

# Entwicklung eines generischen DB-Editors unter JAVA/NSK

April 2002

Jürgen Depping  
© CommitWork GmbH

**CommitWork**

GmbH für Informationstechnologie

Depping@CommitWork.d  
e

www.CommitWork.de

# Agenda

---

- Vorstellung der Applikation "DB-Editor"
- Architektur des DB-Editor-Client (Dreischichtmodell in UML)
- Kommunikation via CORBA
- Der Server des DB-Editors
- Softwareverteilung des Client (Java Web Start)
- Tool-Einsatz
- Geplante Erweiterungen des DB-Editors

# Applikationsziele

---

## **Applikationsziel:**

Entwicklung eines *generischen* DB-Editors, der auf beliebige Datenbanktabellen zugreifen kann und ...

- keinen zusätzlichen Programmcode benötigt (generisch).
- READ, UPDATE, DELETE und INSERT-Funktionalität besitzt.
- eine einschränkende Suche anbietet.

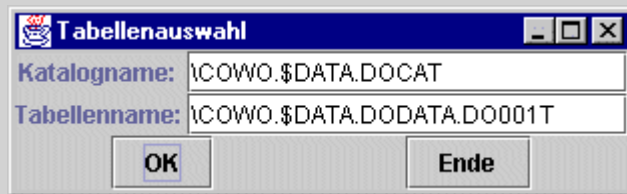
## **Einsatzgebiet:**

- Integration in bestehende Java-Applikationen als einfacher, kostengünstiger Stammdatendialog.
- als Administrator- oder Entwicklertool.

# Die GUI's des DB-Editors

---

Tabellenauswahl nach Katalog und Tabelle:



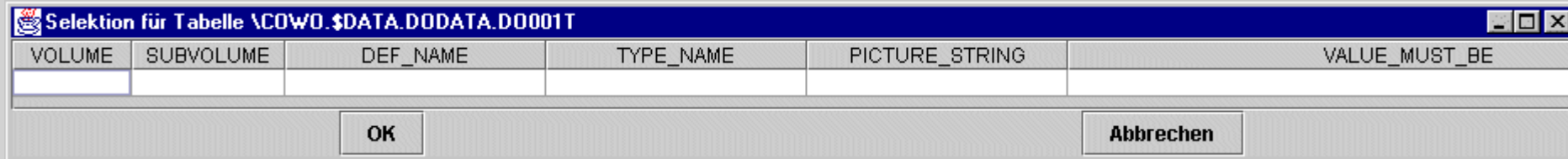
Tabellenauswahl

Katalogname: \COWO.\$DATA.DOCAT

Tabellenname: \COWO.\$DATA.DODATA.D0001T

OK Ende

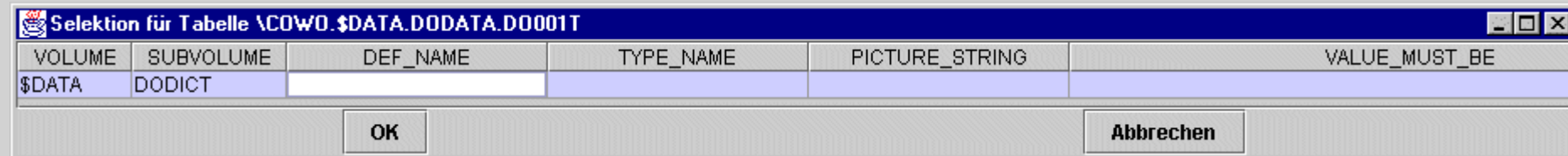
Selektionseinschränkung:



Selektion für Tabelle \COWO.\$DATA.DODATA.D0001T

VOLUME	SUBVOLUME	DEF_NAME	TYPE_NAME	PICTURE_STRING	VALUE_MUST_BE

OK Abbrechen



Selektion für Tabelle \COWO.\$DATA.DODATA.D0001T

VOLUME	SUBVOLUME	DEF_NAME	TYPE_NAME	PICTURE_STRING	VALUE_MUST_BE
\$DATA	DODICT				

OK Abbrechen

# GUI - Tabelle

(StammdatenGUI\_jb)

Lesen der ausgewählten Datensätze:

Akti...	VOLUME	SUBVOLUME	DEF_NAME	TYPE_NAME	PICTURE_STRING
	\$DATA	DODICT	CMSG-1001		X(00122)
	\$DATA	DODICT	CMSG-1002		X(00192)
	\$DATA	DODICT	CMSG-1003		X(02606)
	\$DATA	DODICT	CMSG-1004		X(00142)
	\$DATA	DODICT	CMSG-1005		X(02746)
	\$DATA	DODICT	CMSG-1006		X(01366)
	\$DATA	DODICT	CMSG-1007		X(01364)
	\$DATA	DODICT	CMSG-1008		X(02156)
	\$DATA	DODICT	DATUM		9(8)
	\$DATA	DODICT	DATUM-X		X(10)

Buttons: Neuer Datensatz, OK, Abbrechen

Erneute Selektion der Daten:

Selektion	SUBVOLUME	DEF_NAME
\$DATA	DODICT	CMSG-100
\$DATA	DODICT	CMSG-100

# GUI - Tabelle

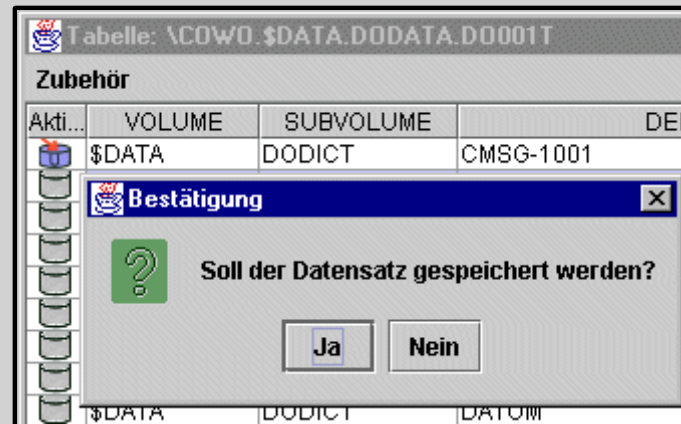
( Datensatz bearbeiten )

Löschen eines Datensatzes:

Akti...	VOLUME	SUBVOLUME	
I	\$DATA	DODICT	CMSG-10
	\$DATA	DODICT	CMSG-10
	\$DATA	DODICT	CMSG-10
	\$DATA	DODICT	CMSG-10

Ändern eines Datensatzes:

Akti...	VOLUME	SUBVOLUME	
f	\$DATA	DODICT	CMSG-1
	\$DATA	DODICT	CMSG-1
	\$DATA	DODICT	CMSG-1
	\$DATA	DODICT	CMSG-1



# Architekturziele des Client

---

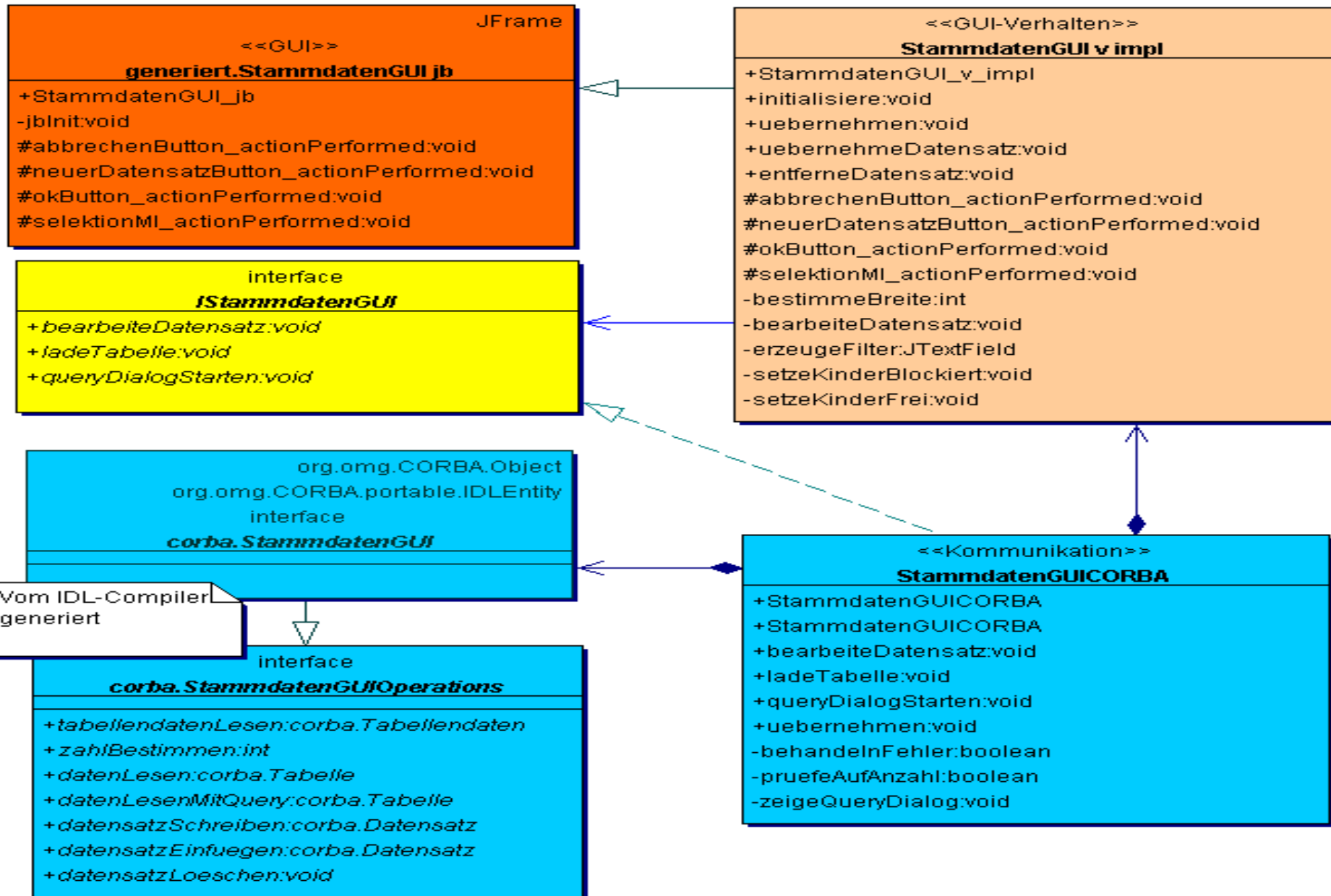
## **Ziele:**

- Unabhängigkeit vom "GUI-Builder".
- Unabhängigkeit von der verwendeten Kommunikations-Technologie.

## **Wird erreicht durch:**

- Dreischicht-Modell
  - Anzeige-Schicht
  - Verhaltens-Schicht
  - Kommunikations-Schicht
- Einsatz von Schnittstellen (Interfaces) und Vererbung.

# Das Dreischicht-Modell des Client



# Kommunikationsschicht via Corba

---

## Common Object Request Broker Architecture:

ist ein offener Standard, durch Object Management Group definiert.

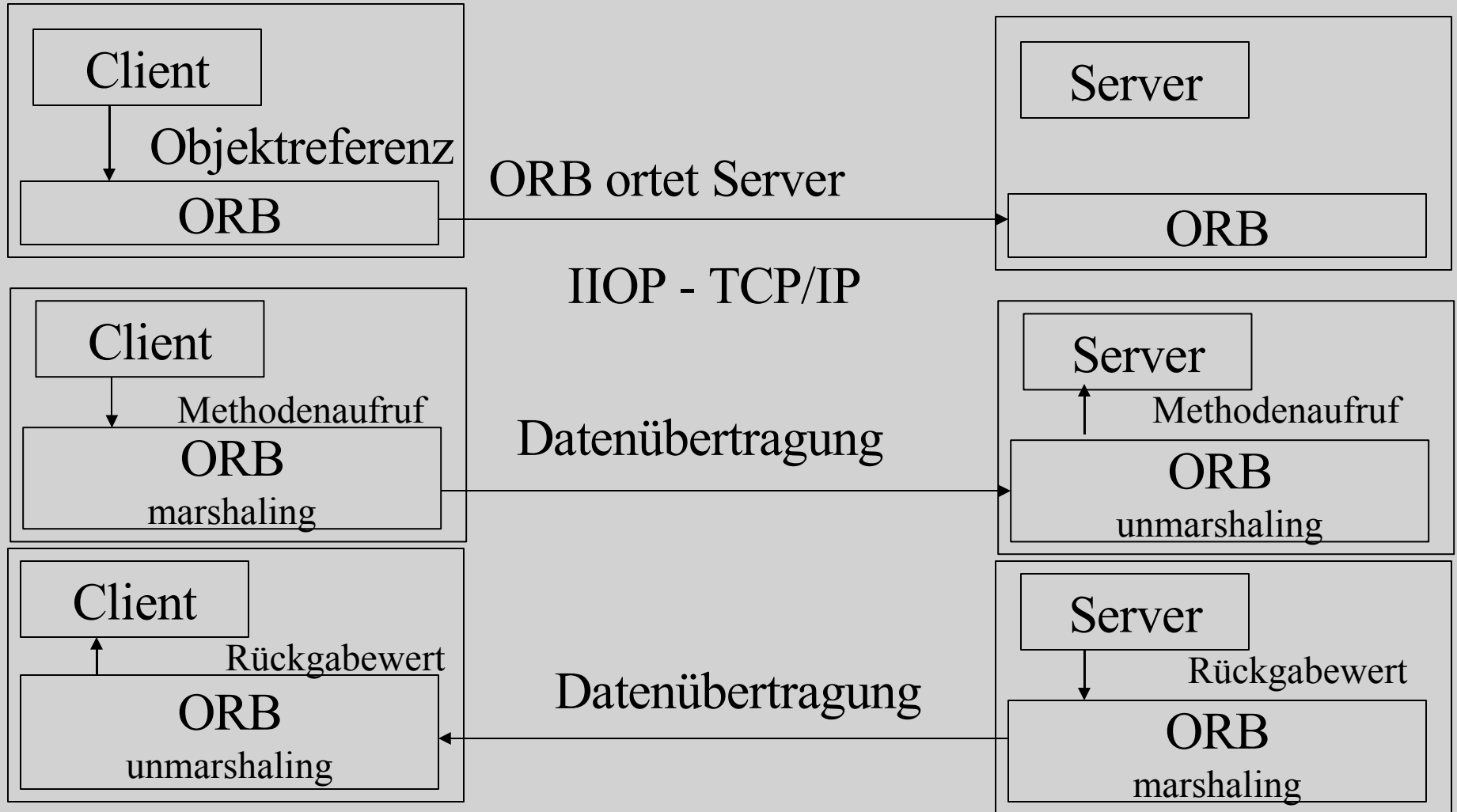
**Aufgabe:** Soll Objekte im Netz transparent verfügbar machen.

- Plattform- und Sprachunabhängig (IDL)
- ab CORBA 2.0 Interoperabilität zwischen verschiedenen Implementationen gewährleistet. ( IIOP=Internet Inter ORB Protokoll )

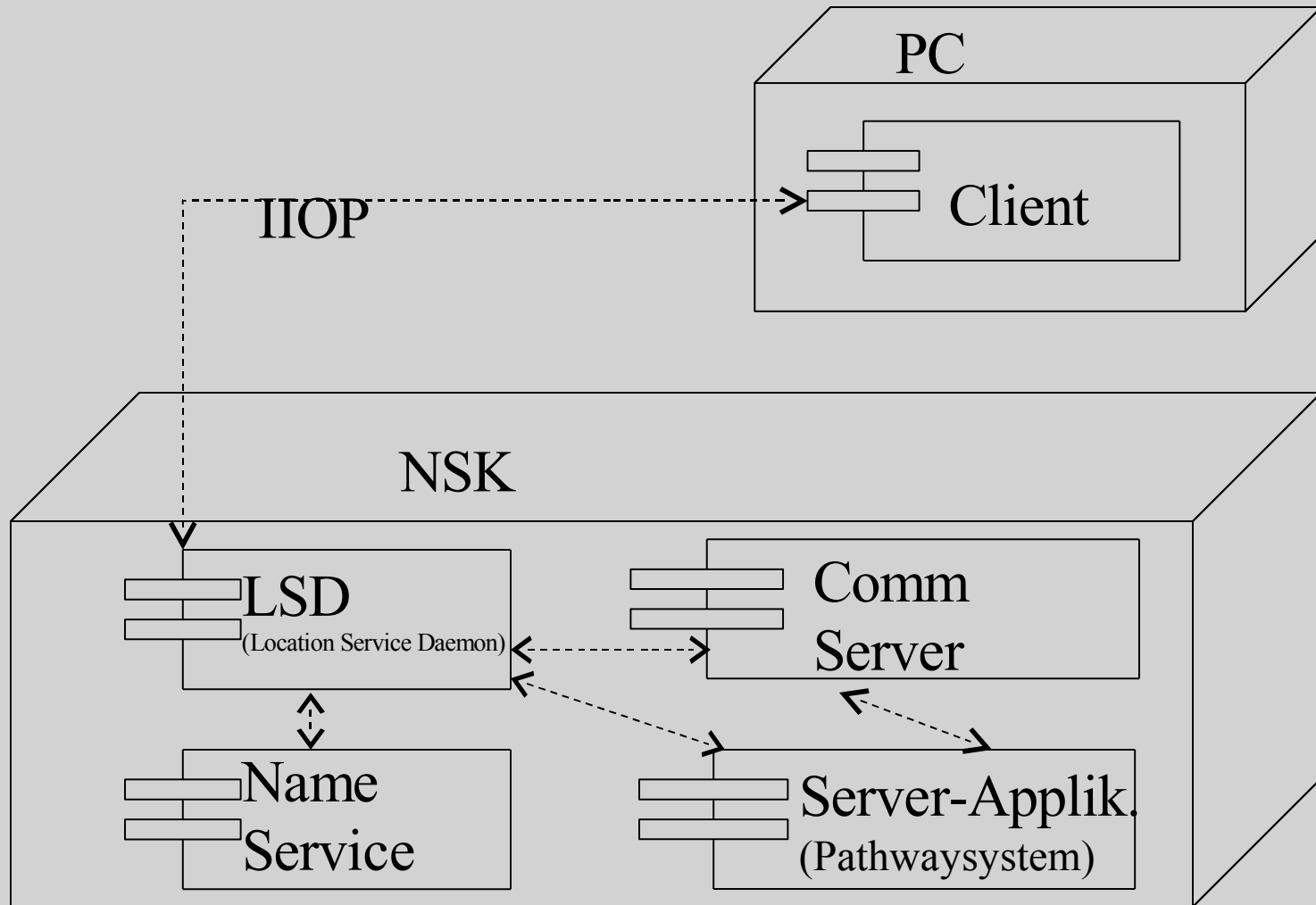
## **Dienste:**

Name Service, Object Transaction Service, Event Service, Security Service

# Systemarchitektur CORBA



# Corba auf NSK (1)



# Corba auf NSK (2)

---

## **Location Server Daemon (LSD) und COMM-Server:**

- Anfragen gehen an LSD.
- LSD verteilt Anfragen auf die COMM Server (load balancing).
- Der ausgewählte COMM-Server startet ggf. den Java-Server

## **Fazit:**

NSK bietet mit seiner Corba-Architektur Funktionalitäten eines Applikation-Servers!

(Alternativ bietet Compaq auch einen J2EE-Applikation Server an.)

# Ziele des Servers

---

- Ausfallsicherheit und Skalierbarkeit des Servers.
- Zugriff auf Metadaten und Daten einer beliebigen Datenbanktabelle.

# Serverkonfiguration

---

Ausfallsicherheit und Skalierbarkeit durch CORBA/NSK mit Pathwaysystemen.

## **Arbeitschritte:**

- Eintrag in Konfigurations-Datenbank (cfgmgt):
  - Protokolleinstellungen, pathmon=\$pmdbe, class\_name=dbeditor
- Pathway-Monitor erzeugen (Shell-Script unter OSS)
  - pmdbe
- Server-Eintrag erzeugen (Shell-Script unter OSS)
  - dbeditor

# Start des Servers

---

1. ORB initialisieren.
2. RootPOA ermitteln.
- 3.1 Eigenschaften per Policies festlegen.
- 3.2 Neuen POA mit den festgelegten Eigenschaften erzeugen und aktivieren.
4. Objekt-Referenz des neuen Corba-Objektes im Nameservice veröffentlichen.
5. ORB starten.

# Informationen einer Tabelle

(JDBC, DatabaseMetaData)

---

```
import java.sql
```

```
...
```

```
// Singleton
```

```
Connection connection = ConnectionFactory.getConnection ();
```

```
...
```

```
DatabaseMetaData md = connection.getMetaData ();
```

```
...
```

```
// Primary Keys bestimmen:
```

```
ResultSet rs = md.getPrimaryKeys ( katalogName, null, tabelle );
```

```
...
```

```
// Tabellenbeschreibung holen:
```

```
rs = md.getColumns ( katalogName, null, tabelle, null );
```

# Datensätze lesen

---

...

```
Connection connection = ConnectionFactory.getConnection ();
```

...

```
Statement stmt=connection.createStatement();
```

...

```
ResultSet rs=stmt.executeQuery(SqlString);
```

...

```
while ( rs.next () ) {  
    for ( int i = 0 ; i < colCount ; i++ ) {  
        ds.daten[i] = rs.getString ( i + 1 );  
    }  
}
```

# Problem: Softwareverteilung des Client

---

1. Auf jedem PC liegt das Clientprogramm:

- Softwareverteilung problematisch.
- + hohe Performance beim Starten.

2. Zentral auf einem Fileserver liegt das Clientprogramm:

- + Softwareverteilung über Netzwerk.
- schlechtere Performance beim Starten.

3. Einsatz der Java-Web-Start-Umgebung:

- a) Entweder Aktualisierung über Webserver oder
- b) falls aktuell, starten der gecachten Version des lokalen Rechners!

+ Vereinigt beide positiven Eigenschaften!

# Zusätzlich eingesetzte Tools

---

## Editoren

- JBuilder ( GUI-Edior )
- Forte for Java ( TEXT-Editor mit CVS-GUI-Tool, ANT-Unterstützung )

## Dokumentation des Systems:

- CASE-Tool Together (Design in UML, Dokumentation nach UML)
- Javadoc (Dokumentation)

## Sourcecode-Verwaltung:

- CVS (Versionierung, Entwicklung im Team)

## Compilieren des Projektes:

- ant

## Client-Softwareverteilung

- Java Web Start

# Erweiterungen des DB-Editors

---

- Benutzerverwaltung mit Nutzungseinschränkung.
- Zugriff auf die Tabellennamen per DEFINE-Name
- Konfigurierbare LOOKUP-Datenbank zur Unterstützung der Feldeingabe.  
(Tabelle, Spalte, SQL-Statement)

# Literaturhinweise

---

## **Java:**

- Core Java Band 1, Prentice Hall, Cay S. Horstmann & Gary Cornell
- Core Java Band 2, Prentice Hall, Cay S. Horstmann & Gary Cornell
- Graphic Java "Die JFC beherrschen (Swing)", Prentice Hall, David M. Geary

## **Corba:**

- "Client/Server Programming with Java and Corba", Wiley Computer Publishing,  
Robert Orfali & Dan Harkey

## **UML:**

- "Analyse und Design mit der Unified Modeling Language", R. Oldenburg Verlag, Bernd Oestereich
- "UML konzentriert", Addison-Wesley Verlag, Grady Booch & Ivar Jacobson & James Rumbaugh
- "The unified modelling language User Guide", Addison-Wesley Verlag, Grady Booch &  
Ivar Jacobson & James Rumbaugh

## **Pattern:**

- "Entwurfsmuster", Addison-Wesley Verlag, Erich Gamma & Richard Helm & Ralph Johnson &  
John Vlissides