORB

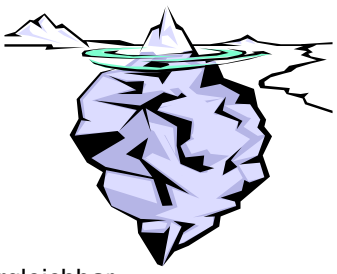


Methodenaufrufe

- Schnittstellenspezifikation über IDL
  - Objektorientiert
  - Sprachbindungen für C, C++, Smalltalk, Java, Cobol, ...
- Aufruf
  - Zielobjekt
  - Aufrufparameter
  - Eventuell Rückgabewerte, Exceptions
- Aufrufsemantik
  - Synchroner Auftrag (vgl. RPC)
  - Asynchroner Auftrag
  - Asynchrone Meldung
- Klassisch statische Schnittstellen (Stubs)
- Dynamische Schnittstellen

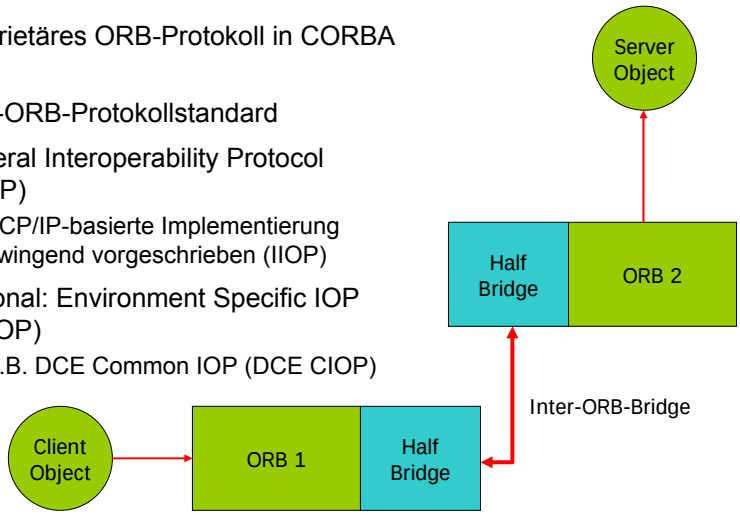
Object Adapter

- Steuert Funktionen des Server-Objekts
  - Aktivierung des Objekts bei eingehendem Request
  - Authentifizierung des Aufrufers
  - Zuordnung Objektreferenzen zu Instanzen
  - Registrierung des Server-Objekts
  - Lebensdauer des Server-Objekts
- Basic Object Adapter (BOA)
  - Standardadapter
- Extremfall
  - ORB und BOA Laufzeitbibliothek in der Anwendung
  - Leistung mit Unterprogrammaufruf vergleichbar
  - Neue Basisarchitektur für Betriebssysteme?



ORB-2-ORB-Kommunikation

- Proprietäres ORB-Protokoll in CORBA 1.x
- Inter-ORB-Protokollstandard
- General Interoperability Protocol (GIOP)
  - TCP/IP-basierte Implementierung zwingend vorgeschrieben (IIOP)
- Optional: Environment Specific IOP (ESIOP)
  - Z.B. DCE Common IOP (DCE CIOP)

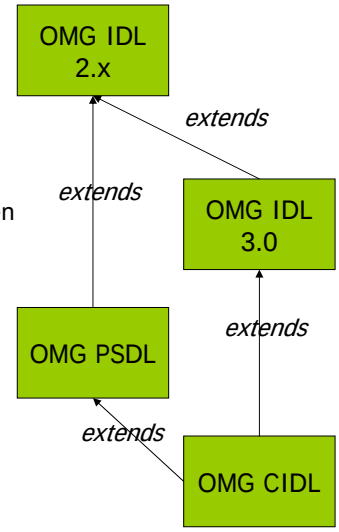


CCM

- CORBA Component Model
- Bestandteil der IDL (CIDL)
  - basic component (= EJB)
  - extended component
- Sogenannte Ports definieren Interaktionsmöglichkeiten
  - Facets: Komponentenfunktionalität
  - Receptables: Übernahme von Objektreferenzen
  - Event Sources
  - Event Sinks
- Komponenten haben ein Home
  - … und überhaupt sieht alles irgendwie nach EJB aus ☺
- CCM in der Praxis nicht weit verbreitet

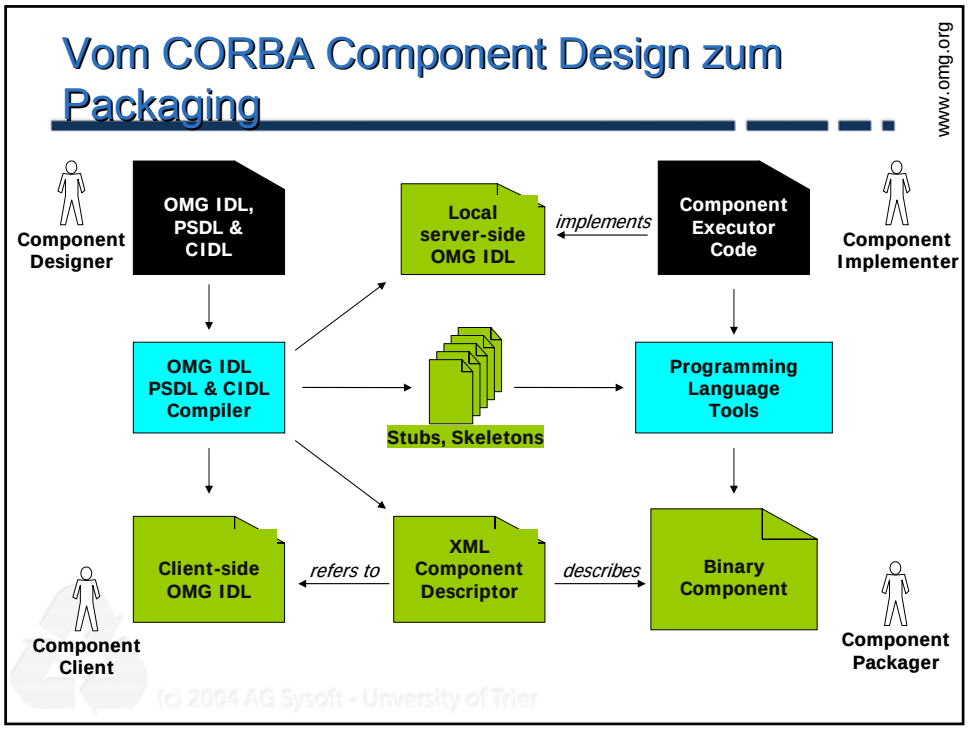
Relationen zwischen OMG Definition Languages

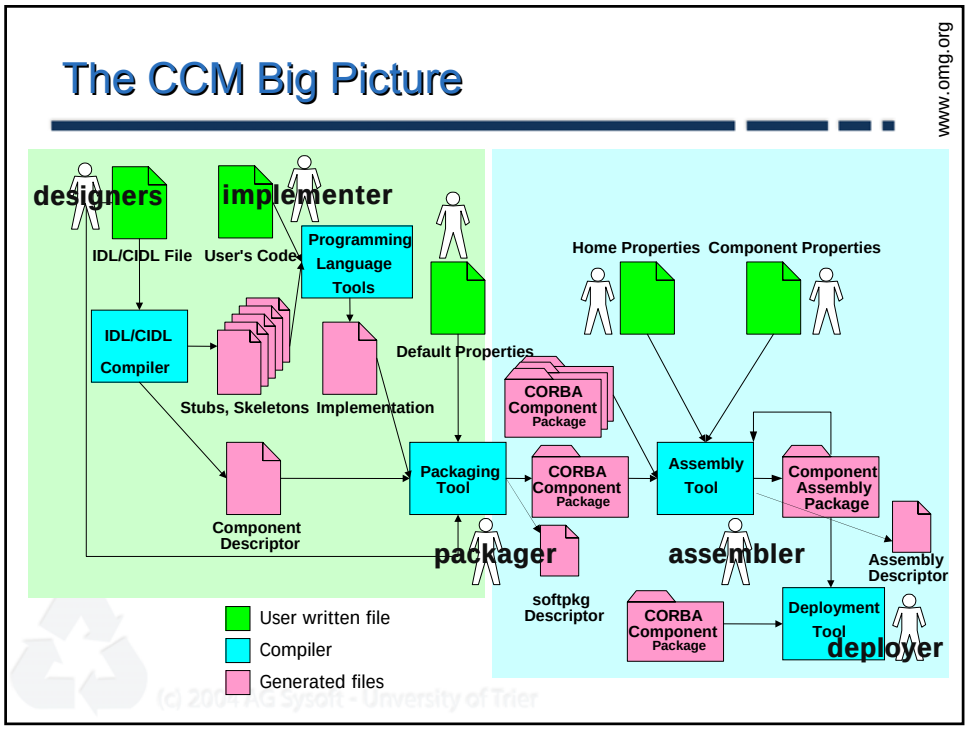
- OMG IDL 2.x
  - Objektorientierte Interaktion
  - Datentypen, Interfaces und Werttypen
- OMG IDL 3.0
  - Komponentenorientierte Interaktion
  - Component types, Homes und Eventtypen
- OMG PSDL
  - Persistent state definition
  - [Abstract] storage types and homes
- OMG CIDL
  - Component implementation description
  - Compositions and segments



CCM User Roles

- Component designers
- Component clients
- Composition designers
  (~ component implementation designers)
- Component implementers
- Component packagers
- Component deployers
- Component end-users


Vom CORBA Component Design zum Packaging
www.omg.org
Component Designer
OMG IDL, PSDL & CIDL
Local server-side OMG IDL
implements
Component Executor Code
Component Implementer
OMG IDL PSDL & CIDL Compiler
Stubs, Skeletons
Programming Language Tools
Client-side OMG IDL
refers to
XML Component Descriptor
describes
Binary Component
Component Client
Component Packager
(c) 2004 AG Sysoft - University of Trier


The CCM Big Picture
www.omg.org
designers
implementer
IDL/CIDL File
User's Code
Programming Language Tools
IDL/CIDL Compiler
Stubs, Skeletons
Implementation
Default Properties
Home Properties
Component Properties
CORBA Component Package
Packaging Tool
CORBA Component Package
Assembly Tool
Component Assembly Package
Component Descriptor
packager
assembler
Assembly Descriptor
softpkg Descriptor
CORBA Component Package
Deployment Tool
deployer
User written file
Compiler
Generated files
(c) 2004 AG Sysoft - University of Trier

Bemerkungen

- Standards, Standards, Standards, …
  - CORBA-Architektur inkl. IIOP: 1150 Seiten
  - C++ Language Binding: 184 Seiten
  - Java Language Binding: 158 + 84 Seiten
  - CORBA CCM: 434 Seiten
  - …
- War einmal technisch anspruchsvoll, aber andere Ansätze haben überholt
  - Sinnvoll einsetzbar bei Verknüpfung von Legacy Software
  - Verbreitet im Telekomumfeld
- Rückzieher
  - Aus CORBA in KDE wurde DCOP
- CORBA paßt sich an (… und wird noch komplexer)
  - Unterstützung und Anbindung an COM+
  - Unterstützung für WSDL und SOAP (.NET)
  - …