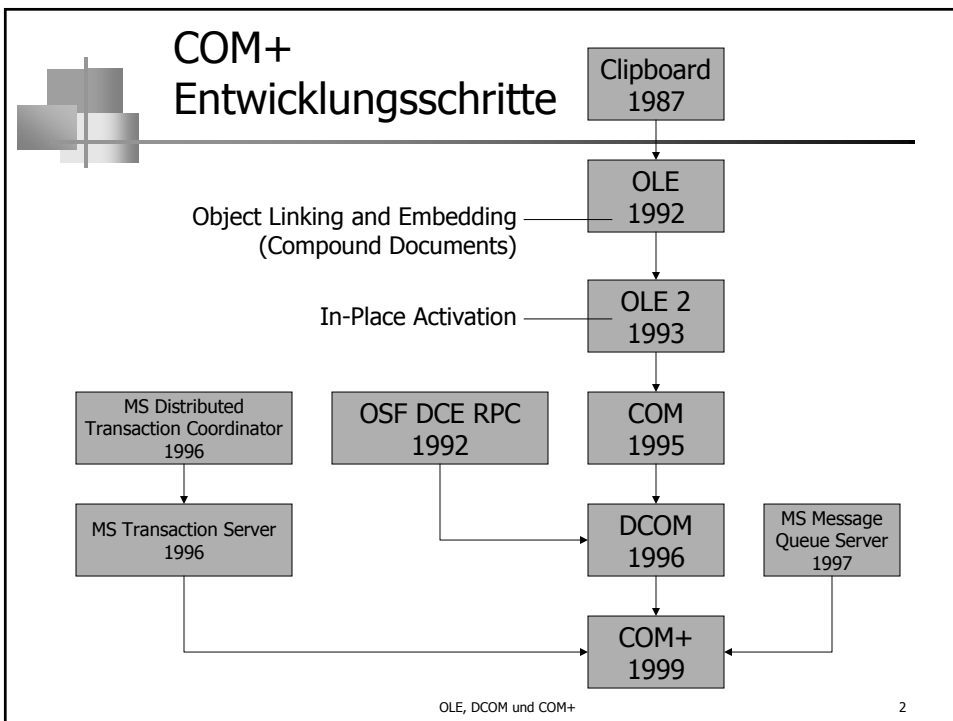


# OLE, DCOM und COM

Die frühen Komponentenmodelle der Firma Microsoft



## Windows DNA

- DNA = Distributed interNet Application Architecture
- 3-Tier Ansatz

OLE, DCOM und COM+


3

## Component Services

- COM
  - Interface-based Programming
  - Basic Component Facilities
- DCOM
  - Remoting Architecture
  - Distributed Component Services
- COM+
  - Load Balancing
  - In-memory Database
  - Object Pooling
  - Queued Components
  - Event Model
- MTS
  - Transaction Services
  - Resource Pooling
  - Role-based Security
  - Administration
  - Just-in-time Activation

OLE, DCOM und COM+

4




## Basics

---

- Binärer Komponentenstandard
  - Implementierungssprache unbestimmt
- Komponente realisieren Interfaces
- Interface = Schnittstellenbeschreibung
  - Abstrakte C++-Klasse mit virtuellen Funktionen
  - vtbl definiert Schnittstelle
- Minimal
  - **IUnknown**: Elementarmethoden jeder Komponente
  - **IClassFactory**: Komponentenerzeugung

OLE, DCOM und COM+ 5




## Identifikation von Komponenten

---

- Interfaces besitzen eine GUID
- 128 Bit „Zufallszahl“
  - z.B. IUnknown {00000000-0000-0000-C000-000000000046}
  - Erzeugung über API: `CoCreateGuid()`
  - Erzeugungstool: guidgen.exe
- Registry = Namensdienst

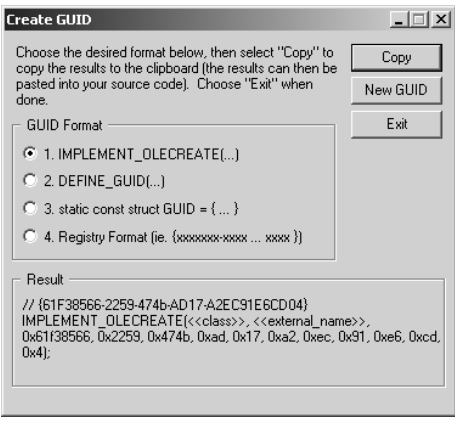
OLE, DCOM und COM+ 6




## guidgen.exe

---

- Erzeugung eigener GUIDs
  - Hinreichend zufällig
- Bestandteile
  - Aktuelles Datum
  - Aktuelle Uhrzeit
  - Fortlaufende „Clock Sequence“ (persistent)
  - Inkrementeller Zähler (hochfrequente Abfragen)
  - MAC-Adresse der Netz Karte (geht aber auch ohne)



OLE, DCOM und COM+
7



## IUnknown

---

```


interface IUnknown
{
    typedef [unique] IUnknown *LPUNKOWN;

    HRESULT QueryInterface (
        [in] REFIID riid,
        [out, iid_is(riid)] void **ppvObject );

    ULONG AddRef ();

    ULONG Release ();
}
  
```

OLE, DCOM und COM+
8



## IClassFactory


---

```
interface IClassFactory: IUnknown
{
    HRESULT CreateInstance (
        [in, unique] IUnknown *pUnkOuter,
        [in] REFIID riid,
        [out, iid_is(riid)] void **ppvObject);

    HRESULT LockServer ( [in] BOOL flock );
}
```


OLE, DCOM und COM+

9



---

Erste COM-Schritte




## Beispiel

---

- Aufgabe
  - Übertragung von n ganzen Zahlen
  - „Komponente“ bildet die Summe
- Schritte
  1. Interface definieren
  2. Header-Dateien generieren
  3. Komponente implementieren
  4. Client programmieren
  5. Registry-Eintrag

OLE, DCOM und COM+ 11



## 1. Interface

---

- Utility-Projekt in Visual Studio
- Interface Isum.idl:

```
import "unknwn.idl";

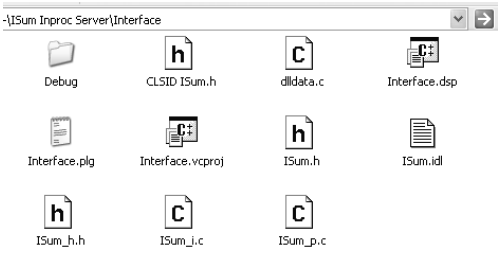
[ object,
  uuid(1E10C200-E306-11D3-A557-B6EEE489CA00)
]

interface ISum : IUnknown {
    HRESULT AddElements (
        [in] int nElem,
        [in, size_is(nElem)] int *elements,
        [out] int *sum
    );
};
```

OLE, DCOM und COM+ 12

## 2. MIDL ausführen

- Mehrere Dateien werden generiert
- Bedeutung:
  - C++-Version des Interfaces
  - GUID-Definitionen
  - Proxy-Stubs für entfernte Server




OLE, DCOM und COM+

13

## 3. Komponente implementieren

- Jedes Interface der Komponente muß implementiert werden
  - Ableitungen von den abstrakten Basisklassen
  - „Pure virtual functions“ implementieren
- Virtuelle Funktionen müssen immer wieder implementiert werden (Binärstandard)



OLE, DCOM und COM+

14

## 4. Client programmieren

- Initialisierung des Komponentenzugriffs
- **Unknown**-Interface über GUID erfragen
- Downcast zum gewünschten Interface
- Komponente benutzen
- Explizites Reference-Counting beachten

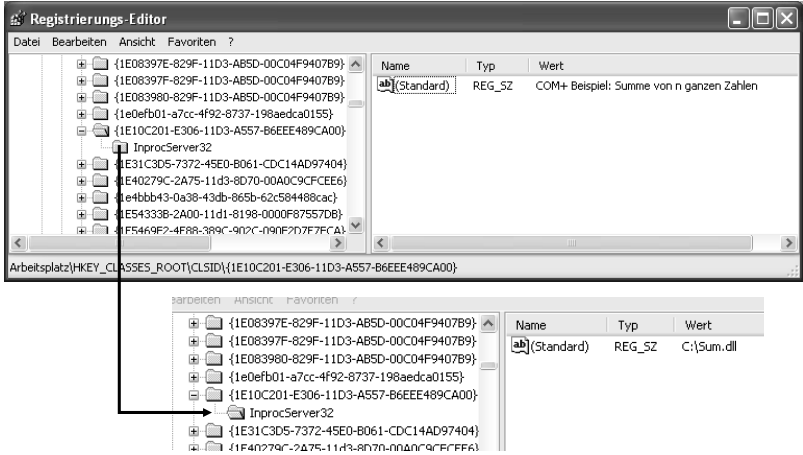
Client.cpp

OLE, DCOM und COM+

15

## 5. Registry-Eintrag

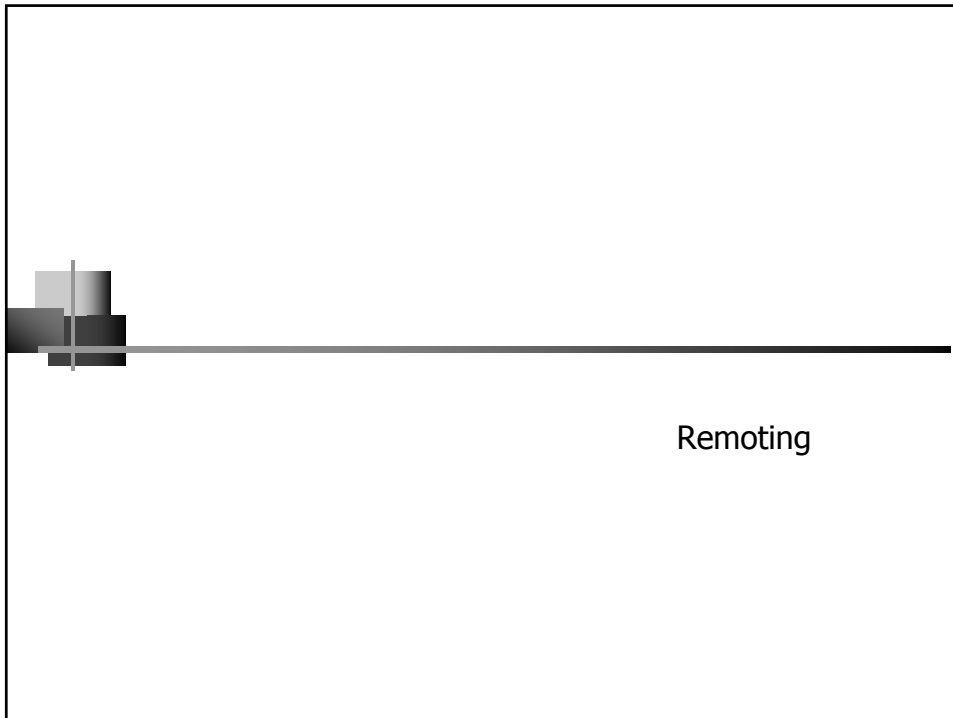
- **HKCR/CLSID/{GUID}**



OLE, DCOM und COM+

16






## Architekturen

- Inproc-Server
  - Komponente als DLL im Client
  - Funktionsaufrufe
  - Hohe Performance
- Server
  - Komponente als EXE auf Client-Rechner
  - Zugriff über Proxy-Objekte
  - Surrogate: Spezifische DLL-Container
  - COM-Laufzeitumgebung: MTS
- Remote Server
  - Komponente auf entferntem Rechner (meist in lokalem MTS)
  - Zugriff über Proxy-Objekte
  - Marshalling, RPC

OLE, DCOM und COM+ 18




## ISum: Local Server

---

- Expliziter Aufruf eines lokalen Servers:
  - `hr = CoCreateInstance(CLSID_ISum, NULL, CLSCTX_LOCAL_SERVER, IID_IUnknown, (void **)&pUnknown);`
  - statt
  - `hr = CoCreateInstance(CLSID_ISum, NULL, CLSCTX_INPROC_SERVER, IID_IUnknown, (void **)&pUnknown);`
- Es geht aber auch transparent
  - Zuerst nach Inproc-Server schauen
  - Dann nach lokalem Server
  - Dann ggf. entfernter Server

OLE, DCOM und COM+ 19




## Wir verlassen die Prozeßgrenzen

---

- RPC = Remote Procedure Call
  - Argumente in Nachricht verpacken
  - Nachrichtenübertragung an den Server
  - Auf Antwortnachricht warten
  - Ergebnis an den Aufrufer zurückgeben
- Marshalling und Unmarshalling
  - MIDL generiert die notwendigen Routinen automatisch
  - Proxy = Client-Seite
  - Stub = Server-Seite
- Verankerung der Proxy- und Stubroutinen in Registry

OLE, DCOM und COM+ 20




## Im Beispiel

---

- Relevante GUIDs für das Beispiel:
- Interface ISum  
10000000-0042-0000-0000-000000000001
- Komponente CoSum (CLSID)  
10000000-0042-0000-0000-000000000002

OLE, DCOM und COM+ 21



## Zusätzlicher Schritt

---

- ProxyStub-DLL erzeugen
  - Benötigt werden die von MIDL erzeugten Dateien:
    - dlldata.c
    - ISum\_i.c
    - ISum\_p.c

OLE, DCOM und COM+ 22

## ProxyStub-DLL registrieren

The top screenshot shows the Registry Editor with the path `HKEY_CLASSES_ROOT\Interface\{10000000-0042-0000-0000-000000000001}\ProxyStubClsid32`. The right pane shows a single registry value:

Name	Typ	Wert
(Standard)	REG_SZ	{10000000-0042-0000-0000-000000000001}

The bottom screenshot shows the path `HKEY_CLASSES_ROOT\CLSID\{10000000-0042-0000-0000-000000000001}\InProcServer32`. The right pane shows two registry values:

Name	Typ	Wert
(Standard)	REG_SZ	e:\Peter\Teach\Components\COM+1\Sum Local Server\Pr...
ThreadingModel	REG_SZ	Both

OLE, DCOM und COM+ 23

## Registryeinträge für lokalen Server

The top screenshot shows the Registry Editor with the path `HKEY_CLASSES_ROOT\CLSID\{10000000-0042-0000-0000-000000000002}\InProcServer32`. The right pane shows two registry values:

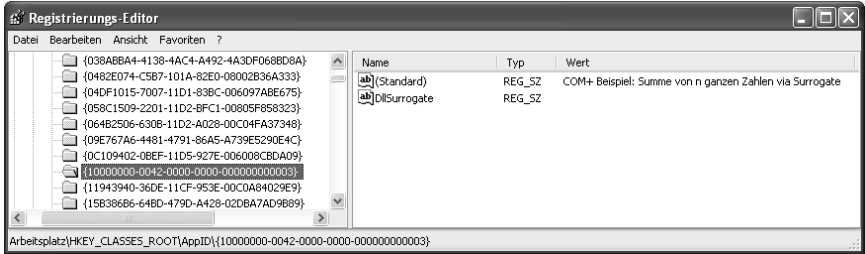
Name	Typ	Wert
(Standard)	REG_SZ	(Wert nicht gesetzt)
AppID	REG_SZ	{10000000-0042-0000-0000-000000000003}

The bottom screenshot shows the path `HKEY_CLASSES_ROOT\CLSID\{10000000-0042-0000-0000-000000000002}\InProcServer32`. The right pane shows a single registry value:

Name	Typ	Wert
(Standard)	REG_SZ	C:\Sum.dll

OLE, DCOM und COM+ 24

## ... und AppID



The screenshot shows the Windows Registry Editor window titled "Registrierungs-Editor". The left pane shows a tree view of the registry, with the path "Arbeitsplatz\HKEY\_CLASSES\_ROOT\AppID\{10000000-0042-0000-0000-000000000003}" selected. The right pane shows a table of registry values:

Name	Typ	Wert
(Standard)	REG_SZ	COM+ Beispiel: Summe von n ganzen Zahlen via Surrogate
DllSurrogate	REG_SZ	

OLE, DCOM und COM+

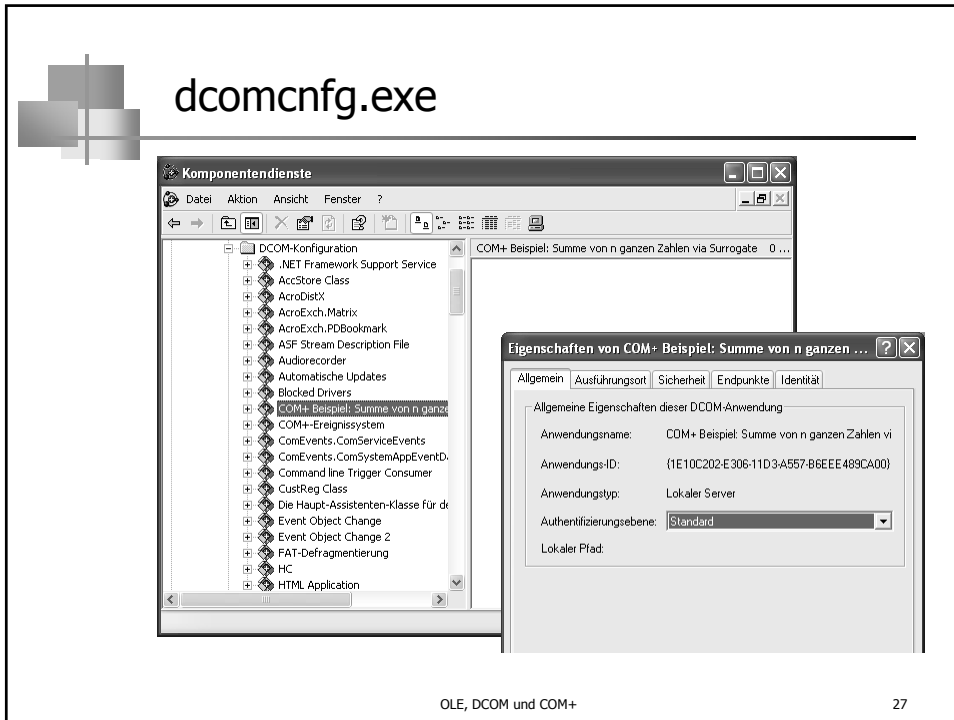
25

## Mögliche AppID-Schlüssel

- AccessPermission
- ActivateAtStorage
- AuthenticationLevel
- DllSurrogate
- LaunchPermission
- LocalService
- RemoteServerName
- RunAs
- ServiceParameter (bei gesetztem LocalService)

OLE, DCOM und COM+

26



**dcomcnfg.exe**

Komponentendienste

- DCOM-Konfiguration
  - .NET Framework Support Service
  - AccStore Class
  - AcroDistX
  - AcroExch.Matrix
  - AcroExch.PDBookmark
  - ASF Stream Description File
  - Audiorecorder
  - Automatische Updates
  - Blocked Drivers
  - COM+ Beispiel: Summe von n ganzen Zahlen
  - COM+-Ereignissystem
  - ComEvents.ComServiceEvents
  - ComEvents.ComSystemAppEventD.
  - Command line Trigger Consumer
  - CustReg Class
  - Die Haupt-Assistenten-Klasse für d
  - Event Object Change
  - Event Object Change 2
  - FAT-Defragmentierung
  - HC
  - HTML Application

Eigenschaften von COM+ Beispiel: Summe von n ganzen ...

Allgemein | Ausführungsort | Sicherheit | Endpunkte | Identität

Allgemeine Eigenschaften dieser DCOM-Anwendung

Anwendungsname: COM+ Beispiel: Summe von n ganzen Zahlen vi

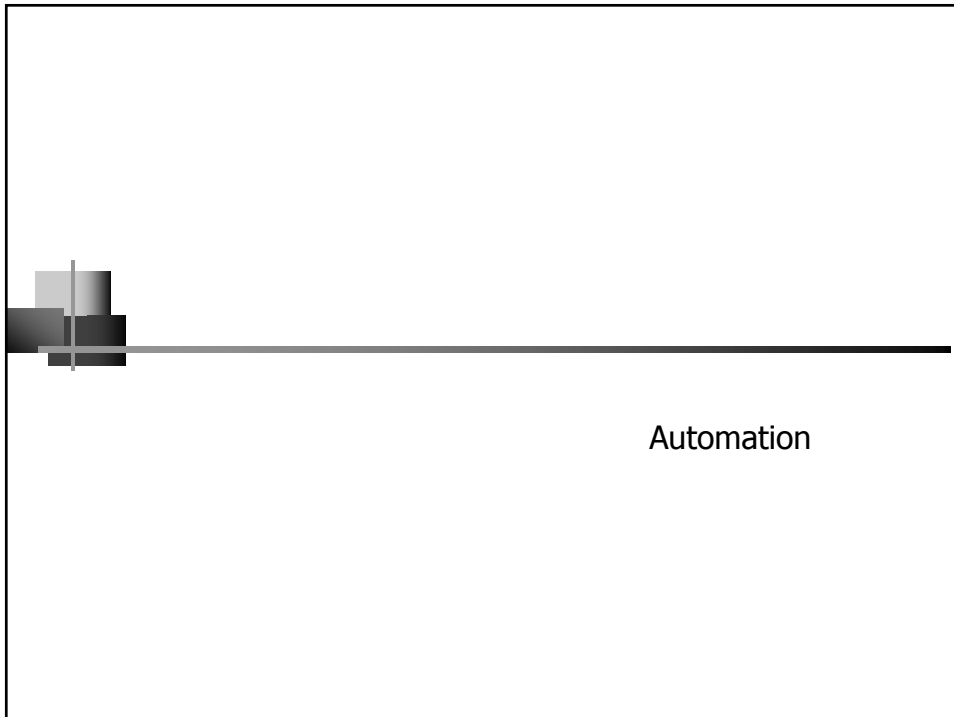
Anwendungs-ID: {1E10C202-E306-11D3-A557-B6EEE489CA00}

Anwendungstyp: Lokaler Server


Authentifizierungsebene: Standard

Lokaler Pfad:

OLE, DCOM und COM+ 27



Automation




## Interfacevarianten

---

- **Statischer Zugriff**
  - Interface der gewünschten Komponente bekannt
  - Zugriff über generierte Schnittstellenbeschreibung
  - Überprüfungen zur Übersetzungszeit
- **Dynamischer Zugriff**
  - Interfacebeschreibung liegt in Form einer Type Library vor
  - Erfragen der Schnittstelle
  - Dynamischer Methodenaufruf
  - Keine Überprüfungen zur Übersetzungszeit
  - Vgl. Introspection (Java), Dynamic Interfaces (Corba)
  - Beim Zugriff über VB, VBA oder Java notwendig

OLE, DCOM und COM+ 29




## Dynamische Interfaces

---

- Automation
- Schnittstelle **IDispatch**
- Kombination aus statischem und dynamischem Interface möglich

OLE, DCOM und COM+ 30



## IDispatch

---

```

[
    object,
    uuid(00020400-0000-0000-C000-000000000046),
    pointer_default(unique)
]

interface IDispatch : IUnknown
{
    typedef [unique] IDispatch * LPDISPATCH;


    HRESULT GetTypeInfoCount(
        [out] UINT * pctinfo );

    HRESULT GetTypeInfo(
        [in] UINT iTInfo,
        [in] LCID lcid,
        [out] ITypeInfo ** ppTInfo);

    ...
}

```

OLE, DCOM und COM+ 31



## IDispatch (2)

---

```


    HRESULT GetIDsOfNames(
        [in] REFIID riid,
        [in, size_is(cNames)] LPOLESTR * rgszNames,
        [in] UINT cNames,
        [in] LCID lcid,
        [out, size_is(cNames)] DISPID * rgDispId );

[local]
    HRESULT Invoke(
        [in] DISPID dispIdMember,
        [in] REFIID riid,
        [in] LCID lcid,
        [in] WORD wFlags,
        [in, out] DISPPARAMS * pDispParams,
        [out] VARIANT * pVarResult,
        [out] EXCEPINFO * pExcepInfo,
        [out] UINT * puArgErr );

```

OLE, DCOM und COM+ 32






## IDispatch (3)

---

```
[call_as(Invoke)]
    HRESULT RemoteInvoke(
        [in] DISPID dispIdMember,
        [in] REFIID riid,
        [in] LCID lcid,
        [in] DWORD dwFlags,
        [in] DISPPARAMS * pDispParams,
        [out] VARIANT * pVarResult,
        [out] EXCEPINFO * pExcepInfo,
        [out] UINT * pArgErr,
        [in] UINT cVarRef,
        [in, size_is(cVarRef)] UINT * rgVarRefIdx,
        [in, out, size_is(cVarRef)]
            VARIANTARG * rgVarRef
    );
}
```

OLE, DCOM und COM+ 33



---

Extras

## Connection Points

- Rückrufbare Client-Objekte
- Connectable Objects

OLE, DCOM und COM+

35

## Moniker

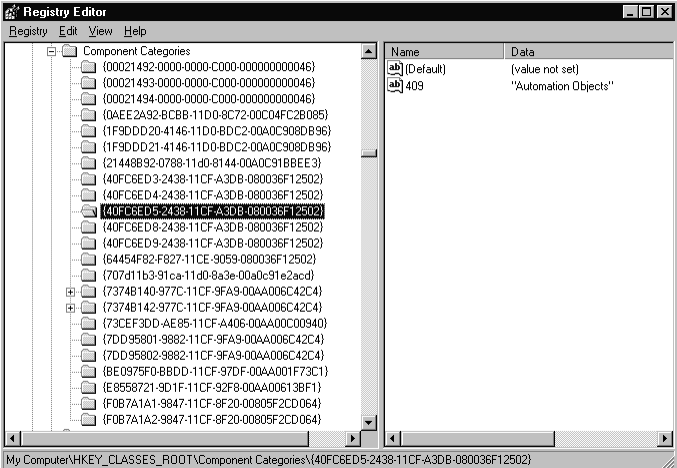
- Moniker benennen andere Objekte
- Varianten
  - FileMoniker - Wrapper für Pfadname einer Datei
  - ItemMoniker - Identifiziert Objekt innerhalb eines Objekts
  - PointerMoniker - Verweis auf ein aktives Objekt
  - AntiMoniker
  - CompositeMoniker
  - ClassMoniker - Wrapper für CLSID einer COM-Klasse
  - URL Moniker
  - OBJREF Moniker - Kapselt marshaled IUnknown Interface Pointer auf ein bestimmtes Objekt

OLE, DCOM und COM+

36

## Component Categories

- Category Identifier (CATID)



The screenshot shows the Windows Registry Editor window. The left pane displays a tree view of the registry, with the path `My Computer\HKEY_CLASSES_ROOT\Component Categories\{40FC6ED5-2438-11CF-A3D8-080036F12502}` selected. The right pane shows the details for this registry value, with the Name field containing '409' and the Data field containing '"Automation Objects"'. The status bar at the bottom of the window confirms the selected path.

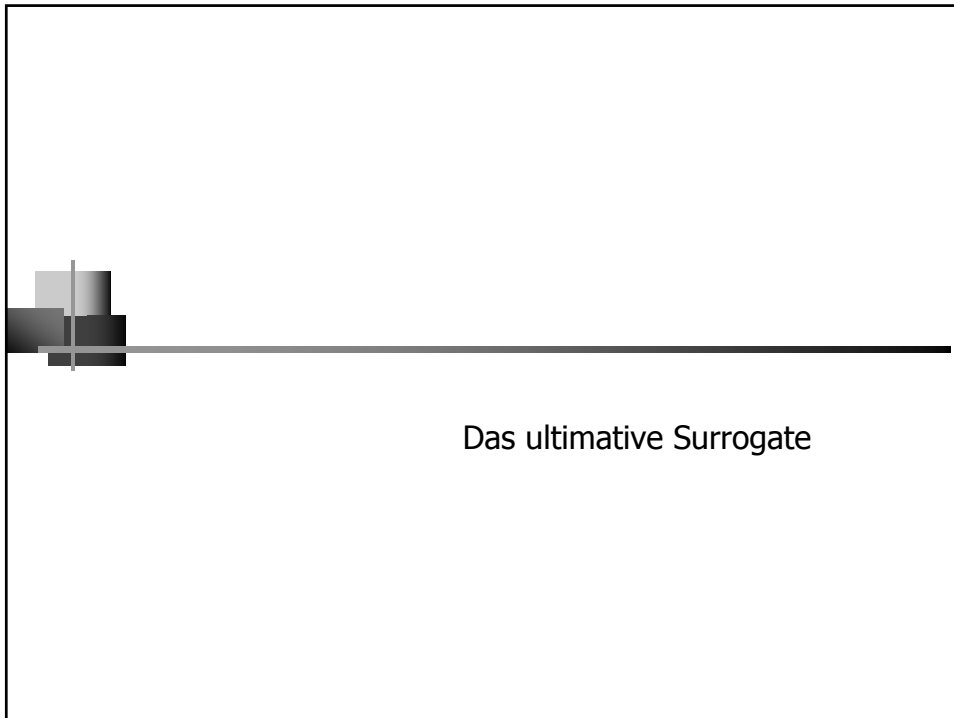
37

## Security

- ACL innerhalb des Systems (vgl. Windows NT)
- Im Verteilungsfall in Anlehnung an OSF DCE
- Authentizitätsprüfung, wenn
  - 0: Default (Systemdefinierter Standard 1-6)
  - 1: None
  - 2: Connect
  - 3: Call
  - 4: Packet
  - 5: Packet Integrity: Test auf Paketveränderung
  - 6: Packet Privacy: Inkl. Verschlüsselung

OLE, DCOM und COM+

38



## MTS

- COM+-Laufzeitumgebung (vgl. EJB Server)
- MTS = „Ultimate Surrogate“
- Wesentliche Funktionen
  - Just-in-time Aktivierung
  - Resource Pooling
  - Transaktionen
  - Persistenz
  - Administration
  - Security

Client

mtb.exe

MTS object

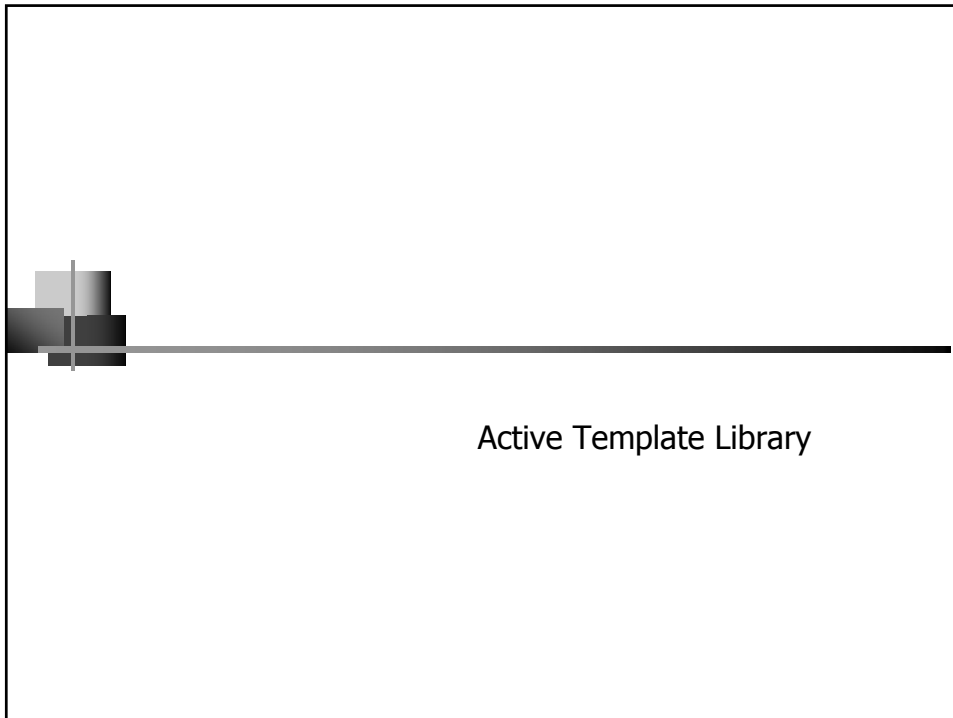
MTS object

MTS object

MTS Executive (mtbex.dll)

OLE, DCOM und COM+

40



## ATL

- Active Template Library
  - Teil der MFC
  - Basiert auf STL
- Grundfunktionen
  - Implementierungen für
    - **IUnknown**
    - **IClassFactory**
    - **IDispatch**
  - Komponentenregistrierung
- DLL oder EXE möglich

**Neues Projekt**

Projekttypen: Visual Basic-Projekte, Visual C#-Projekte, Visual C++-Projekte, Setup- und Weitergabeprojekte, Andere Projekte, Visual Studio-Projektmappen

Vorlagen: Assistent für erweiterte..., ATL-Projekt, ATL-Serverprojekt, ATL-Serverwebedienst

Ein Projekt, das ATL (Active Template Library) verwendet.

Name: Fibonacci Server

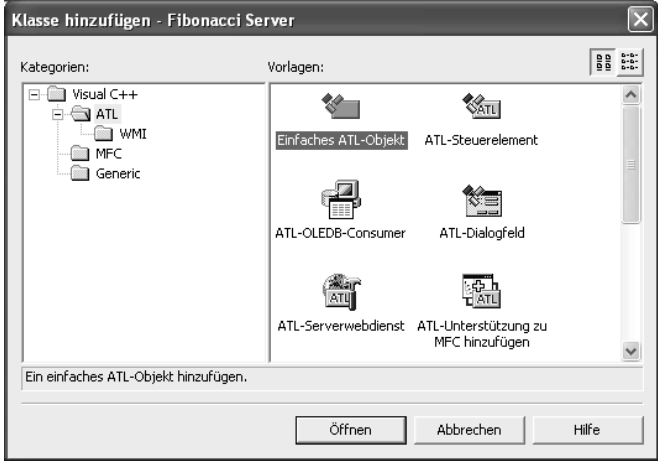
Speicherort: E:\Peter\Teach\Components\COM+ Durchsuchen...

Das Projekt wird erstellt in: E:\Peter\Teach\Components\COM+\Fibonacci Server.

OLE, DCOM und COM+

42


## Die Basis um ein ATL-Objekt ergänzen



OLE, DCOM und COM+

43

## Wie heißt das Kind?



OLE, DCOM und COM+

44

## Und was kann die Klasse?

Methode hinzufügen

OLE, DCOM und COM+

45

```

// Fibonacci.h
#pragma once
class CFibonacci
{
public:
    STDMETHODIMP Calculate(LONG n, DOUBLE* result);
};

// Fibonacci.cpp
#include "stdafx.h"
#include "Fibonacci.h"


// CFibonacci
STDMETHODIMP CFibonacci::Calculate(LONG n, DOUBLE* result)
{
    // TODO: Fügen Sie hier Ihren Implementierungscode ein.

    return S_OK;
}

```

Z 1 S 1 Zeil 1 EINF

Start Registrierungs-Editor COM+ Fibonacci Server - MIC... DE 21:19



## Literatur

---

- Guy Eddon, Henry Eddon  
*Inside COM+ Base Services*  
Microsoft Press 1999

geplanter 2. Band „Inside COM+ Component Services“  
ist immer noch nicht erschienen ☺

OLE, DCOM und COM+ 47