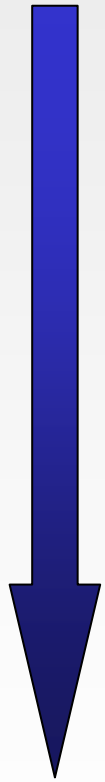
Middleware für verteilte industrielle Umgebungen

(Distributed) Component Object Model

ifak

Entwicklungsschritte der Software-Entwicklung

- Traditionelle Software-Entwicklung
  - Strukturierter Entwurf, ...
  - C, Pascal, VB
- Objektorientierte Software-Entwicklung
  - OOD, Implementierung
  - C++, Modula, Java, C#
- Komponentenorientierte Software-Entwicklung
  - (D)COM
  - JavaBeans
  - .Net
- Serviceorientierte Architektur ???



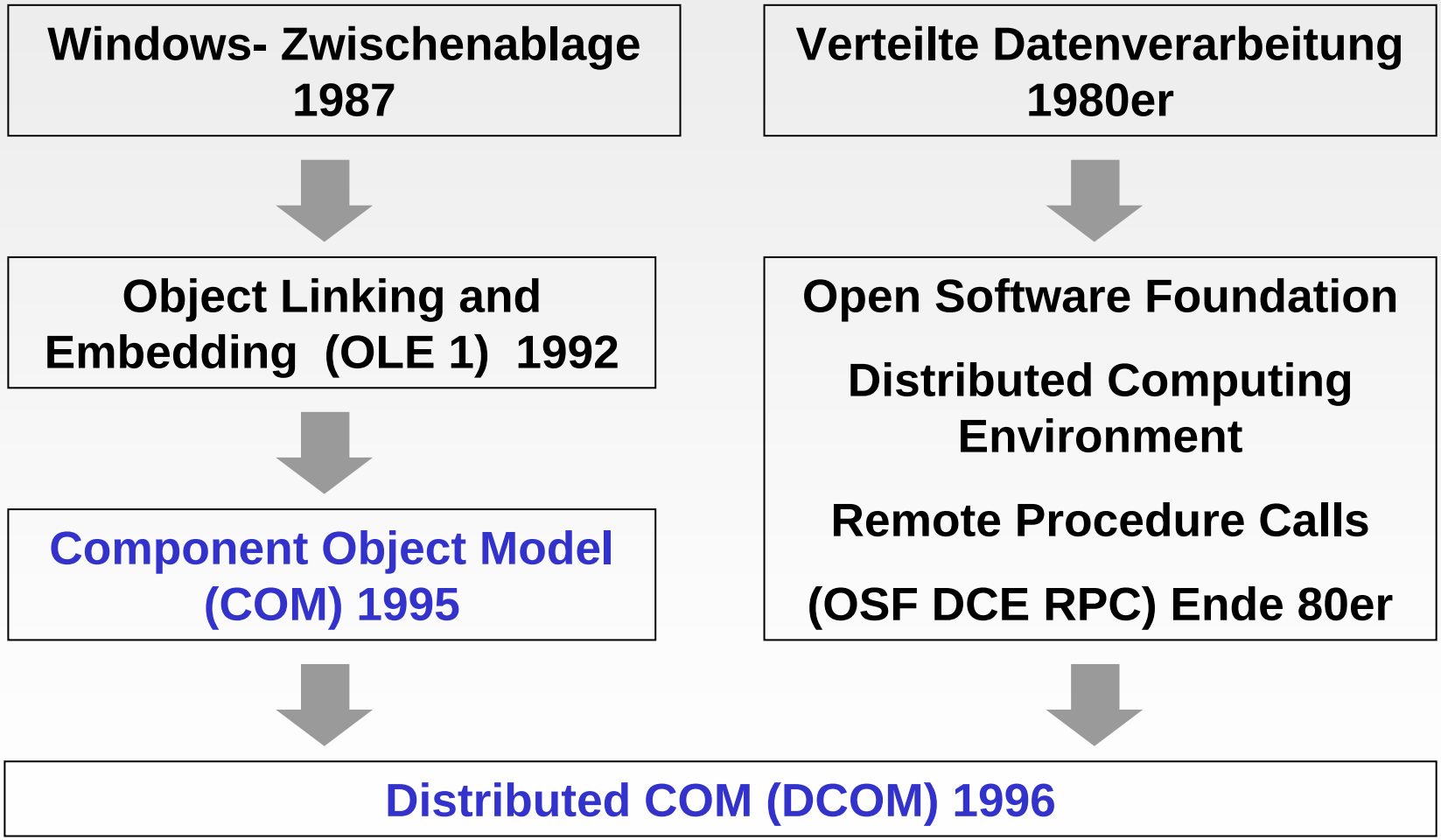
Kapselung,
Wiederver-
wendbarkeit

Warum Komponenten?

- Traditionelle Software
  - unübersichtliche, monolithische Struktur (teilweise)
  - aufwendig zu erstellen
  - schwierig zu warten bzw. an Umweltveränderungen anzupassen
- Componentware (Komponentensoftware)
  - besteht aus wiederverwendbaren, kombinierbaren Funktionseinheiten (Komponenten)
  - Komponenten sind sprachenunabhängig
  - fertige Komponenten stehen in Softwarebibliotheken zur Verfügung
  - Schnelle, flexible Entwicklung und Aktualisierung von Softwaresystemen wird ermöglicht

ifak

Geschichte von COM/DCOM



Nutzung von COM

- COM ist Bestandteil der Windows-Betriebssysteme
- Im Gegensatz zu JavaBeans definiert COM einen binären Standard
  - Keine Vorgabe, wie die Bindung einer konkreten Programmiersprache an COM erfolgen soll
- Unabhängig von Programmiersprachen
- Erzeugung von Objekten
  - Clients können von einer COM-Komponente Objekte erzeugen und diese COM-Objekte benutzen, indem sie einen Zeiger auf eine Schnittstelle der Komponente anfordern
  - Über diesen Zeiger kann der Client dann Operationen der Komponente aufrufen.

ifak

Nutzung von COM (ff)

- COM-Objekt vs. COM-Komponente
  - Literatur : COM-Objekt (COMobject) 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.

ifak

Begriffe

- Die COM-basierte Komponente
  - Kann mehrere COM-Klassen enthalten
  - kann sprachenunabhängig vom Client instantiiert werden

- Eine COM-Klasse implementiert mindestens eine Schnittstelle.
- Eine Schnittstelle in COM besteht aus einer Menge von Methoden, die in einer semantischen Beziehung zueinander stehen.

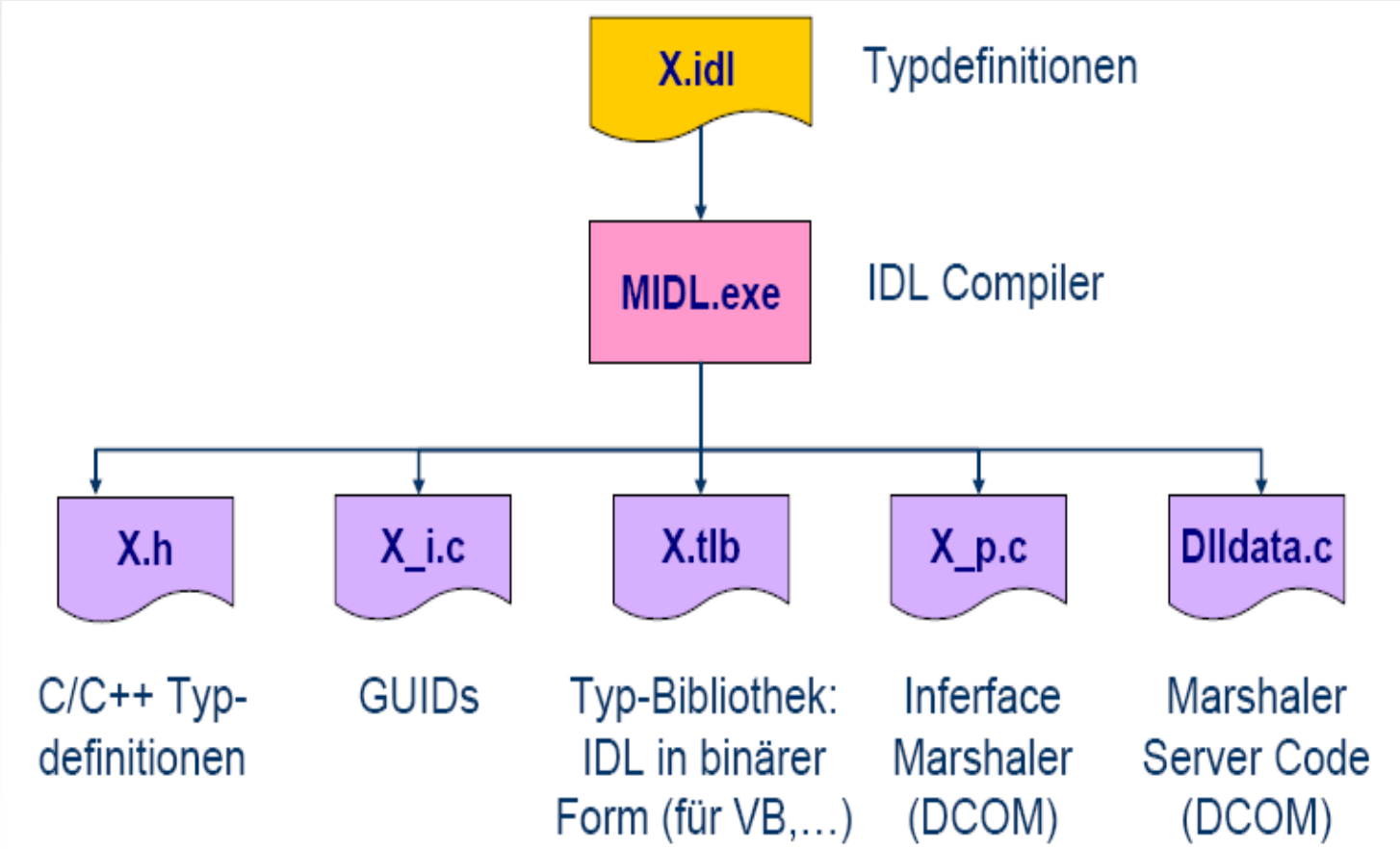
- Die Instanz einer COM-Klasse heißt COM-Objekt.
- Das COM-Objekt
  - bietet seine Funktionen als Methoden für Clients über eine oder mehrere Schnittstellen an
  - kann von Clients über einen Schnittstellenzeiger angesprochen werden

ifak

Interface Definition Language

- Basiert auf IDL von OFS DCE RPC (wie CORBA).
- Beschreibung von Interfaces und Komponenten in C-ähnlicher Syntax.
- Erweiterung um objektorientierte Aspekte (Vererbung, Polymorphismus).
- Gewährleistet Sprach- und Ortsunabhängigkeit.
- Transportprotokoll von DCOM ist Microsofts Implementierung von DCE RPC.

Microsoft IDL Compiler (MIDL)



Beispiel IDL File

```
import "unknwn.idl";

[
  uuid(2A09015D-EB91-452d-B594-AD39665BB52E),
  object,
  helpstring("String Component")
]
interface IString : IUnknown {
  HRESULT Length([out, retval] int* len);
}

[
  uuid(9ACD001E-023D-40a0-A8C9-FD64C2FB8DC2),
  object,
  helpstring("Serialization Interface")
]
interface ISerizalizable : IUnknown {
   HRESULT Load([in] BSTR fileName);
   HRESULT Save([in] BSTR fileName);
}
```

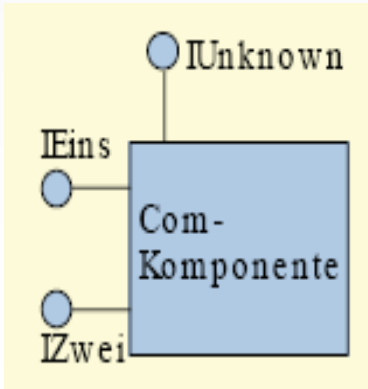
UUIDs und GUIDs

- Weltweit eindeutiger Schlüssel
  - 128 bit: C0F6EE7E-9F71-4252-9A2E-9878A15B2D83
  - Abhängig von Netzwerkadresse und Zeitpunkt der Generierung.
  - UUID= Universally Unique Identifier
    - Bezeichnung in DCE RPC
  - GUID= Globally Unique Identifier
    - Bezeichnung in COM
- Eindeutige Identifizierung von
  - Interfaces (IID)
  - Komponenten (ClassFactories) (CLSID)
  - Typbibliotheken (LIBID)

ifak

IDL Schnittstellen

- COM-Komponenten implementieren i. Allg. nicht nur eine Schnittstelle
- 2 Schnittstellen-Namen
  - Ein textueller Name, der nicht eindeutig ist und per Konvention mit dem Buchstaben »I« beginnt
  - GUID (Global Unique Identifier)
    - 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 InterfaceIdentifier (IIDs) bezeichnet



IDL Methoden

```
HRESULT Methodname([in]shortarg1,
                   [out, retval]int* ret);
```

- HRESULT
  - 32-Bit Fehler-Code: allgemeiner + Methoden-spezifischer Teil.
  - Verwendung in C++:

```
HRESULT hr= pIf->Methodname (i, &r);
if (SUCCEEDED(hr)) { … }
```

  - Kann z.B. auf Exceptions (oder ähnliches) abgebildet werden.
- Directional Attributes
  - **[in]**: Client →Server
  - **[out]**: Server ←Client
  - **[in,out]**: Server ←→ Client
  - **[retval]**: Kann auf Rückgabewert abgebildet werden
    z.B. **VB**: res = obj.**Methodname**(i)

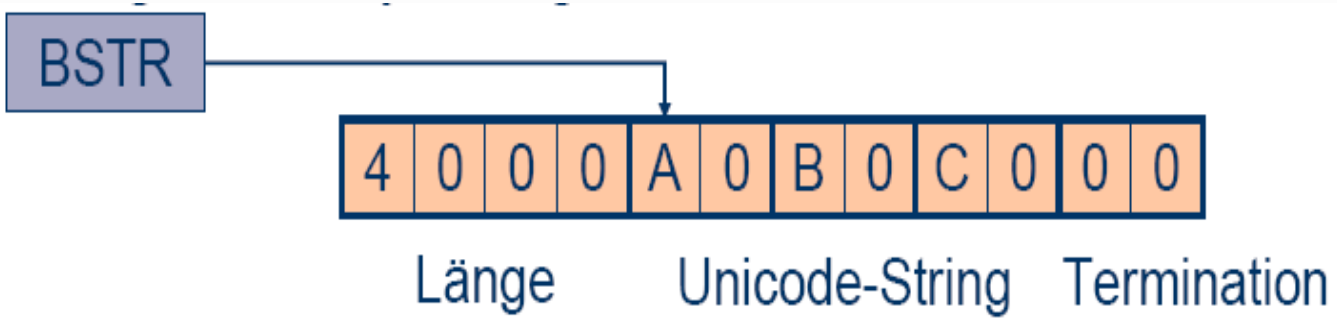
ifak

IDL Datentypen

- Standardtypen:

  ```
  boolean, byte, small, short, long, int, char, wchar_t,
  float, double
  ```

- Enumerationen: C-Syntax
- Strukturen:C-Syntax
- Pointer: C-Syntax
  - Zusätzliche Attribute für effizienteren Marshaling-Code
- Strings: "BinaryStrings"



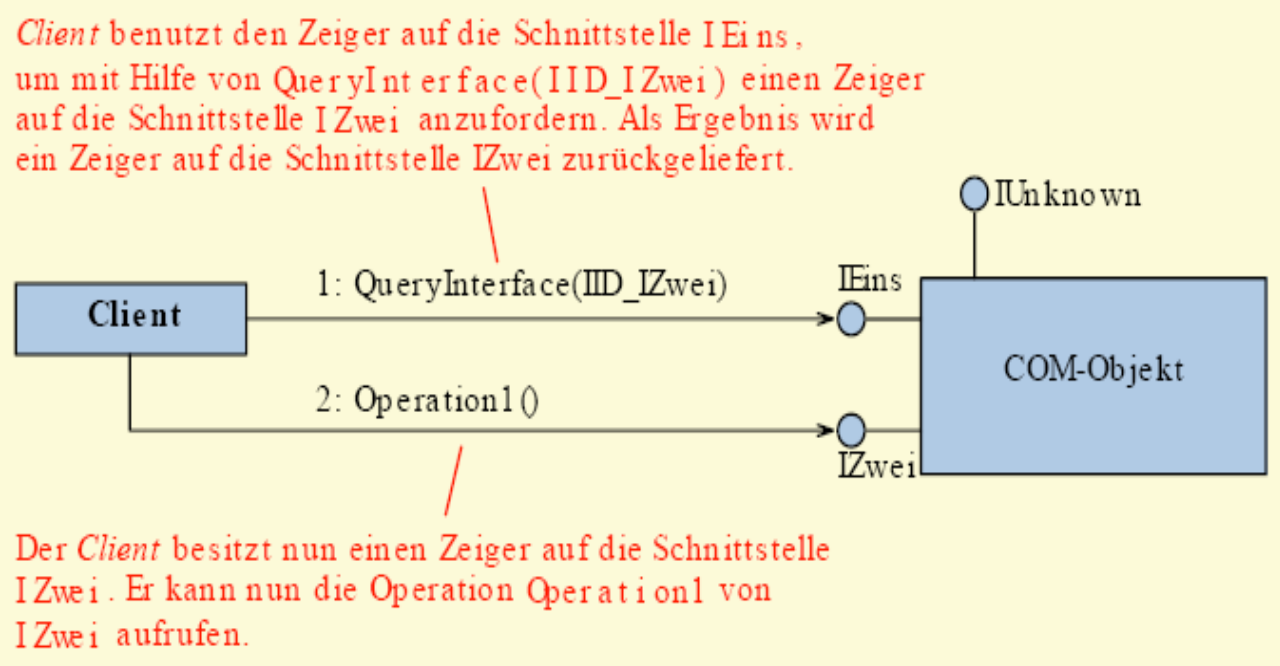
Regeln

- Vererbung von Schnittstellen
  - COM unterstützt die Schnittstellenvererbung
- Schnittstelle IUnknown
  - Es gibt eine von COM vorgegebene Schnittstelle, die jede COM-Komponente implementieren muss
    - Schnittstelle IUnknown
    - Alle benutzerdefinierten Schnittstellen müssen von IUnknown erben
    - IUnkown enthält die Operationen QueryInterface(), AddRef() und Release()
- Methode QueryInterface
  - Erlaubt es, einen Zeiger auf eine der Schnittstellen einer Komponente anzufordern. Dies geschieht, indem QueryInterface die IID einer Schnittstelle als Eingabe übergeben bekommt
  - Alle Standard-Schnittstellen sind von Microsoft dokumentiert

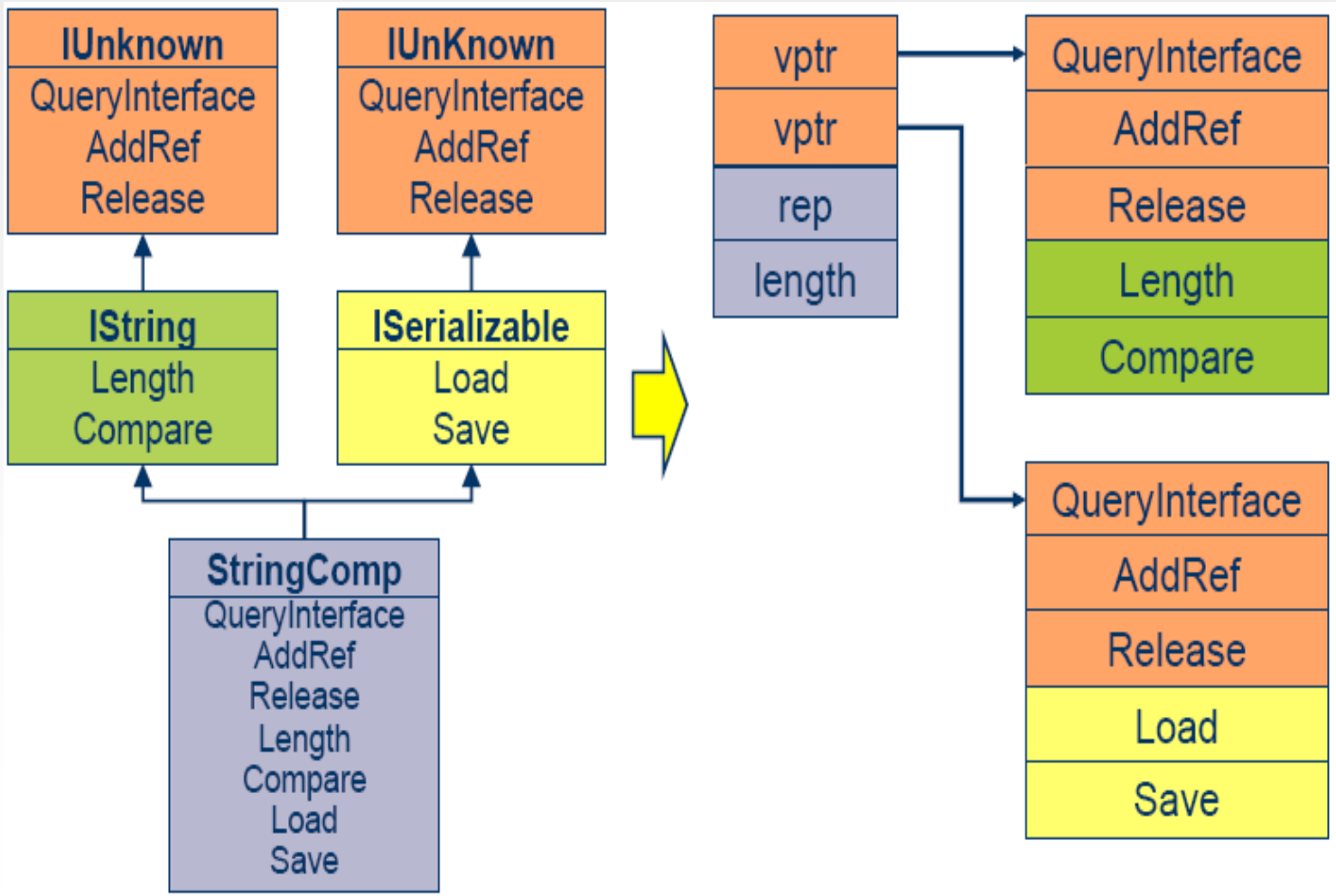
ifak

Regeln (ff)

- Die IID der gewünschten Schnittstelle entnimmt der Entwickler vor dem Aufruf von QueryInterface der zugehörigen Dokumentation
- Als Ergebnis liefert QueryInterface im Erfolgsfall einen gültigen Zeiger auf die angeforderte Schnittstelle
- Im Fehlerfall wird ein Fehlercode zurückgegeben.



Implementierung der Schnittstellen



Implementierung von QueryInterface

```
HRESULT __stdcall QueryInterface(REFIIDriid, void** ppi) {
  if (IID_IUnknown == riid)
    *ppi = static_cast<IString*>(this);
  else if (IID_IString == riid)
    *ppi = static_cast<IString*>(this);
  else if(IID_ISerializable == riid)
    *ppi= static_cast<ISerializable*>(this);
  else {
   *ppi = NULL;
  return E_NOINTERFACE;
  }
  (reinterpret_cast<IUnknown*>(*ppi))->AddRef();
  returnS_OK;
}
```

ifak

IUnknown: Reference Counting

```
class xyz: public Iabc{
  ULONG refCnt;
public:
  xyz() { refCnt= 0; }
  ULONG __stdcall AddRef() { return ++refCnt; }
  ULONG __stdcall Release() {
    if (--refCnt== 0) {
        delete this;
        return 0;
    }
    return refCnt;
 }
}
```

- Client muss mit `AddRef()` und `Release()` die Anzahl der Referenzen auf Komponente verwalten.
- Bei MTA müssen diese Funktionen threadsafe gemacht werden → Verwendung von `InterlockedIncrement` und `InterlockedDekrement`.

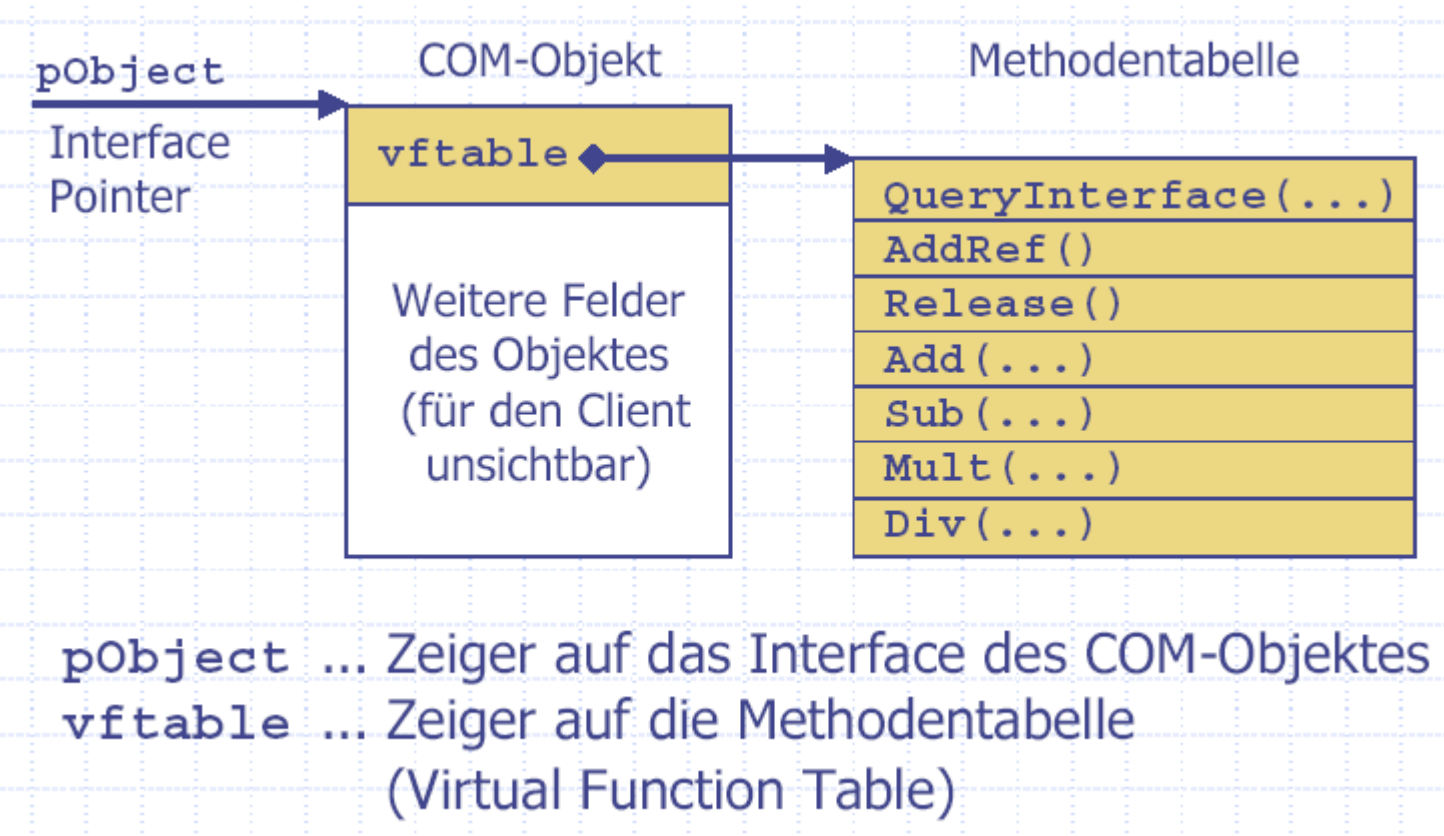
ifak

Implementierung der benutzerdefinierten Schnittstellen

```
class xyz: public Iabc{
  ULONG refCnt;
  int   length;
public:
  xyz() { refCnt= 0; length = 0; }
  HRESULT __stdcall Length(int* len) {
    *len = length;
    return S_OK;
  }
  …
}
```

ifak

Zugriff auf Methoden - Die Tabelle virtueller Funktionen



Die vtbl gibt es pro Klasse nur einmal, unabhängig von der Anzahl der Instanzen.

Deployment (Modultypen)

- Module
  - Auslieferung als DLLs oder Anwendungen (.exe)
  - Jedes Modul: ein oder mehrere COM-Komponenten
- 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.

ifak

Merkmale der Module

- Stück Binärcode
- Kann eine oder mehrere Schnittstellen implementieren
- Können nicht voneinander erben (in OOP-Terminologie)
  - Nur Schnittstellen-Vererbung
- Es kann nur über ihre Schnittstellen kommuniziert werden
- Ein Client kann einen Zeiger auf alle Schnittstellen, die eine Komponente implementiert, mit Hilfe der Operation QueryInterface und der entsprechenden IID erhalten

ifak

COM-Objekte erstellen: ClassFactory

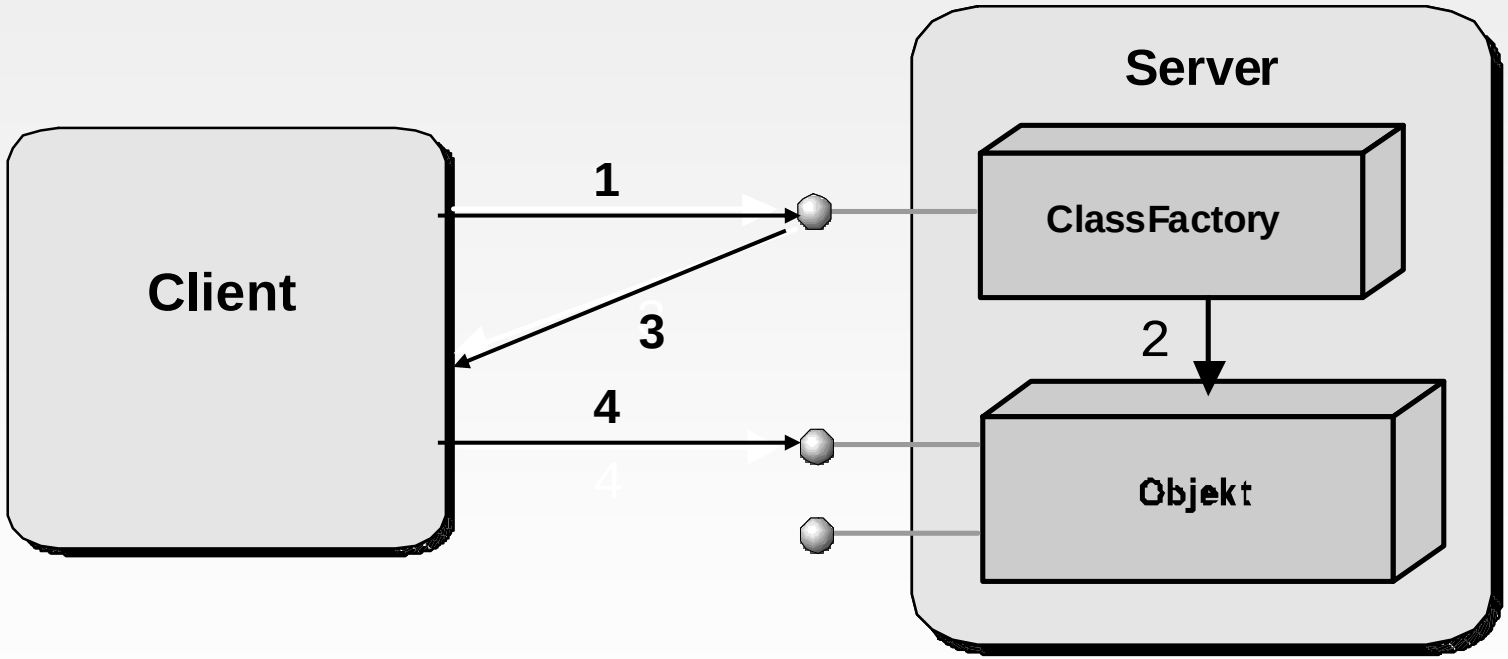
- ClassFactory ist eine Komponente, mit der COM-Objekte erzeugt werden können.
- Muss für jede Komponente implementiert werden.
- Interface:

```
interface IClassFactory: IUnknown {
  HRESULT CreateInstance([in, unique] IUnknown* pUnkOuter,
                         [in] REFIID rrid,
                         [out, iid_is(rrid)] void** ppObj);
  HRESULT LockServer([in] BOOL lock);
}
```

  - CreateInstance: Erzeugen eines COM-Objekts.
  - LockServer: Verhindert frühzeitige Freigabe des COM-Objekts.

ifak

COM-Objekte erstellen: ClassFactory (ff)



Bedeutung der Registry

- Wie erzeugt man COM-Objekte und wie gelangt man an den ersten Schnittstellenzeiger?
  - Die CLSID einer Komponente wird zusammen mit dem Aufenthaltsort des Moduls, das sie implementiert, in der Windows Registry abgelegt
  - Eine COM-Komponente muss registriert werden
  - Im Zweig HKEY_CLASSES_ROOT/CLSID hinterlegt jede Komponente als Schlüssel ihre CLSID
    - Dieser Schlüssel enthält als Wert den Namen der Komponente
  - Ein Unterschlüssel enthält den Dateinamen des Moduls, welches die Komponente implementiert
    - Der Name des Schlüssels ist InprocServer32 bei DLL-Modulen, LocalServer32 bei Exe-Modulen.

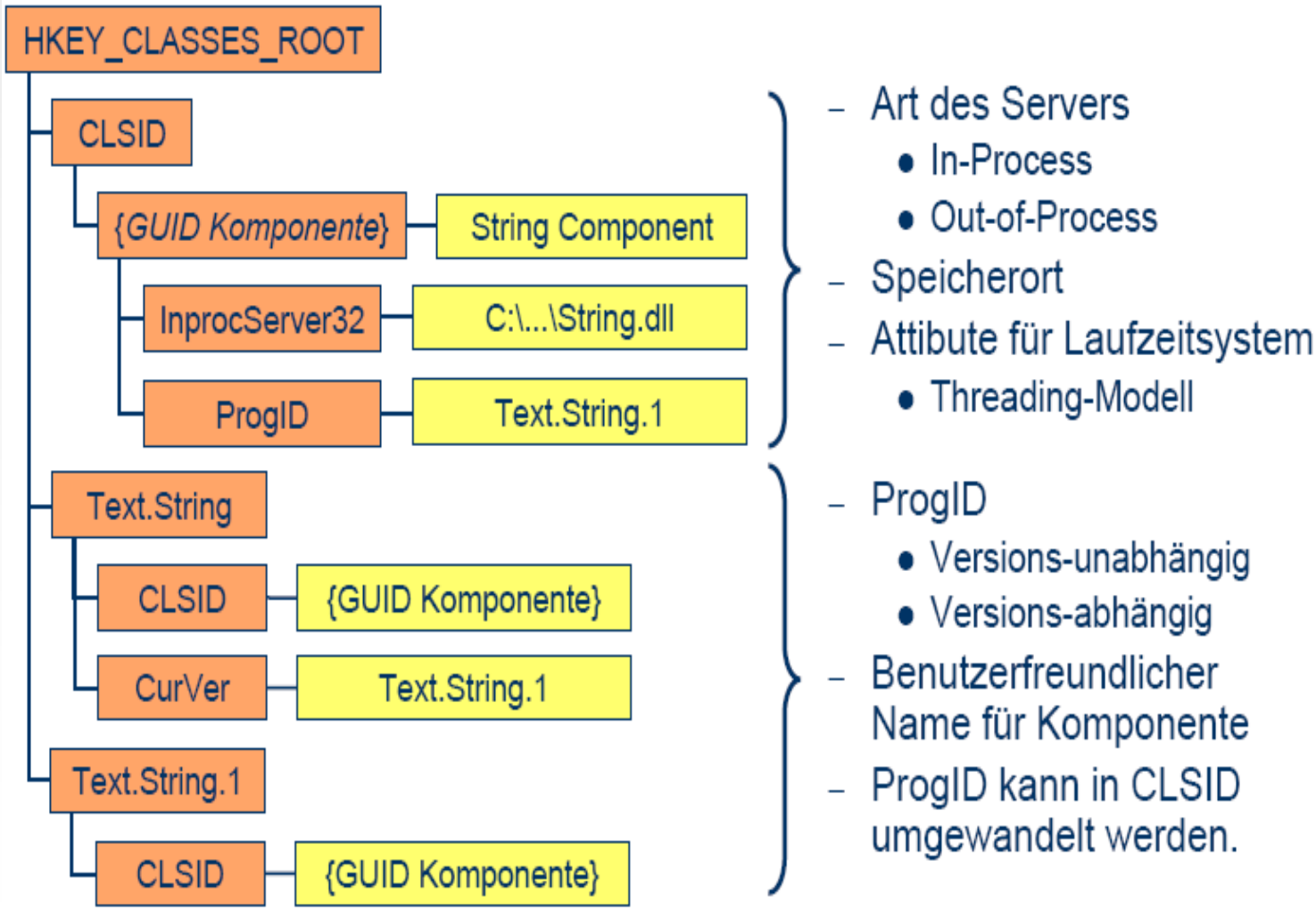
ifak

Bedeutung der Registry (ff)

- Zweig HKEY_CLASSES_ROOT
  - Jede Komponente trägt zusätzlich als Schlüssel ihren Namen ein. Dies kann zu Konflikten führen:
    - Haben 2 Komponenten den gleichen Namen, dann überschreibt die zuletzt registrierte Komponente den Schlüssel der anderen Komponente
    - Beispiel: HKEY_CLASSES_ROOT/Excel.Application
    - Ein Unterschlüssel namens CLSID enthält als Wert die CLSID der Komponente
- Mit den Informationen aus der Registry können die COM-Bibliotheken Komponenten zur Verfügung stellen.

ifak

Einträge in der Windows Registry



Schnittstelle einer COM-DLL

- Jede COM-DLL kann mehrere Komponenten enthalten.
- COM-DLL muss folgendes Interface aufweisen:

```
LIBRARY “xyz"
EXPORTS
  DllGetClassObject@1 PRIVATE
  DllCanUnloadNow@2 PRIVATE
```

  - DllGetClassObject: Instanziert eine ClassFactory und gibt eine Referenz darauf zurück.
  - DllCanUnloadNow: Gibt zurück, ob noch Referenzen auf COM-Objekte oder Locks auf Komponente existieren.

ifak

Selbstregistrierung einer COM-DLL

- Schnittstelle der COM-DLL kann um zwei Funktionen für Selbstregistrierung erweitert werden:

```
LIBRARY „xyz"
EXPORTS
  DllGetClassObject @1 PRIVATE
  DllCanUnloadNow @2 PRIVATE
  DllRegisterServer @3 PRIVATE
  DllUnregisterServer @4 PRIVATE
```

- Registrierung der Komponente

```
regsvr32 <Pfad>\xyz.dll
```

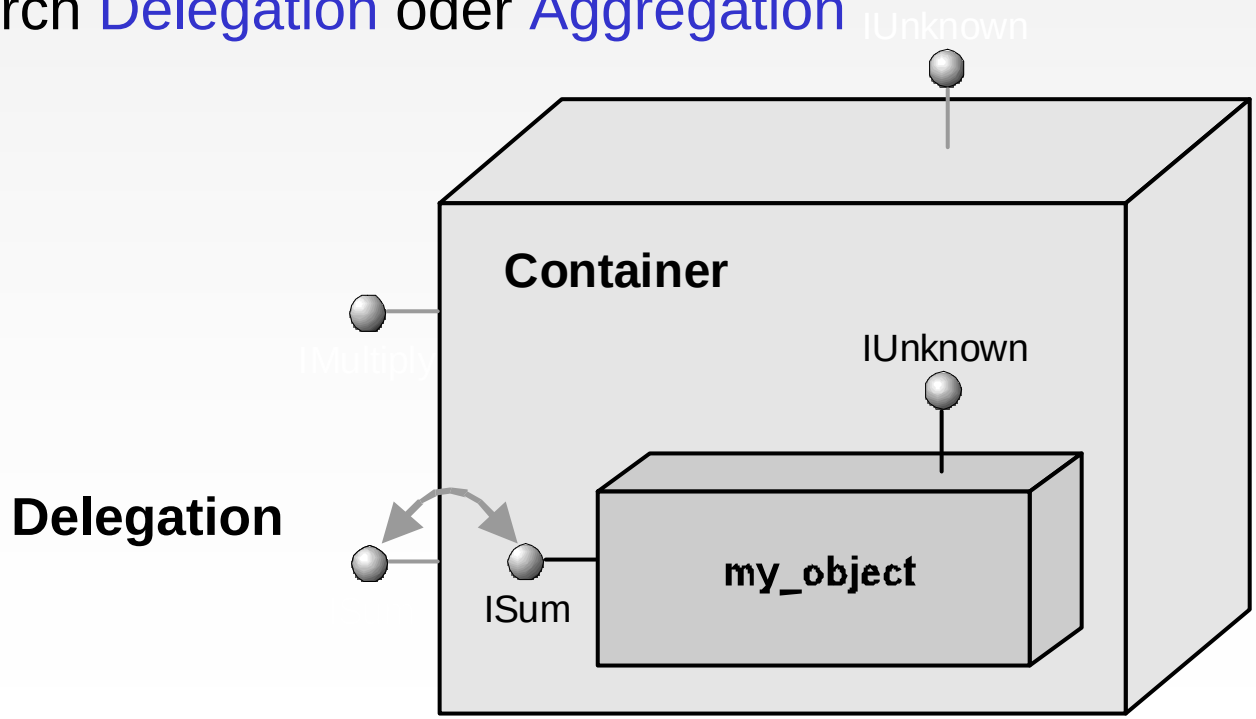
- Deregistrierung der Komponente

```
regsvr32 /u <Pfad>\xyz.dll
```

ifak

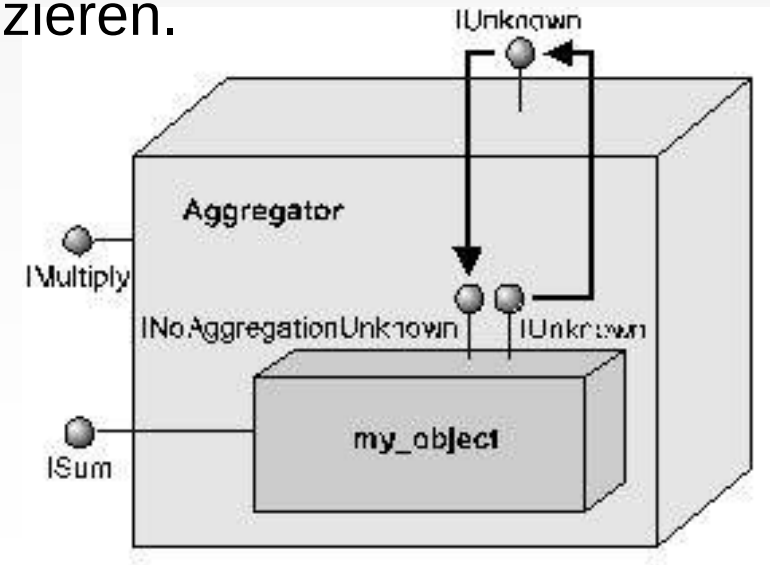
Wiederverwendungsmechanismen
Delegation

- Schnittstellenvererbung statt Quellcodevererbung
- Quellcodevererbung ist nur innerhalb von Komponenten möglich
- Wiederverwendung von COM-Objekten erfolgt zur Laufzeit durch Delegation oder Aggregation



Wiederverwendungsmechanismen
Aggregation

- Mechanismus kann nur zwischen prozessinternen Komponenten eingesetzt werden
- Schnittstellen des internen Objekts werden tatsächlich als Schnittstellen des äußeren Aggregators offengelegt
- Im Gegensatz zur Delegation kann der Client also direkt mit dem internen Objekt kommunizieren.

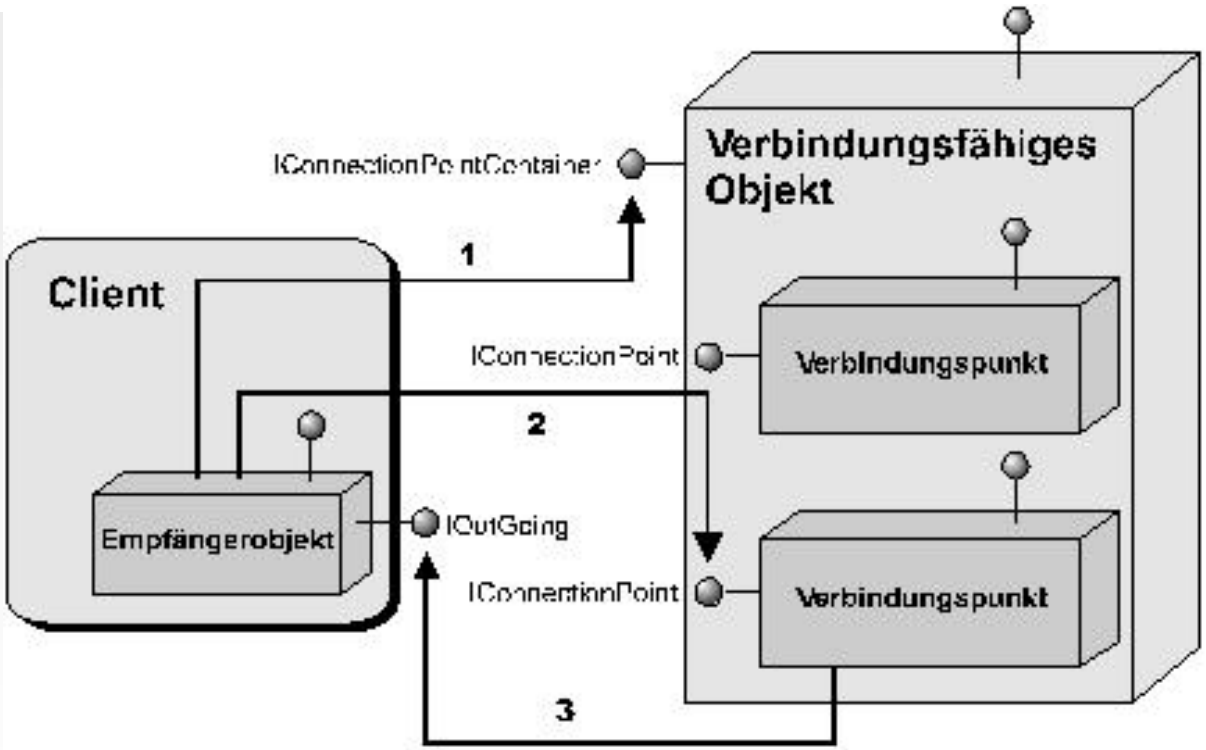


Verbindungsfähige Objekte

- Verbindungsfähige Objekte sind in der Lage, den Client über einen vom Objekt selbst initiierten Aufruf anzusprechen. Hierzu ist ein empfangendes Objekt innerhalb des Client nötig, das die Aufrufe des verbindungsfähigen Objekts entgegennimmt.
- Dadurch sind Komponenten in der Lage, Funktionen im Client unabhängig von ihm zu steuern.
- Notwendige Schnittstellen: IConnectionPointContainer, IConnectionPoint
- Der bekannteste Typ verbindungsfähiger Objekte ist das ActiveX-Steuerelement

ifak

Verbindungsfähige Objekte (ff)



DCOM

- Netzwerkprotokoll höherer Ebene, das es COM-basierten Komponenten ermöglicht, über ein Netzwerk zusammenzuarbeiten
- unterstützt neben TCP/IP und HTTP viele weitere Protokolle
- Stellung von DCOM in einem Protokollstapel im Vergleich zum OSI -Schichtenmodell:

Anwendung	DCOM / RPC
Darstellung	
Sitzung	Winsock-Treiber
Transport	UDP
Netzwerk	IP
Sicherung	Ethernet-Treiber
Physisch	Ethernet-Karte

- DCOM ist kein eigenständiges Protokoll, das in einer höheren Schicht über RPC steht
- benutzt RPC und verändert die PDU vom RPC-Protokoll durch eigene Daten

ifak

DCOM (ff)

- ist verbindungsorientiert
- garantiert Datenreihenfolge
- sorgt nach einer Remote-Instantiierung für die Übermittlung einer Objektexporteurkennung (OXID) an den Client

- OXID-Resolver
  - Dienst von DCOM, ist auf jedem DCOM-Rechner installiert
  - speichert und übersetzt OXIDs (bezeichnen RPC-Verbindungszeichenfolgen) und stellt sie lokalen Clients zur Verfügung
  - sendet für jeden Client in regelmäßigen Abständen Ping-Nachrichten an alle Server im Netz, damit Komponenten einen Ausfall bemerken und aus dem Speicher entfernt werden können

  → verteilter Garbage-Collector !

ifak

COM+

- Weiterentwicklung von COM
- wird zusätzliche Dienste bieten (z.B. Implementierungsvererbung oder Abfangen von Methodenaufrufen)
- das COM+-Laufzeitmodul wird Standardimplementierungen der Standard-COM-Schnittstellen anbieten (sind in J++ und Visual Basic bereits integriert, deshalb größerer Nutzen für C++-Programmierer)
- definiert Standarddatentypen mit Speicherdarstellung in der COM+-Spezifikation als weitere Unterstützung der Interoperabilität (wichtig bei Erstellung von Komponenten in verschiedenen Programmiersprachen)

ifak