Vorlesung
Component Ware und Web-Services

- Komponentenkonzepte -

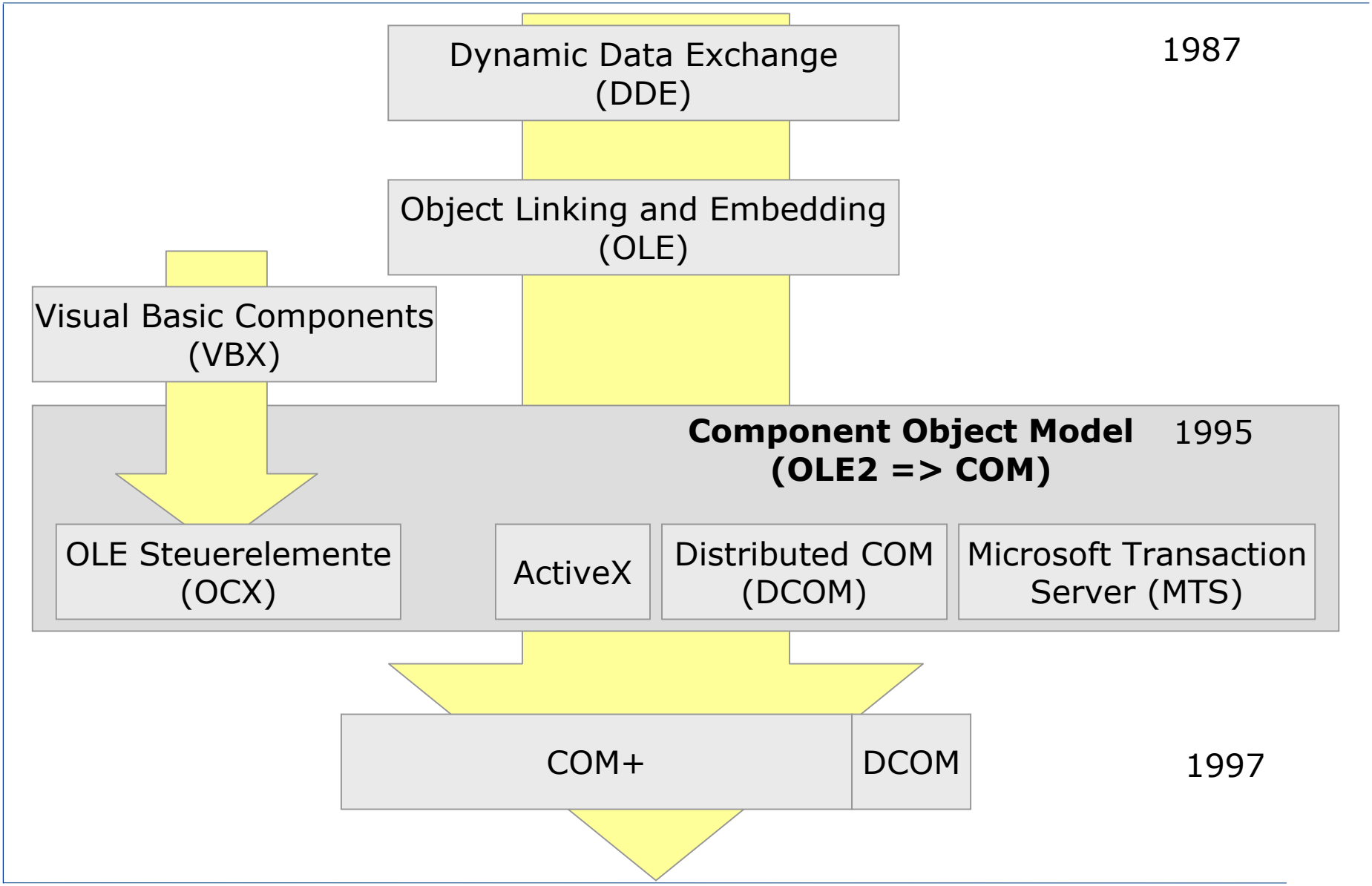
4. Component Object Model (COM)

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

Inhalt

- **Geschichte**
- **Die Schnittstelle**
- **COM-Objekte**
  - **Bedeutung der Registry**
  - **Erzeugung eines COM-Objektes**
  - **Kommunikation**
  - **Lebenszyklus**
- **COM-Dienste**
- **OLE**
- **Das strukturierte Speichermodell**
- **Wiederverwendung**
- **DCOM**
- **COM+**

Geschichte


Dynamic Data Exchange (DDE)
1987
Object Linking and Embedding (OLE)
Visual Basic Components (VBX)
Component Object Model (OLE2 => COM)
1995
OLE Steuerelemente (OCX)
ActiveX
Distributed COM (DCOM)
Microsoft Transaction Server (MTS)
COM+
DCOM
1997

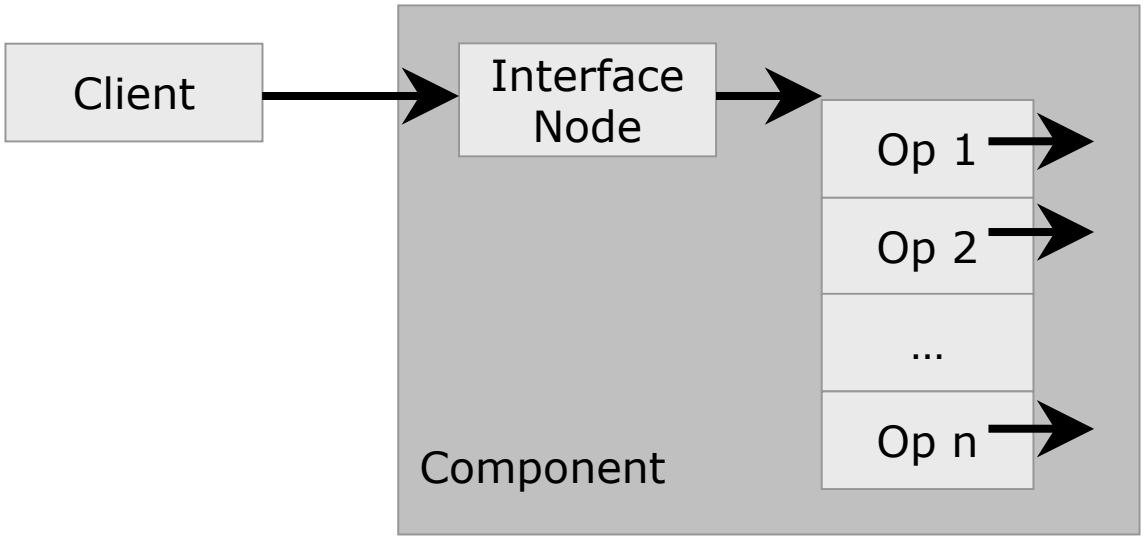
Geschichte

- **Historie**
  - Idee des dokumentenzentrierten Arbeitens
  - Ein Verbunddokument wird aus vielen Teildokumenten zusammengesetzt
    - OLE (object linking and embedding)
  - Teildokumente bzw. Komponenten werden in ein Zentraldokument / einen Container eingebettet
  - OLE 1: Modell speziell für Verbunddokumente
  - OLE 2: Allgemeines Modell für Komponenten
    - Umbenennung in Component Object Model
- **Visual Basic Controls (VBX)**
  - Einfaches, statische Modell zum Einbetten von Controls in Formulare
- **OLE controls (OCX)**
  - generelle COM-Container

Geschichte

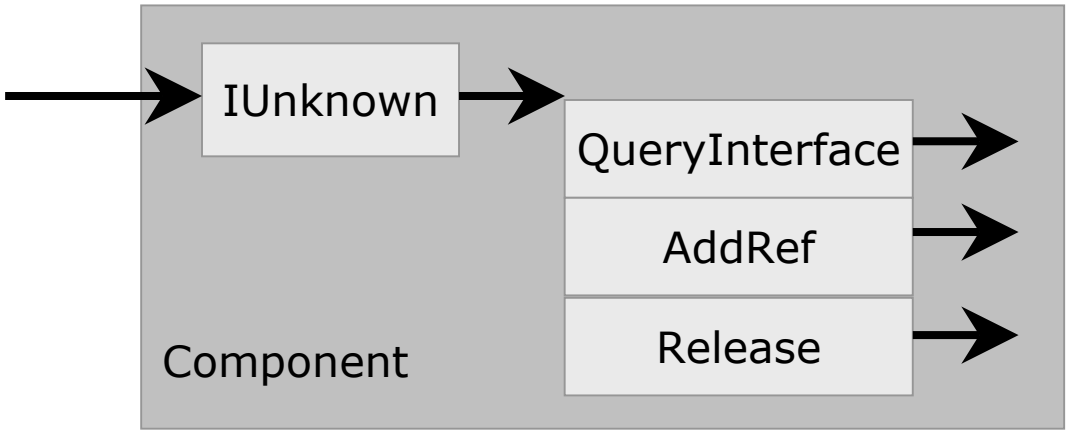
- **ActiveX**
  - alle Schnittstellen (bis auf IUnknown) optional
  - selbstregistrierende Server
- **DCOM** (Distributed COM)
  - Zugriff auf COM-Objekte auf entfernten Maschinen
- **Microsoft Transaction Server**
  - Transaktionsverarbeitung COM-basierter Applikationen
  - Zuweisung von Serverressourcen
- **COM+**
  - Erweiterung von COM

Die Schnittstelle



- COM ist binärer Standard
- Schnittstelle ist zentraler Punkt von COM
- Schnittstellenvererbung
- Interface Node
  - Methoden eines Objekts => **"this"**-Parameter wird an jede Methode weitergereicht
  - Komponente kann n Interface implementieren

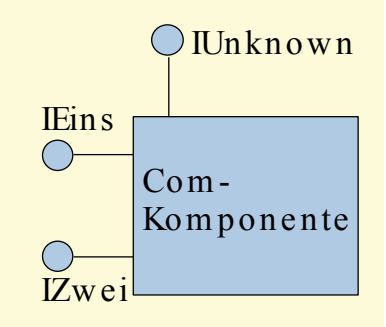
Die Schnittstelle IUnknown



- Schnittstelle IUnknown **muß** von jeder Komponente implementiert werden
  - jede andere Schnittstelle erbt von IUnknown
- QueryInterface
  - erste Methode jedes COM-Objektes
  - gibt Zeiger auf gesuchtes Interface zurück
  - benutzt Interface Identifier (IID)
- AddRef, Release
  - Verwaltung des Lebenszyklus
  - Referenzzähler

Die Schnittstelle IUnknown

```
[ uuid(12345678-1234-1234-1234-123456789ABC) ]
interface IUnknown {
    HRESULT QueryInterface ([in] const IID iid, [out, iid_is(iid)]
    IUnknown iid);
    unsigned long AddRef();
    unsigned long Release();
}
```



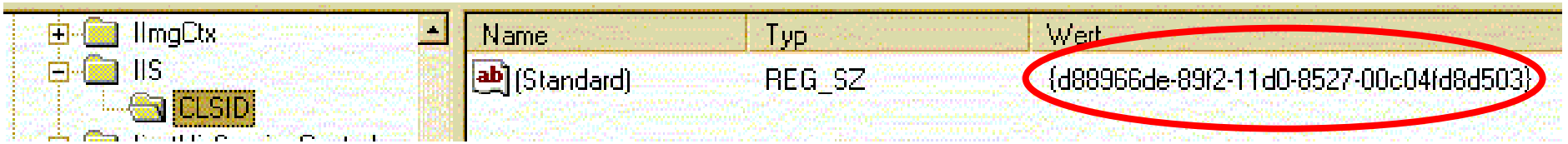
- IID ist eindeutiger Bezeichner/Name
- "sprechender" Name als Alias, generell beginnend mit einem I
- IID ist Global Unique Identifier (GUID)
  - Global eindeutige Nummer, die nach einem speziellen Algorithmus gebildet wird und in Zeit und Raum eindeutig ist
  - Im Zusammenhang mit Schnittstellen werden GUIDs auch als Interface Identifier (IIDs) bezeichnet

COM-Objekte

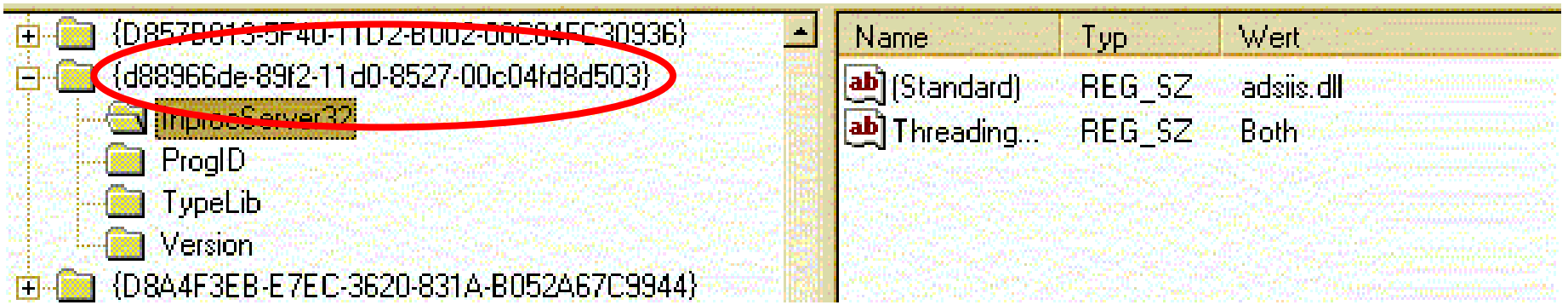
- COM-Objekt vs. COM-Komponente
  - Literatur : COM-Objekt (COM object) wird Irreführenderweise synonym für die Komponenten und die von ihnen erzeugten Objekte benutzt
  - Nachfolgend wird explizit zwischen einer COM-Komponente und einem COM-Objekt unterschieden
  - Ein COM-Objekt ist ein von einer COM-Komponente erzeugtes Objekt
- Komponente in einer DLL
  - Die DLL wird in den Prozessraum des aufrufenden Clients geladen
    - Deshalb auch prozessinterner Server genannt
    - Vorteil: Schnelle Kommunikation
- Komponente in einer Anwendung
  - Als lokaler Server (local server) bezeichnet
    - Das Modul ist eine Anwendung (.exe-Datei)
    - Nachteil: langsamer Zugriff
    - Vorteil: Laufen als eigenständige Anwendung.

COM-Objekte – Bedeutung der Registry

- Eine COM-Komponente **muss** registriert werden
- Zweig HKEY_CLASSES_ROOT/CLSID

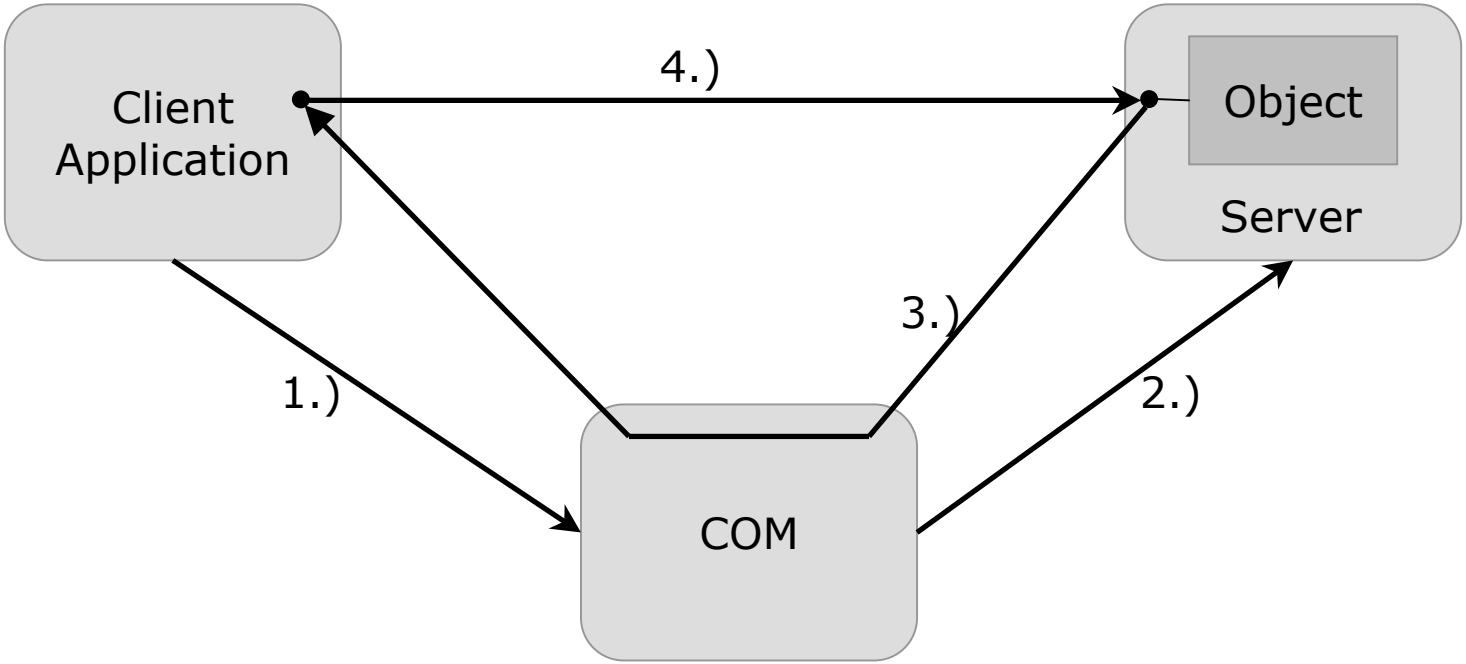


- Ein Unterschlüssel enthält den Dateinamen des Moduls, welches die Komponente implementiert
  - Der Name des Schlüssels ist InprocServer32 bei DLL-Modulen, LocalServer32 bei Exe-Modulen.



- Problem: Schlüssel kann (mutwillig oder irrtümlich) von anderen Komponenten überschrieben werden

COM-Objekte – Erzeugen eines Objektes



1. Aufruf der Funktion "CreateObject"
2. COM lokalisiert/erzeugt den Server
3. Serverprozess erzeugt das Objekt und gibt einen Schnittstellenzeiger zurück
4. Client kommuniziert über den Schnittstellenzeiger mit dem Serverobjekt

COM-Objekte – Erzeugen eines Objektes

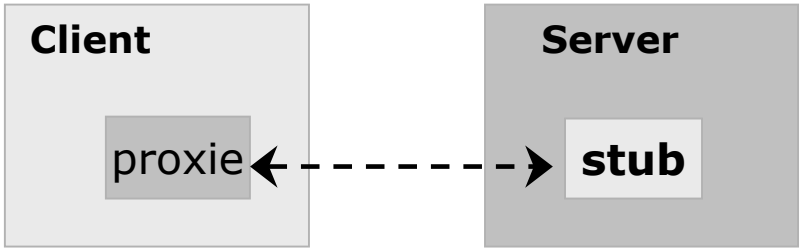
- Komponente wird über ClassId (CLSID) identifiziert
- COM bietet Bibliothek zur Erzeugung neuer Objekte über die CLSID der zugehörigen Komponente

- Funktion CoCreateInstance(CLSID, IID, Server)
  - "Co" für COM-Prozedur
  - Erzeugt Instanz des Objektes (CLSID) und gibt Zeiger zurück (IID)
  - Server: In-Process, lokal oder remote
  - COM findet Objekt über Registry
  - in-process server: Laden und linken einer DLL
  - local server: seperate ausführbare Datei (EXE)
  - remote server: Service Control Manager (SCM) auf entfernter Maschine wird kontaktiert

COM-Objekte – Kommunikation

- bei Kommunikation über Prozessgrenzen hinweg (Local Server), keine direkte Kommunikation möglich
  - Client und Komponente laufen in eigenen Adressräumen

- Stellvertreter und Stummel
  - Kommunikation zwischen Client und Server erfolgt über Stellvertreter (proxies) und Stummel (stubs)
  - Stellvertreter laufen im Adressraum des Clients und vertreten die eigentliche Komponente
  - Stummel laufen im Adressraum des Servers und stellen den Kontakt zwischen Stellvertreter und eigentlicher Komponente her

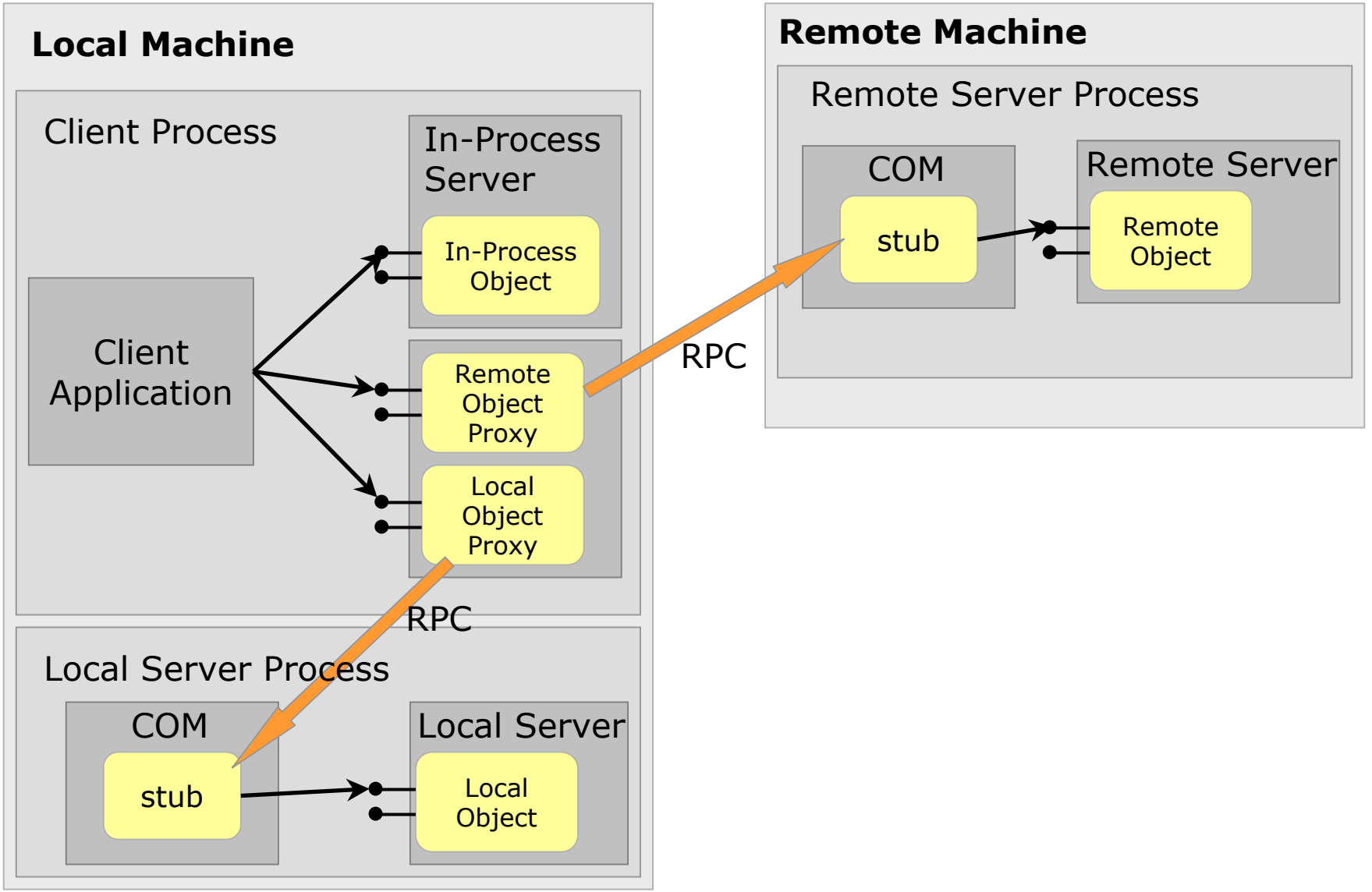


COM-Objekte – Kommunikation

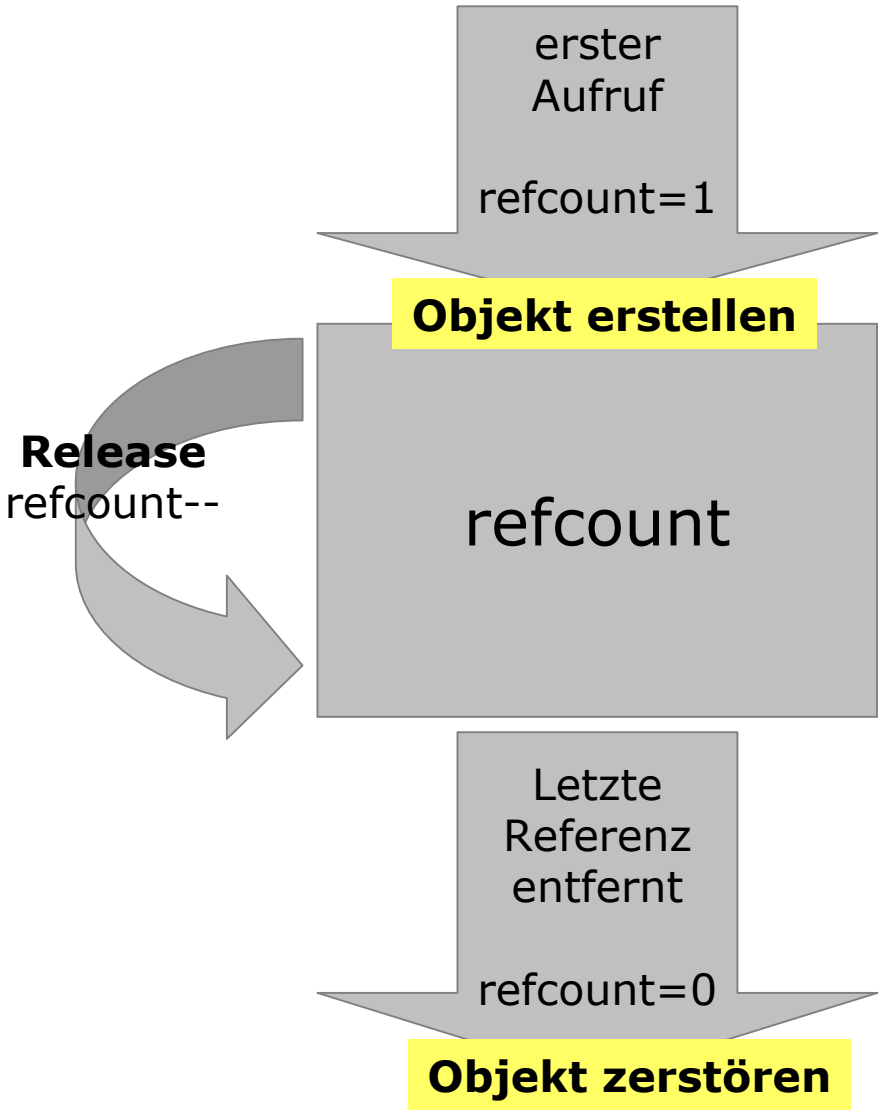
- **Stellvertreter** und **Stummel**
  - Stellvertreter und Stummel werden von einem Werkzeug generiert
  - Das Werkzeug benötigt dazu eine Beschreibung der Komponente und vor allem ihrer Schnittstellen
  - Diese Beschreibung liefert die IDL
  - Hat man eine Schnittstelle oder Komponenteninformationen in der IDL formuliert, dann kann man anschließend mit dem Microsoft IDL-Compiler (MIDL-Compiler) Stellvertreter und Stummel automatisch erzeugen.

- **Begriffsverwirrung**
  - Außerhalb des COM-Kontextes
    - Stellvertreter/proxie (COM) = Stummel (stubs)
    - Stummel/stub (COM) = Skelette (skeletons)

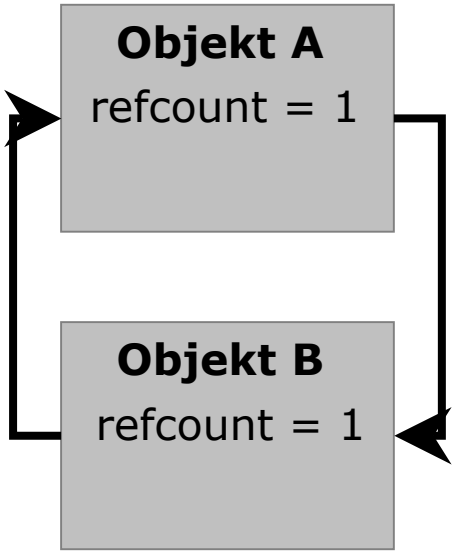
COM-Objekte - Aufrufe



COM-Objekte - Lebenszyklus



Problem zyklische Referenzen



COM – Dienste

- Type Information
  - Laufzeitzugriff auf Typinformationen des COM Objektes
  - wird vom Microsoft IDL Compiler generiert und in einer Typbibliothek gespeichert
  - COM-Schnittstellen zum Interagieren mit dieser Bibliothek
- Structured Storage and Persistence
  - von COM unterstützte Methode zum Speichern von Daten
  - Speicherung erfolgt analog des Dateisystems in einer Datei
- Monikers
  - COM-Objekt, welches ein einzelnes Objekt in einem genau definierten Zustand erzeugen und initialisieren kann
  - für Clients, die mit exakt dem gleichen Objekt weiterarbeiten müssen
  - Moniker kann an gesamtes Objekt oder an einen Teil gebunden werden

COM – Dienste

- Uniform Data Transfer
  - Datentransfer zwischen COM-Objekten
  - Benachrichtigung von Datenänderungen einer Quelle (Datenobjekt) und einem Datenkonsumenten
- Connectable Objects
  - zur Ereignisverarbeitung
  - Objekt definiert ein Interface, welches für das Ereignis genutzt werden soll
  - Client implementiert dieses Interface

OLE

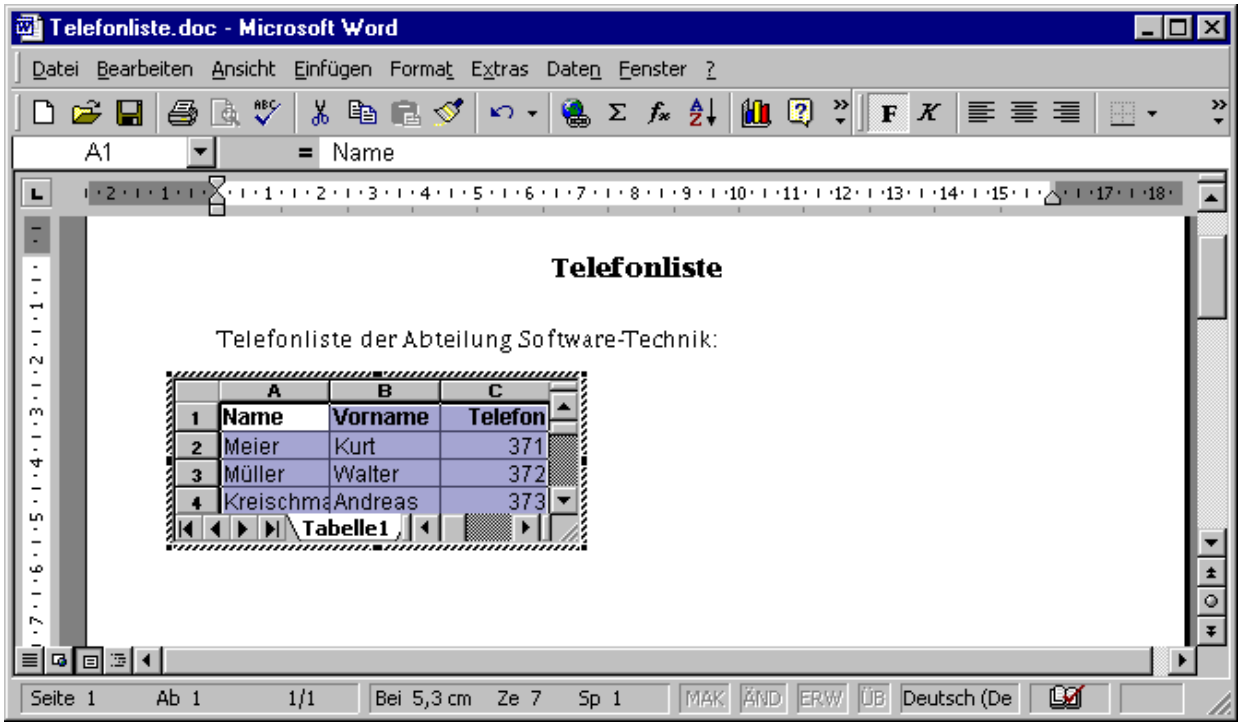
- **Anforderung**
  - In dokumentenbasierten Anwendungen besteht der Wunsch, Bilder, Tabellen usw. per »drag and drop« in andere Dokumente zu übernehmen

- **Architektur**
  - Dokument-Server
    - Kann Inhalte verwalten und darstellen
  - Dokument-Container
    - Kann Dokumenten-Server aufnehmen
  - Eingebettete Dokumente sollen dort bearbeitet werden, wo sie im umgebenen Dokument dargestellt werden (in-place activation).
  - Wird einem eingebetteten Dokumenten-Server mitgeteilt, dass der dargestellte Inhalt verändert werden soll, so wird – unsichtbar für den Benutzer – vom Dokument-Server über dem eigentlichen Inhalt ein neues Fenster geöffnet, in dem der Inhalt verändert werden kann

OLE – Eingebettet Dokumente

- Es entsteht die Illusion, dass der Inhalt des eingebetteten Dokuments direkt im umgebenden Dokument verändert werden kann
- Zusätzlich wird der Inhalt und/oder die Semantik einiger Menüs und Werkzeugleisten ausgetauscht.

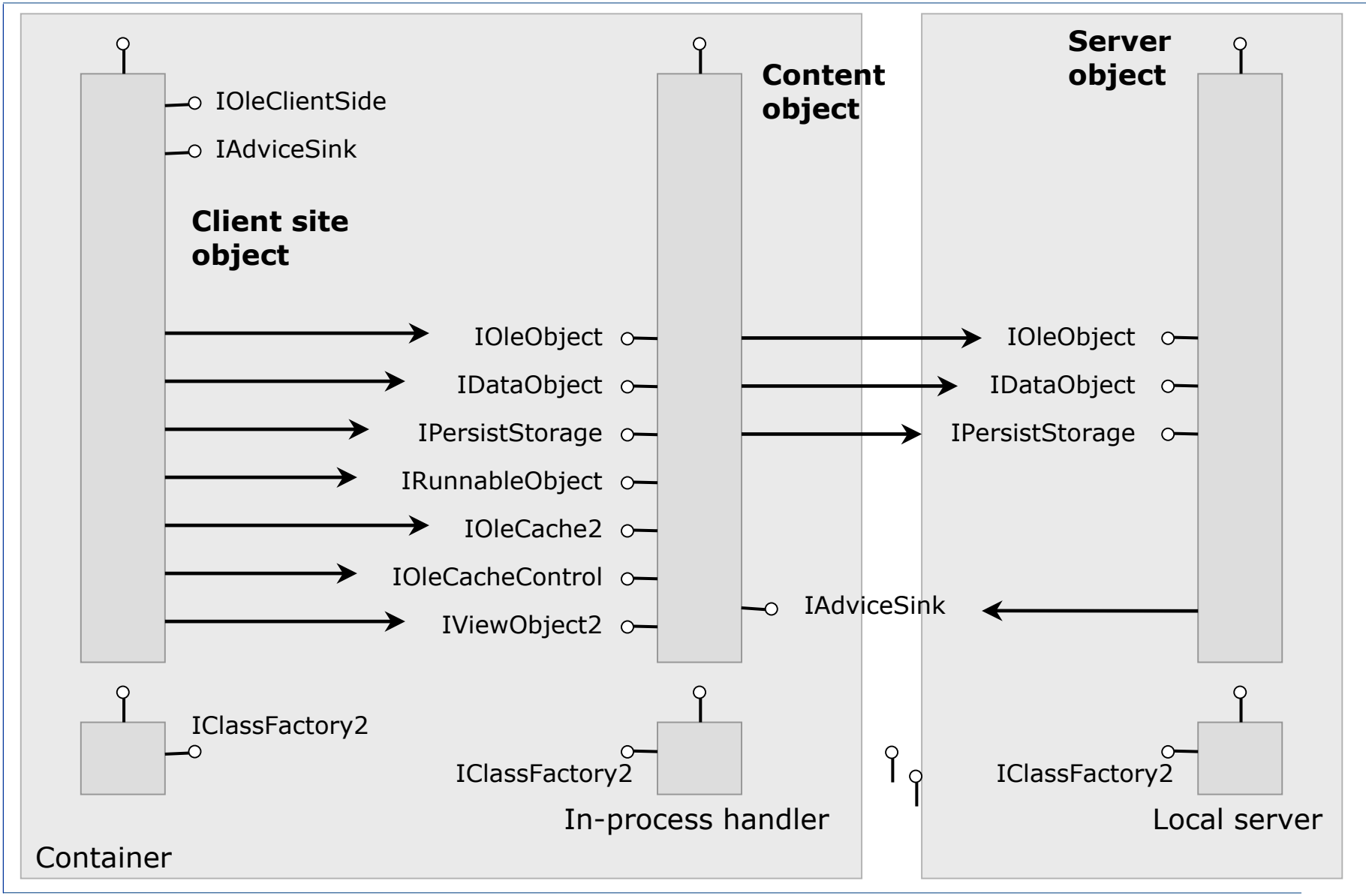
- Aktivierte Excel-Tabelle in Word



OLE - Verweise

- Nur ein Verweis auf das eigentliche Dokument wird in dem Dokument-Container gespeichert
- Die Daten des Dokuments, auf das verwiesen wird, werden in einer externen Datei gespeichert
- **Vorteile**
  - Derselbe Inhalt kann in mehrere Dokumente gleichzeitig integriert werden
  - Änderungen am Dokument bleiben dadurch in allen Dokumenten konsistent
- **Nachteile**
  - Verweis wird ungültig, wenn das externe Dokument gelöscht oder verschoben wird
  - Keine in-place activation (es wird ein separates Fenster geöffnet).

OLE - Schnittstellen

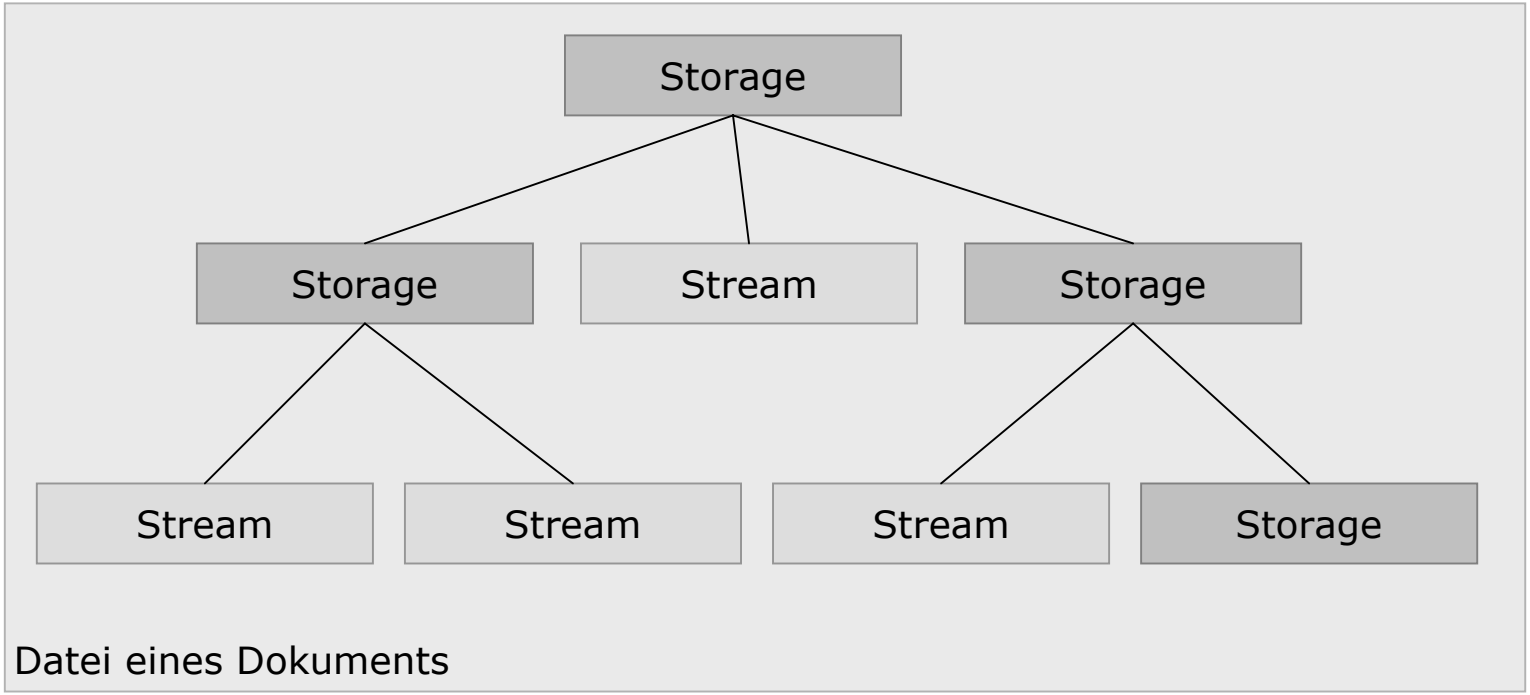


Das strukturierte Speichermodell

- Wird eine Excel-Tabelle in ein Word-Dokument eingebettet, dann wird das gesamte Verbund-Dokument nur in einer .doc-Datei abgespeichert
- Dieselbe Komponente, die die Darstellung und Verwaltung der Tabellendaten realisiert, muss also ihre Daten je nach Kontext auf völlig unterschiedliche Art und Weise speichern können
- Wenn die Komponente in Excel selbst benutzt wird, muss eine eigene Datei angelegt werden
- Das strukturierte Speichermodell (structured storage) hilft einer Komponente dabei – unabhängig vom Kontext – ihre Daten persistent abzuspeichern.
- Es existieren analog zu Verzeichnissen und Dateien sogenannte storages und streams
- Die Verzeichnisse bilden die storages und die Dateien die streams in diesem internen Dateisystem

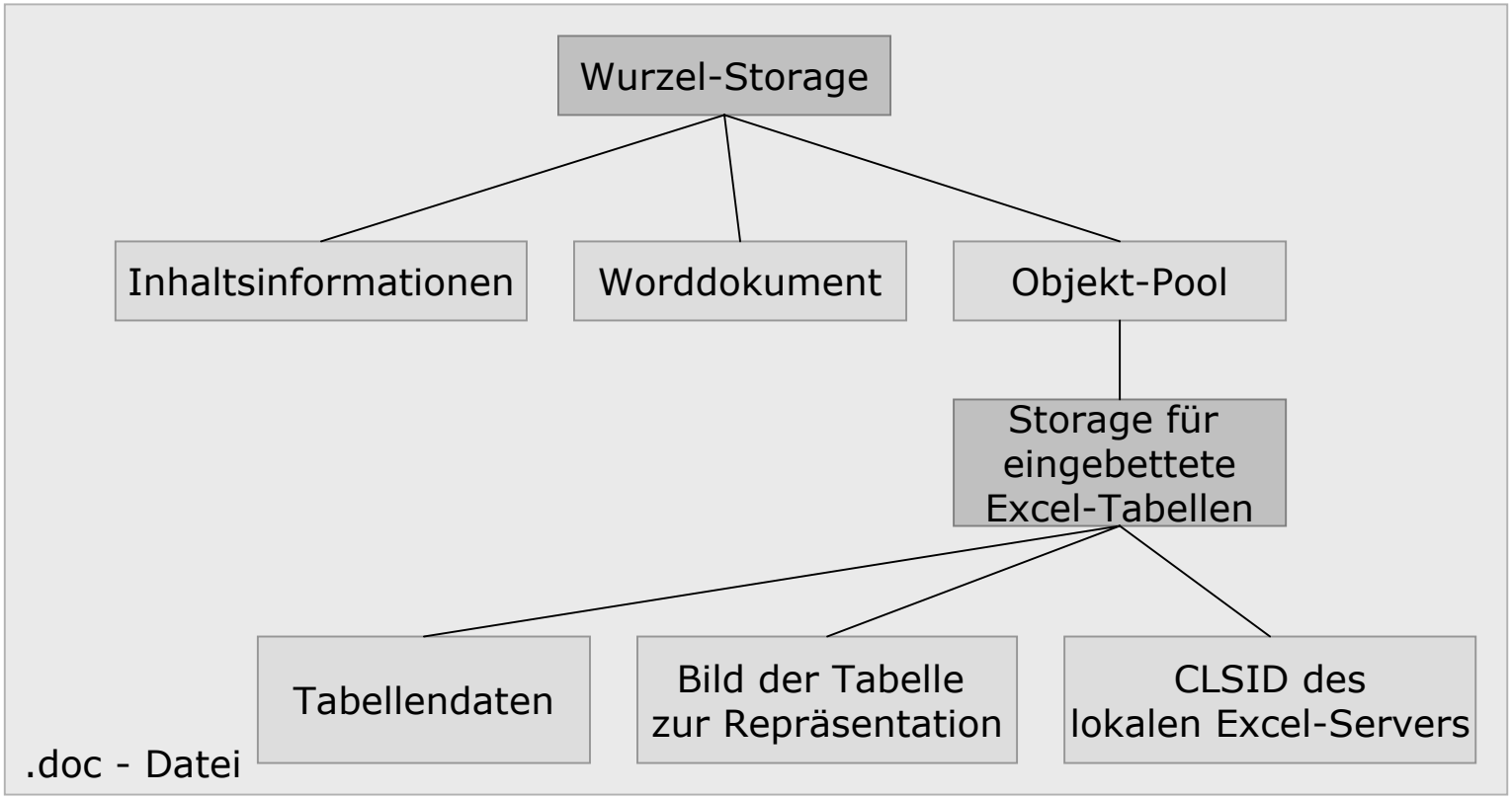
Das strukturierte Speichermodell

- Storages und Streams in einer Datei



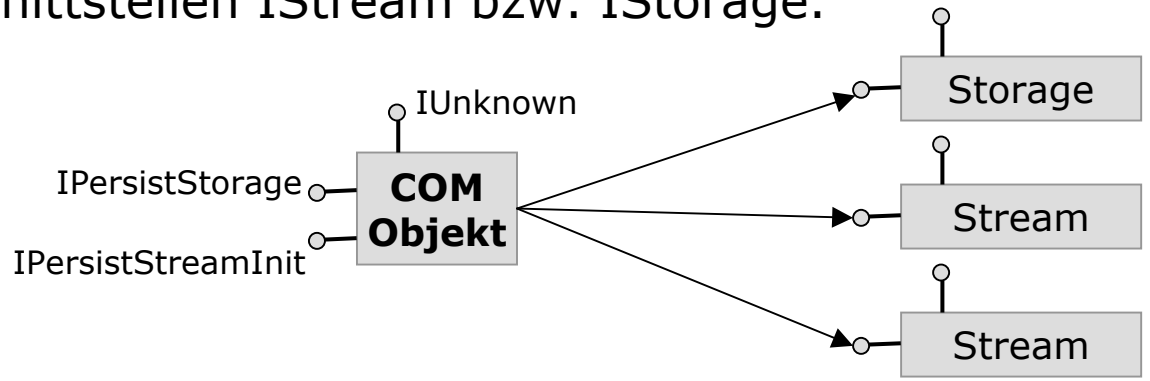
Das strukturierte Speichermodell

- Will ein Container (z.B. ein Word-Dokument) sich und seine Komponenten abspeichern, so legt er für jede Komponente storages und streams in der eigentlichen Datei an und übergibt der Komponente einen Zeiger auf das ihr zugehörige storage



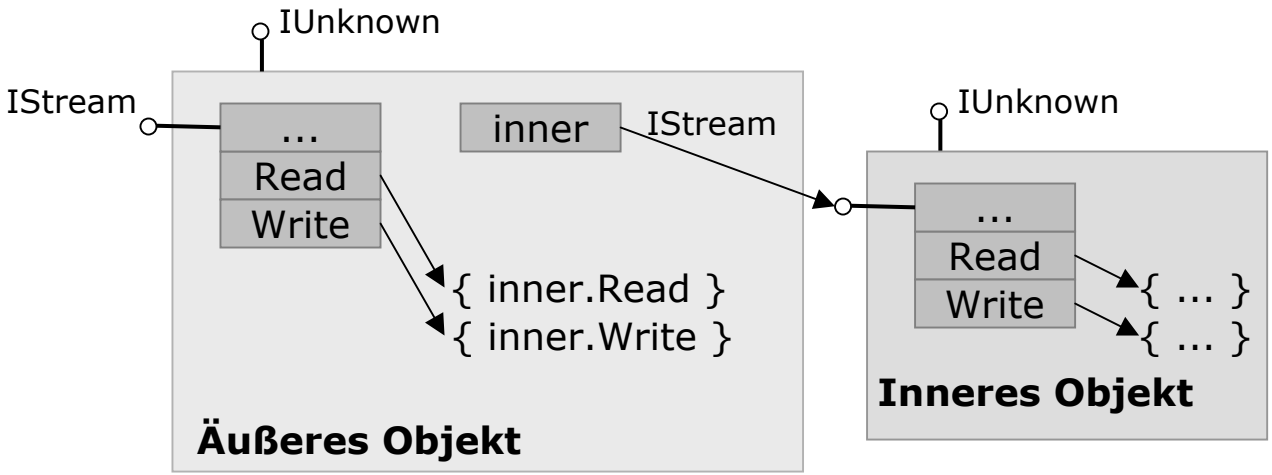
Das strukturierte Speichermodell

- **Schnittstelle IPersistStorage**
  - Jede Komponente, deren Objekte in einen OLE-Container eingebettet werden sollen, muss diese Schnittstelle implementieren
  - Sie enthält die Operationen Load und Save, die ein storage-Objekt übergeben bekommen und vom Container aufgerufen werden
  - Einfach-strukturierte Daten können direkt in ein stream-Objekt gespeichert werden
    - Solche Komponenten implementieren die Schnittstelle IPersistStreamInit
  - Die stream- und storage-Objekte selbst implementieren die Schnittstellen IStream bzw. IStorage.



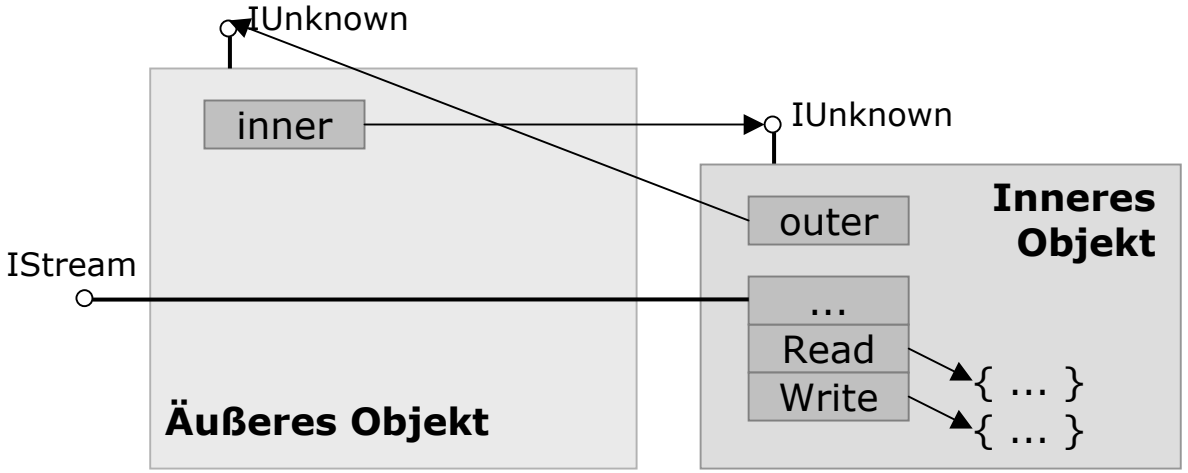
Wiederverwendung

- COM unterstützt keine Vererbung (nur Schnittstellenvererbung)
- Innere der Komponente kann auf Vererbung basieren
- 2 Wege für Wiederverwendung
  - **Kapselung** (containment)
    - Objekt hat exklusive Referenz auf ein anderes
    - Äußeres Objekt "enthält" Inneres Objekt
    - Aufrufe an Inneres Objekt werden weitergeleitet (simpler Methodenaufruf
    - Transparent gegenüber Client-Objekten



Wiederverwendung

- COM unterstützt keine Vererbung (nur Schnittstellenvererbung)
- Innere der Komponente kann auf Vererbung basieren
- 2 Wege für Wiederverwendung
  - **Aggregation**
    - Referenz des Inneren Objekte wird herausgereicht
    - Kein Weiterleiten => Zeitersparnis
    - Inneres Objekt muss Aggregation unterstützen
    - Interface des Inneren Objekte muss Interface des Äußeren kennen
    - Mehrere Hierarchieebenen möglich



Wiederverwendung

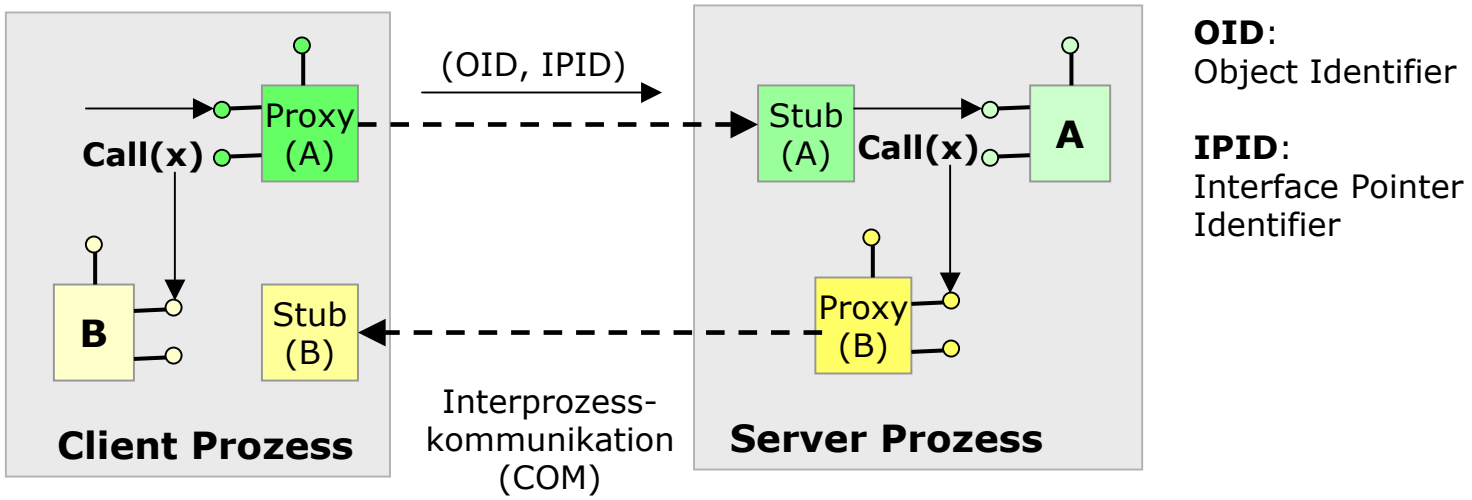
- **Aggregation** versus **Kapselung**
  - Wegen Zirkelbezug bei Aggregation spezielle (fehleranfällige) Regeln für Referenzzählung
  - Aggregation für Performancegewinne und stark verschachtelte Konstruktionen
  - Aggregation für generische Wrapper (Aggregation beliebiger Interfaces)
    - Technik wird beim Microsoft Transaktion Server verwendet
  - Kapselung ist weniger komplex
  - Kapselung wird in der Praxis häufiger eingesetzt

Distributed COM (DCOM)

- DCOM verwendet Proxies und Stubs von COM
- Schnittstellenreferenz muss auf Objektreferenz gemapt werden

1. COM stellt sicher, dass Serverobjekt (stub) existiert
2. COM wählt das korrespondierende Interface des Proxies
3. diese Referenz wird an Stelle der eigentlichen weitergegeben

- Beispiel: Client ruft Objekt **A** auf dem Server -Call(x)- und übergibt dabei eine Referenz auf Objekt **B**



Distributed COM (DCOM)

- 2 Unterschiede zu COM
  - Repräsentation der Datentypen kann unterschiedlich bei unterschiedlichen Maschinen sein
  - Objektreferenzen benötigen mehr Daten als OID und IPID

- **Datenrepräsentation**
  - Serialisierung der Daten in "network data representation" (NDR) – plattformunabhängiges Format

- **Objektreferenzen**
  - Informationen zur Lokalisierung eines "object exporter"
  - object exporter ID (OXID) wird der Objektreferenz hinzugefügt

Distributed COM (DCOM)

- **object exporter**
  - Objekt von DCOM, welches weiß, wie Objekte, die von einem Server exportiert wurden, gebunden werden
  - Information können im Cache gehalten werden
    - beschleunigt Auflösung von Objektreferenzen
  - beim erstmaligen Aufruf werden Informationen aus der Objektreferenz verwendet
  - URL-ähnlicher String zur Identifikation der entfernten Maschine
  - OXID resolver object auf entfernter Maschine liefert Informationen über object exporter zurück

Distributed COM (DCOM)

- **Abstrahierte Mechanismen (high-level)**
  - Beschleunigung entfernter Operationen
  - Sicherheit
  - Auffinden von Fehlern entfernter Maschinen

- **Sicherheitsmechanismen und –granularitäten**
  - Sicherheitsrichtlinien für
    - die gesamte Maschine
    - einzelnen COM-Server
    - individuelle Schnittstellen eines Servers
  - Zugriffskontrolle über "access control lists" (ACL)
    - Authentifizierung pro Verbindung
    - Authentifizierung pro Nachricht
    - Authentifizierung pro Paket
  - Schutz der Daten
    - Verschlüsselung
    - Prüfsumme (fingerprinted)

COM+

- Nachfolger von COM, MTS Oktober 1997
- ab Windows 2000 im Betriebssystem enthalten
- leichtgewichtige Objektmodelle
- benutzt DCOM für Interprozesskommunikation
- Garbage Collection, keine Referenzzähler
- Standard Services
  - Sicherheit
  - Transaktionen
  - Message Queue
  - Load Balancing
- COM und COM+ Objekte können miteinander interagieren

COM+ Event Service

- Synchrone Kommunikationen zwischen Komponenten führt zu Abhängigkeiten
- Deshalb Kommunikation über Event Service
- Publisher sendet Ereignis
- Subscriber empfängt Ereignis
- Publisher muß Subscriber nicht kennen
- Definition von Filtern möglich

COM+ Message Queue Server

- Verfügbarkeit des Empfängers nicht immer gewährleistet
- Message Queue Server bietet:
  - Garantierte Auslieferung der Nachricht
  - Beachtung der Sendereihenfolge
  - Asynchrone Kommunikation mit Möglichkeit der Empfangsbestätigung
  - Kostenbasiertes Routing mit Umgehung von Netzwerkproblemen