COM, DCOM, COM+

OVERVIEW OF MICROSOFTS COM, DCOM AND COM+ COMPONENT TECHNOLOGIES

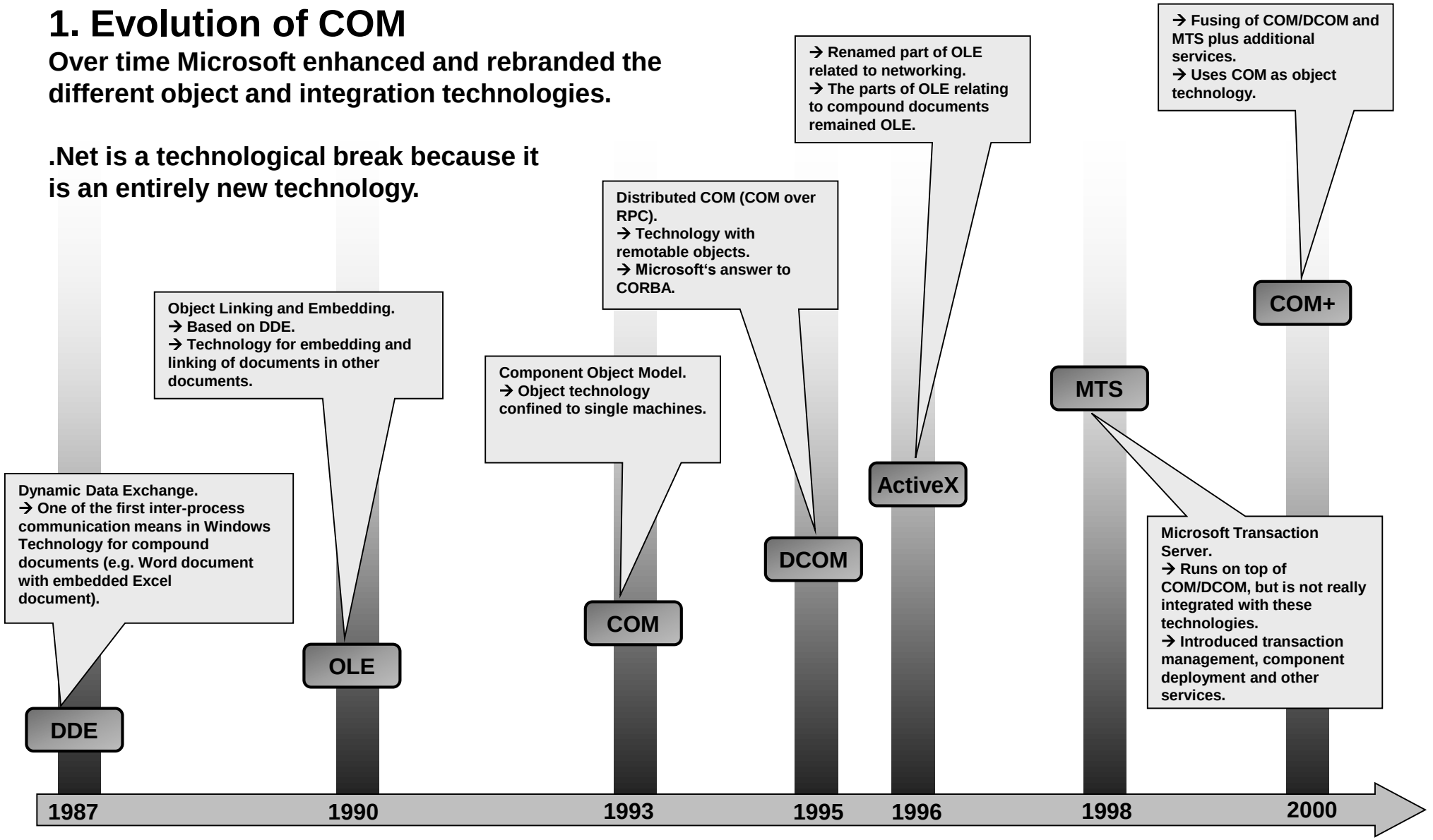
Peter R. Egli
peteregli.net

Contents

1. Evolution of COM

Over time Microsoft enhanced and rebranded the different object and integration technologies.

.Net is a technological break because it is an entirely new technology.



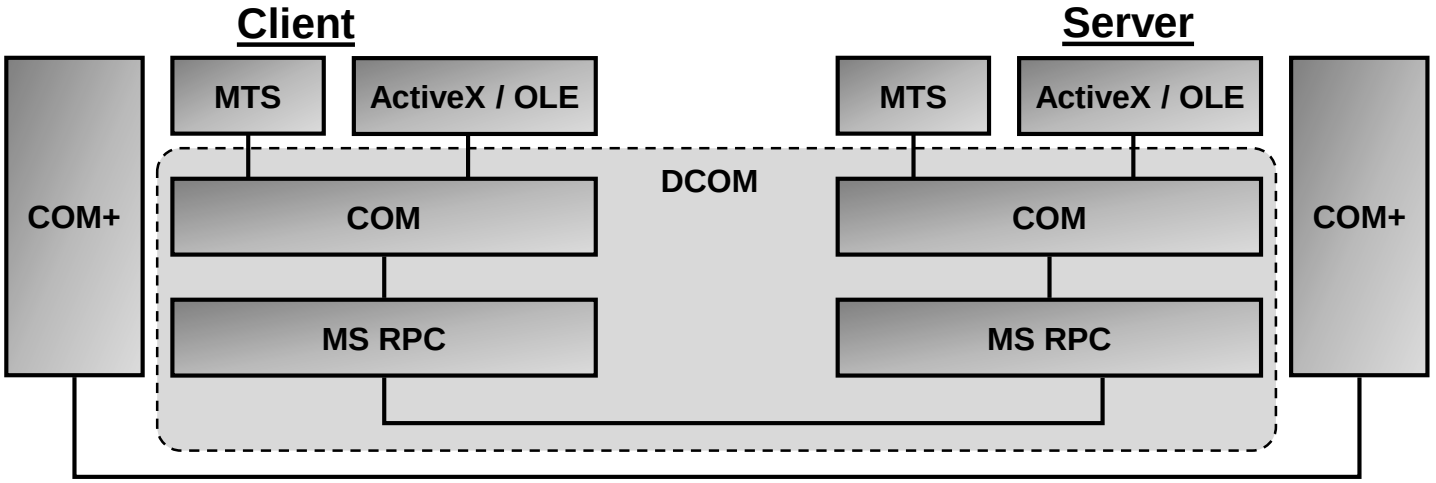
2. What are COM, DCOM, ActiveX, OLE, COM+ ?

COM is Microsofts object / component technology.

DCOM = remote access to COM objects / components (wire protocol = MSRPC which is a version of DCE RPC).

ActiveX/OLE uses COM as the underpinning. ActiveX / OLE provide additional services like reusable / programmable controls (OCX – OLE Control Extensions), automation access between office documents and in-process activation.

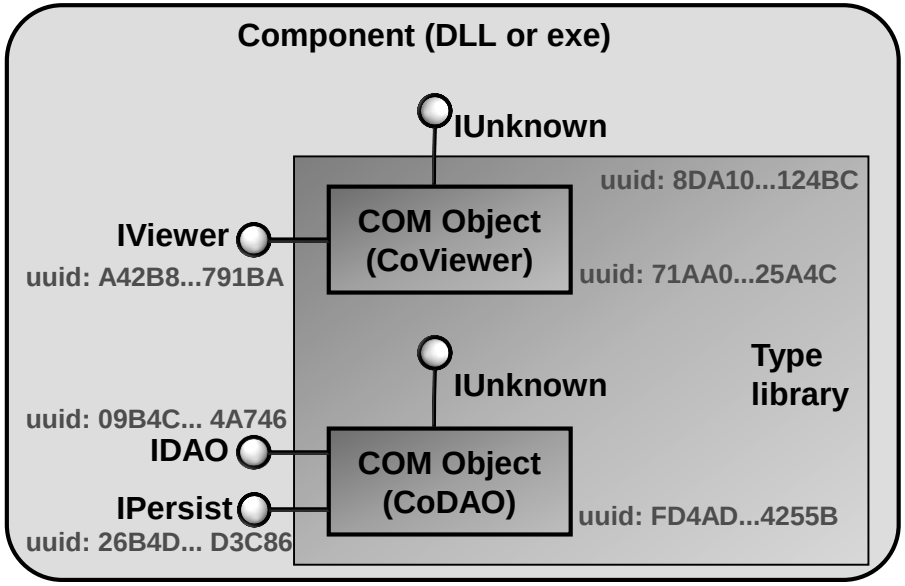
COM+ is the successor to the MTS/COM combo and provides a unified distributed component/object technology including the transaction services of MTS.
COM+ uses the COM component specification and adds additional component services.



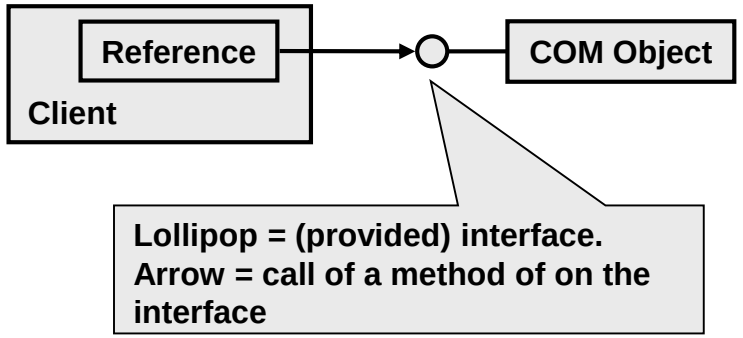
3. Structure of COM Components

Relationship of objects, components, interfaces, classes, applications:

- **A COM component contains 1..* objects.**
- **A COM component exists either as DLL (linked to the calling client, in-process) or as a separately running executable (out-of-process server).**
- **A COM object implements 1..* interfaces which are defined in the IDL-file.**
- **All objects (classes) of a component make up a type library.**
- **Every object has an IUnknown interface (see below).**
- **Component type library, every interface and every class / object has a globally unique ID (GUID, uuid).**



Notation:

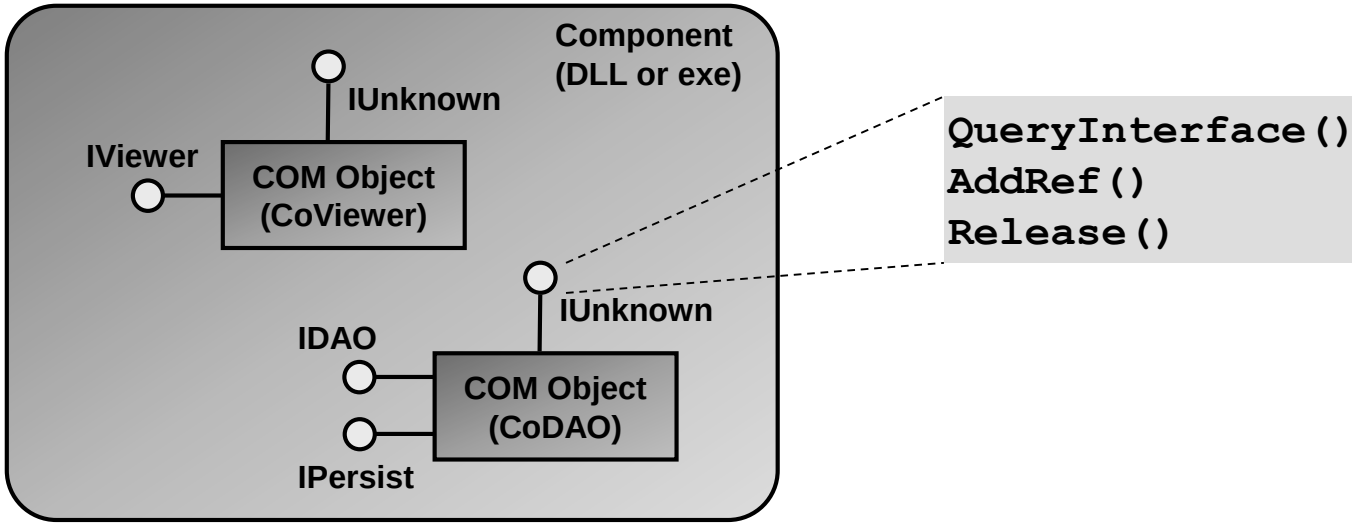


4. (D)COM IUnknown Interface

Every COM object has an IUnknown interface.
The IUnknown interface is used for:

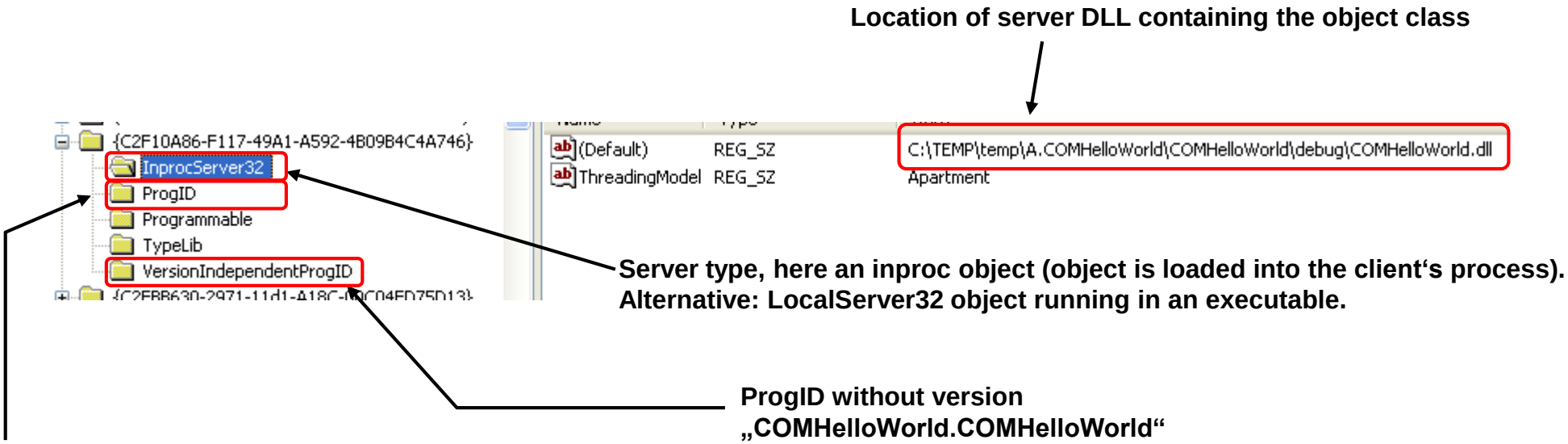
a. Introspection / reflection:
Method QueryInterface() allows to dynamically discover other interfaces of an object (check if an object supports a specific interface identified by an interface ID).

b. Life-cycle control / garbage collection (GC):
AddRef() → Client increases reference count.
Release() → Client releases its reference thus decrementing the reference count.
GC collects object once the reference count becomes 0.



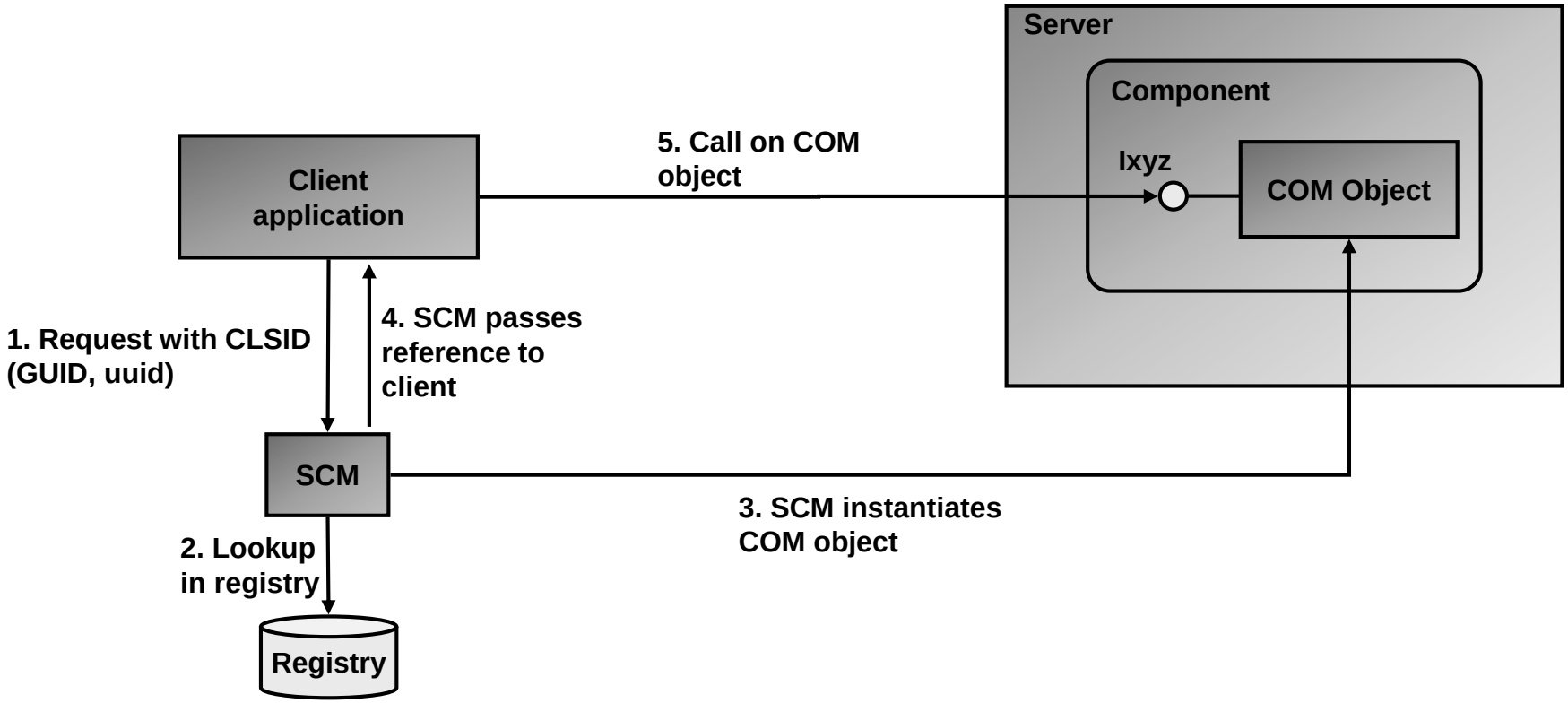
5. Component Lookup and Access (1/3)

Objects are registered in the registry under HKEY_CLASSES_ROOT/CLSID.
Every object and interface has a registry entry.
The registry is consulted for locating (lookup) objects based on a ProgID (mapping of GUID/uuid to implementation).



5. Component Lookup and Access (2/3)

Clients contact the SCM (Service Control Manager) in order to obtain an object reference.
In case of remote objects (DCOM), the local SCM contacts the remote SCM.



5. Component Lookup and Access (3/3)

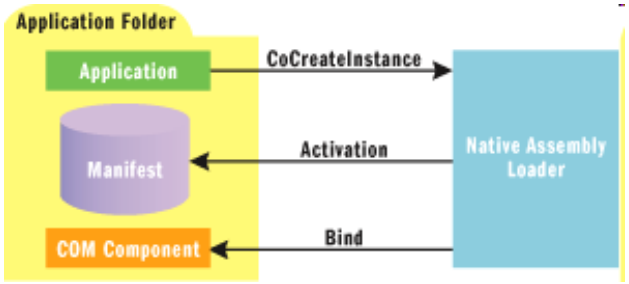
Problem of registry-based component lookup:

DLL-hell (different applications requiring different and possibly incompatible versions of COM-libraries).

Solution:

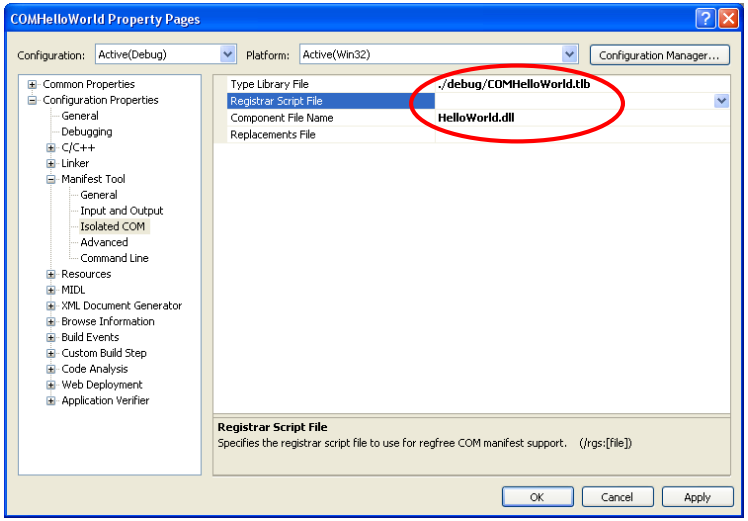
Registry free creation of objects (requires Windows XP or higher). Also called „isolated COM“.

- **Different applications may use different versions of COM-components.**
- **COM-components no longer need to be registered but may be deployed with XCOPY-deployment (simple copying of components without creating registry entries by an installer).**
- **Component is described in a manifest file which is used by the calling application to load and create the component.**



Source: http://msdn.microsoft.com/de-ch/magazine/cc188708(en-us).aspx

Visual Studio settings (create manifest file as part of build):

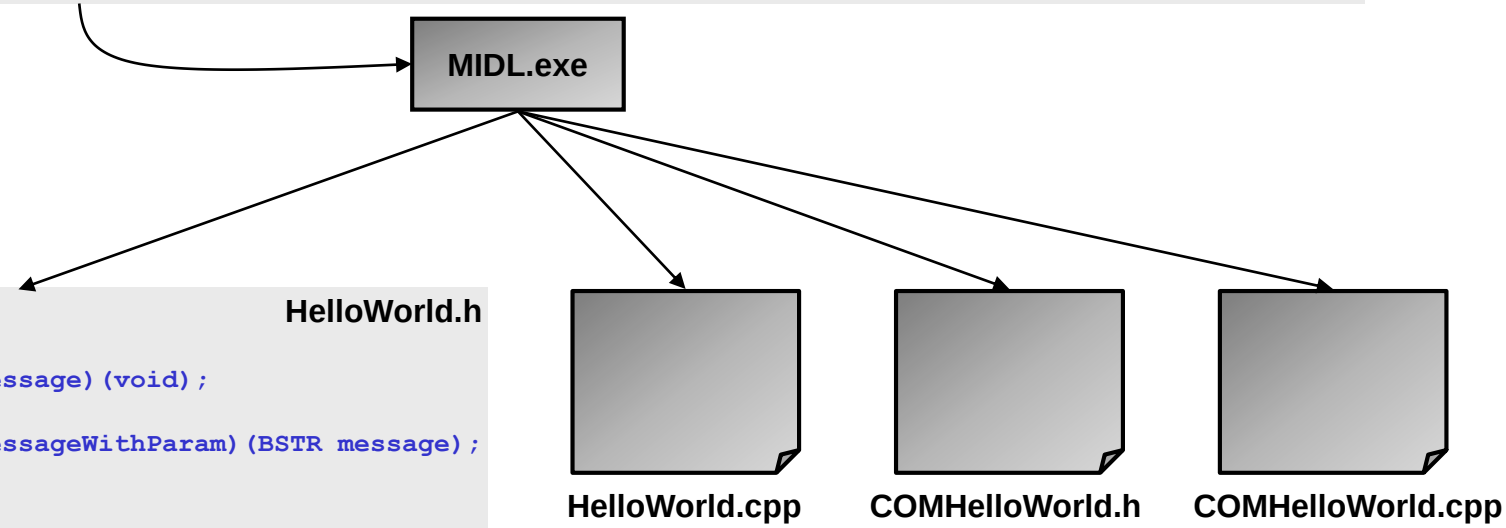


6. Microsoft IDL File

IDL-files for COM use the MIDL-format (Microsoft IDL).
IDL files are compiled with MIDL.exe (automatically in Visual Studio ATL-project).

COMHelloWorld.idl

```
...
[
            object,
            uuid(FD4ADCD2-C7FC-466E-AD75-EBC03024255B),
            dual,
            nonextensible,
            helpstring("ICOMHelloWorld Interface"),
            pointer_default(unique)
]
interface ICOMHelloWorld : IDispatch{
            [id(1), helpstring("method ShowMessage")] HRESULT ShowMessage(void);
            [id(2), helpstring("method ShowMessageWithParam")] HRESULT ShowMessageWithParam([in] BSTR message);
};
...
```



HelloWorld.h

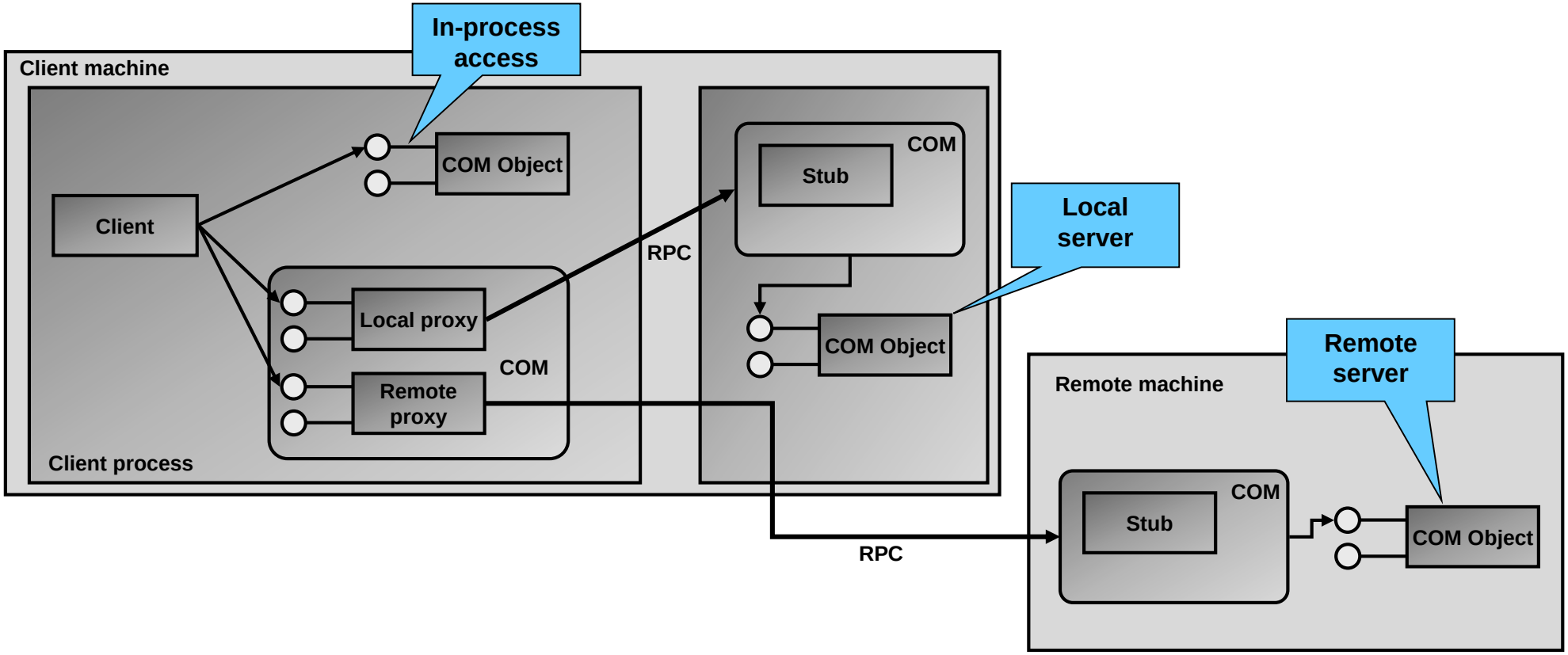
```
...
public:
            STDMETHOD(ShowMessage)(void);
public:
            STDMETHOD(ShowMessageWithParam)(BSTR message);
};
...
```

7. Execution / Access Models

In-proc access: Component resides in a DLL. The client loads the component into its process.

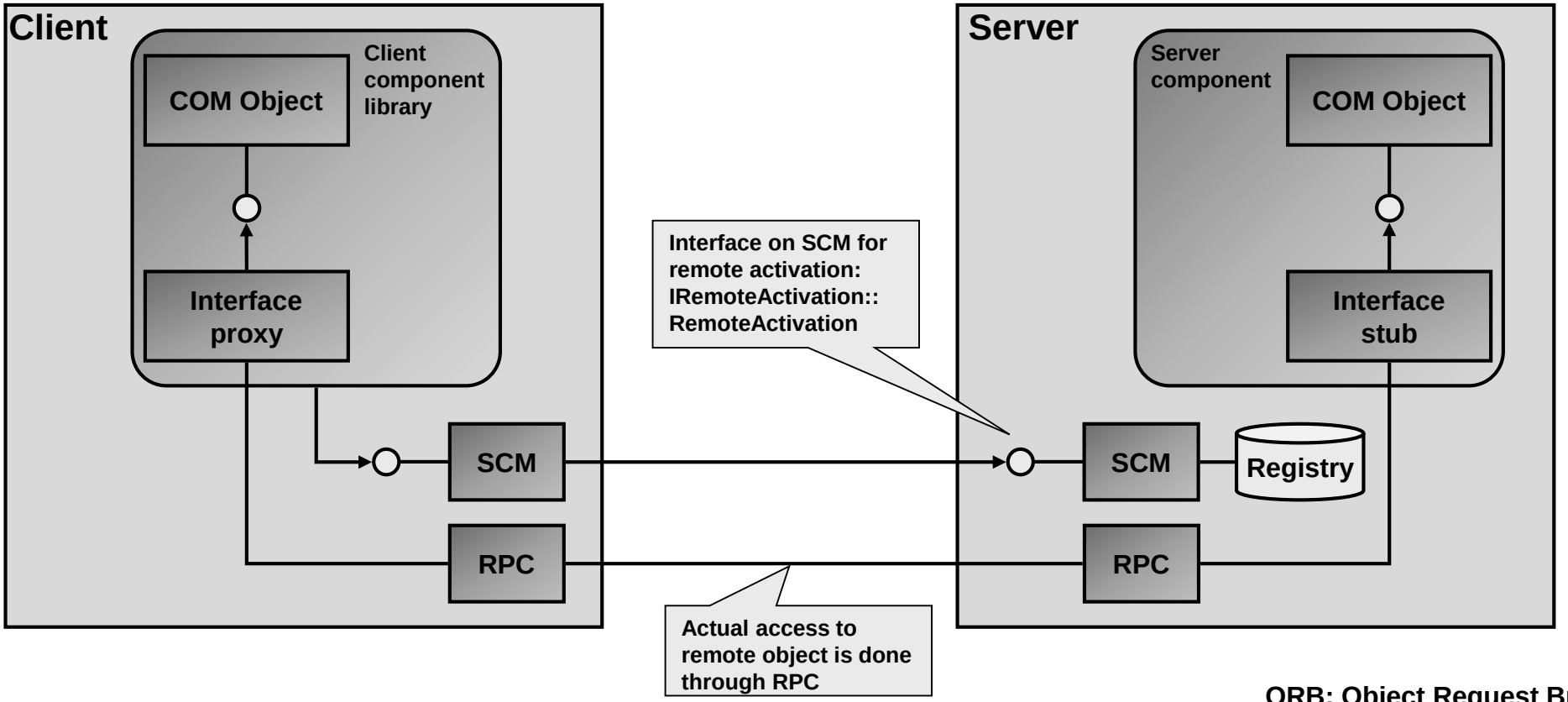
Local server: Component resides in an executable and runs in its own process on the local machine. Access through RPC.

Remote server: Component resides in an executable and runs on a remote machine. Access through RPC.



8. DCOM Architecture

Client proxy: Proxy object on client side for accessing the server object.
Stub: Server interface stub that complements the client interface proxy.
Registry: Contains a list of mappings of class / object GUID to implementation library.
SCM: Service Control Manager (RPCSS.exe) which consults registry and creates / instantiates a new server object based on the GUID (comparable to ORB in CORBA). The SCM hides the registry from (D)COM.



ORB: Object Request Broker

9. COM / DCOM / COM+ Tools

Register COM component:

regsvr32 <COM-lib>.dll

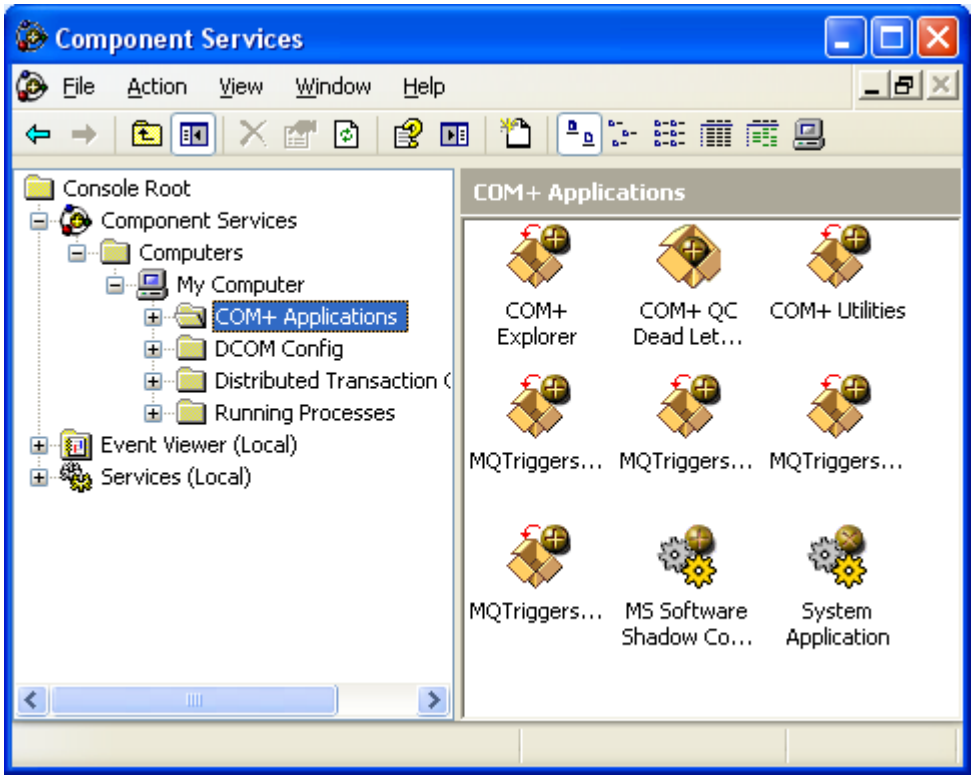
Registry of objects:

Windows registry (edit with regedit.exe or regedt32.exe)

Component service explorer:

Control Panel → Administrative Tools → Component Services

or simply start dcomcnfg.exe



10. Access (D)COM Objects from .Net (COM-.Net Interop)

.Net introduced an entirely new object model:

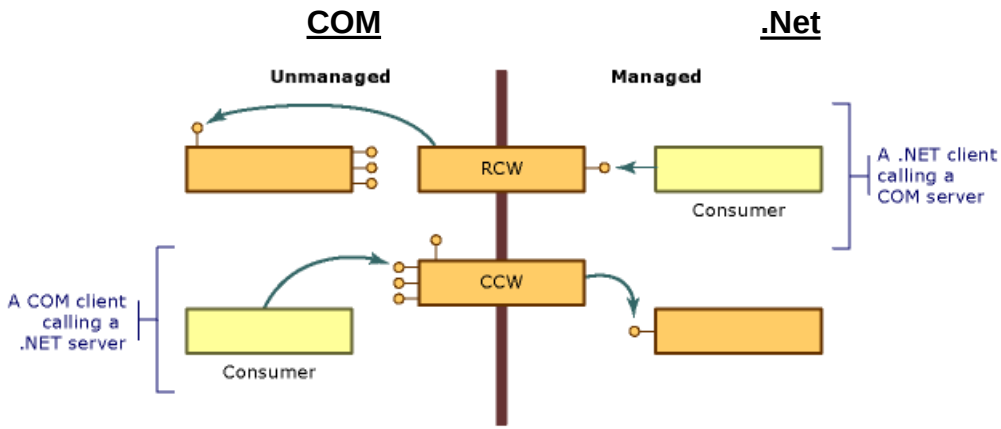
- **.Net object lifetime is managed by garbage collector (COM: lifecycle controlled by client).**
- **.Net clients have far greater introspection possibilities through reflection.**
- **.Net objects reside in a managed environment (managed code).**

The .Net environment (CLR: Common Language Runtime) provides wrappers for COM-.Net interoperability:

COM→.Net: COM callable wrapper CCW .
.Net→COM: Runtime Callable Wrapper (RCW).
The wrappers are contained in DLLs called Interop.xxx.

Tasks of wrappers:

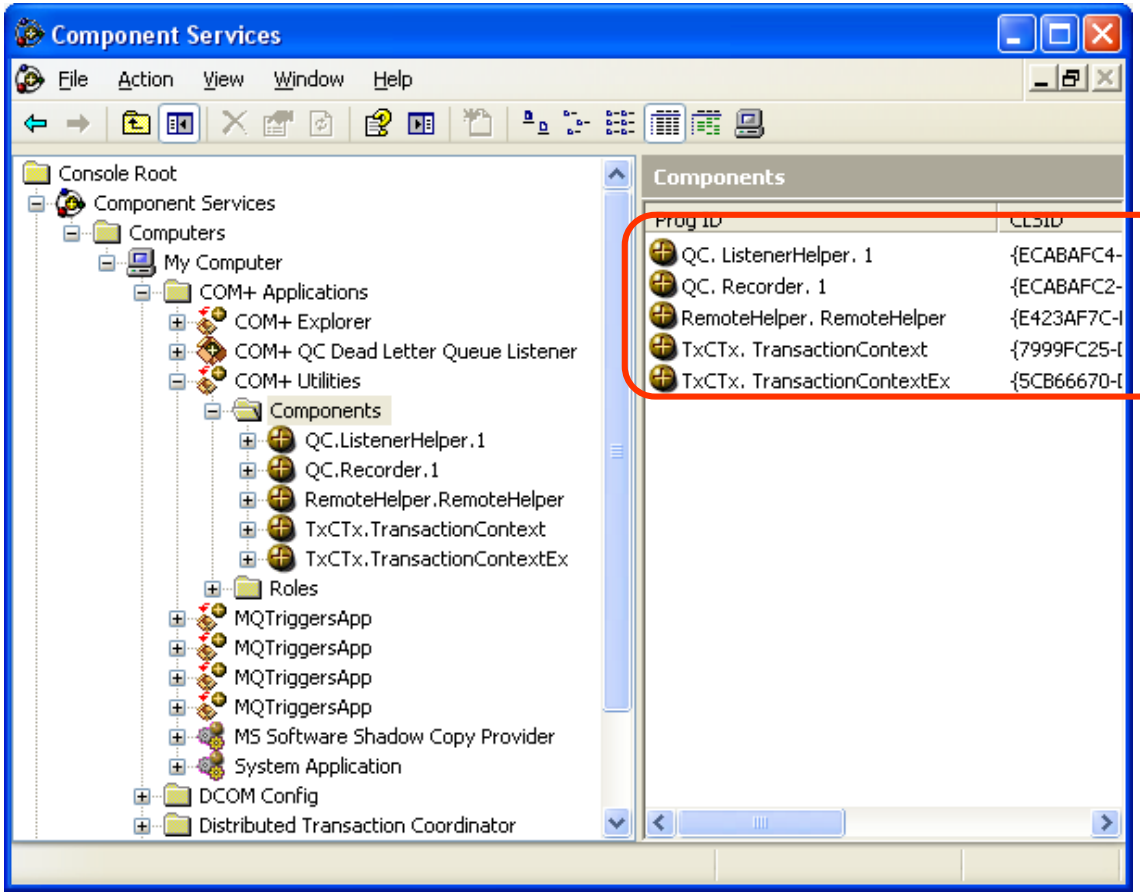
- **Marshalling of parameters (e.g. MFC-type BSTR ←→ .Net string)**
- **RCW: COM object reference counting (RCW); decreases reference count on COM object if object is no longer needed on .Net managed side.**
- **.Net object reference release (CCW); map COM-side Release() call to .Net managed object release**



Source: http://msdn.microsoft.com/en-us/library/5dxz80y2.aspx

11. COM+ Applications

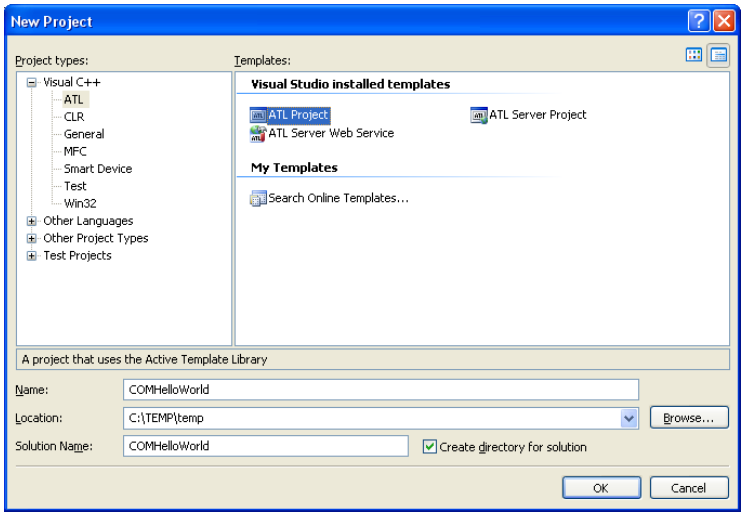
COM+ applications host (contain) 1..n COM-components.
Dcomcnfg.exe shows a list of active COM+ applications.

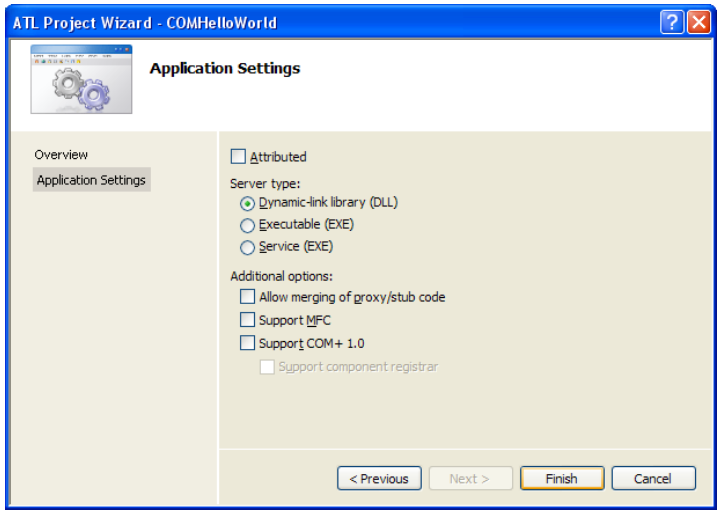


COM components of „COM+ Utilities“ COM+ application

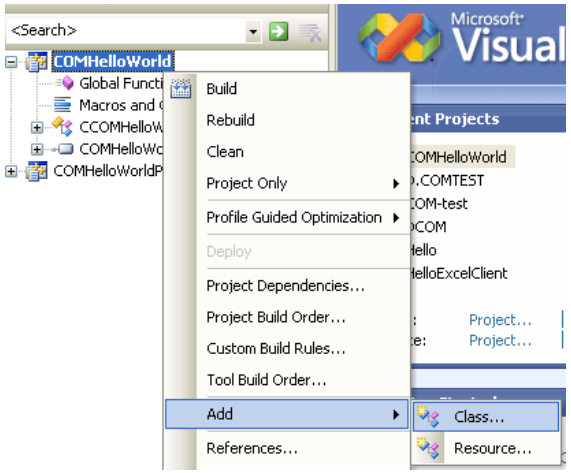
12. Creation of COM Projects in Visual Studio (1/4)

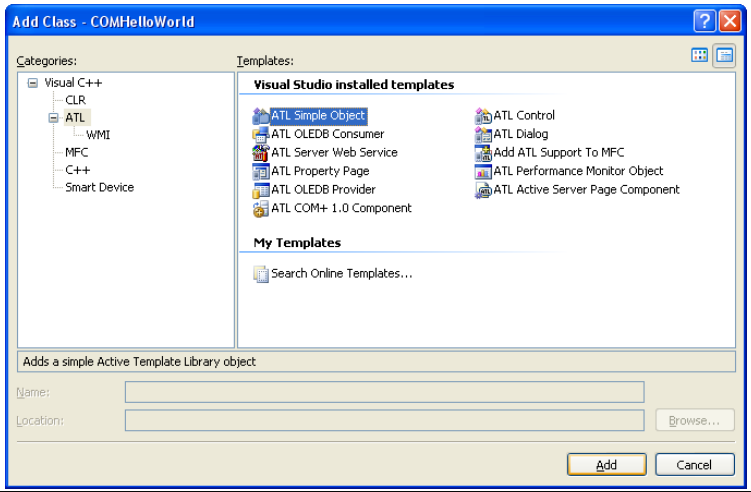
1. Create Visual Studio ATL C++ project with the following settings:





2. Add a new class to the COM component:

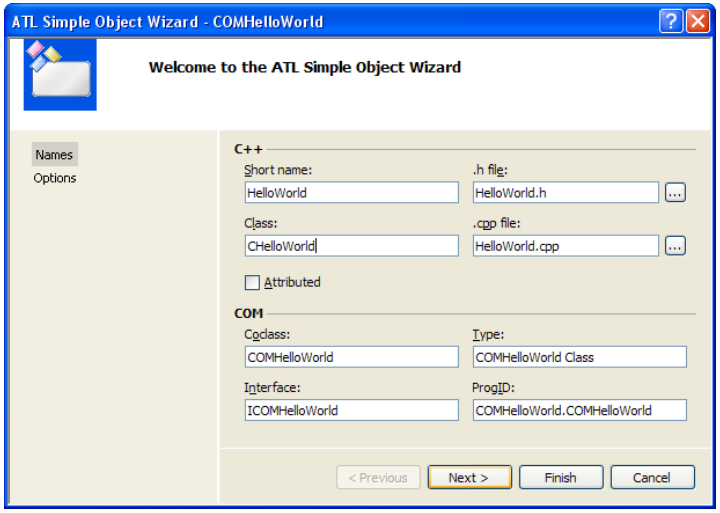


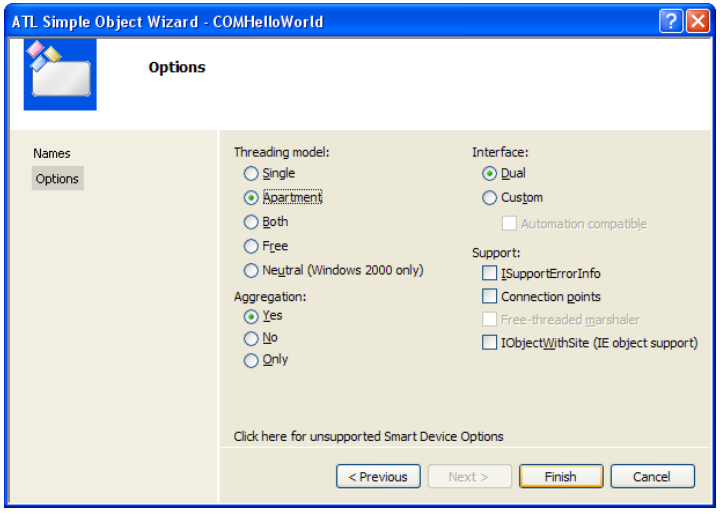


12. Creation of COM Projects in Visual Studio (2/4)

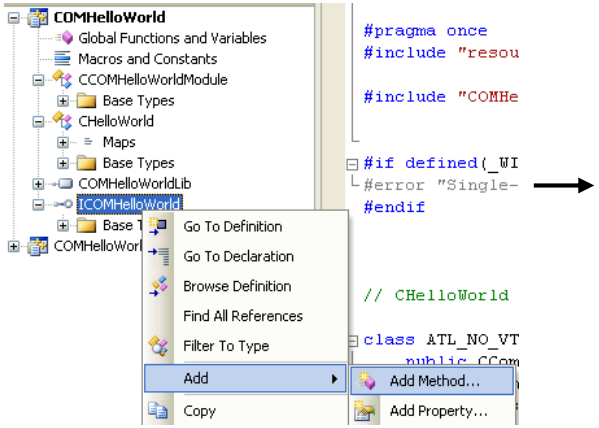
3. ATL Simple Object Wizard:

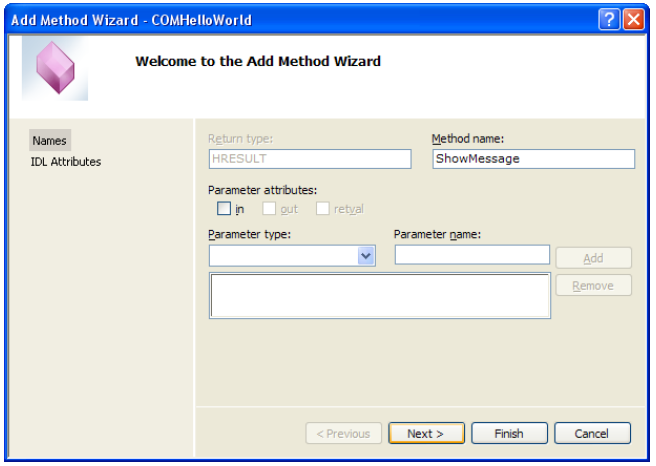
Set names (Class, Coclass and Interface names may be different):

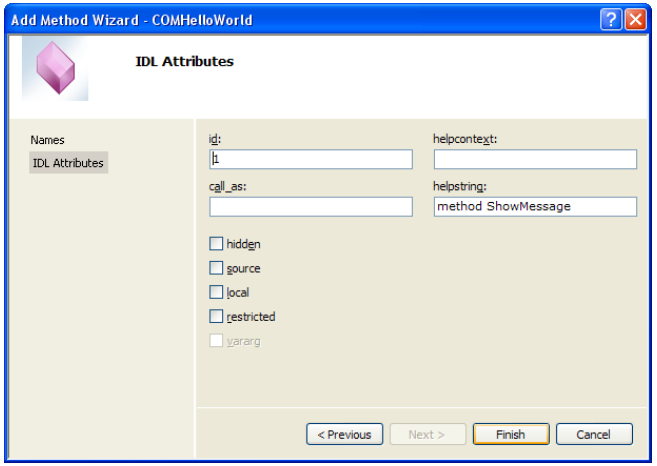




4. Add method to interface (e.g. ShowMessage()):







12. Creation of COM Projects in Visual Studio (3/4)

5. Implement interface method (ShowMessage()):

Add user code to implementation of method in implementation class (HelloWorld.cpp):

```
STDMETHODIMP CHelloWorld::ShowMessage(void)
{
   ::MessageBox(::GetActiveWindow( ),_T("Hello World from COMHelloWorld."),
                                      _T("First COM+ Application"),MB_OK);
    return S_OK;
}
```

6. Compile ATL COM project

7. Register COM:

Open command shell, change to debug directory of COM project.

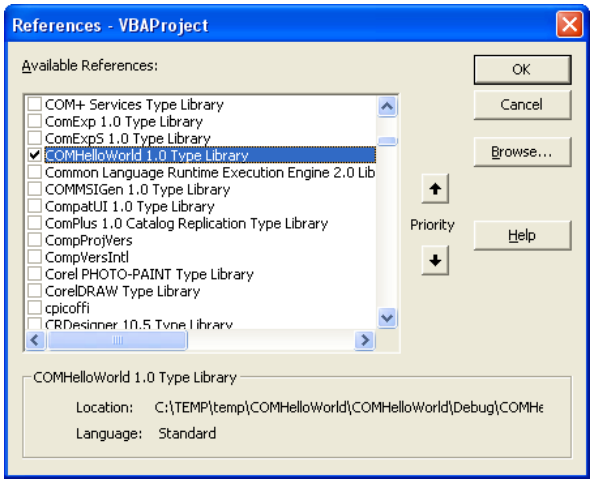
```
>regsvr32.exe COMHelloWorld.dll
```

8. Create client, e.g. in Excel-VBA:

Create new Excel file

Tools→Macro→Visual Basic Editor

Add reference to COM component (Tools→References):

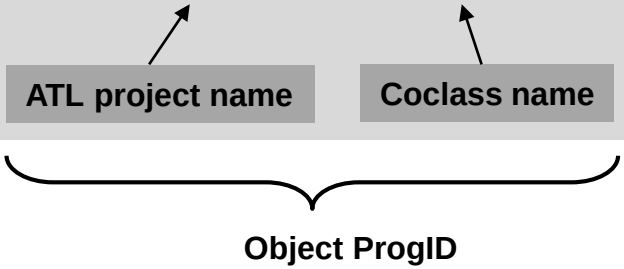


12. Creation of COM Projects in Visual Studio (4/4)

9. Access the COM component from VBA code:

```
'TestHelloClient_TLB accesses the COM component using the COM type library (TLB)
'that has to be added to this project in the references (Menu Tools->References)
Sub TestHelloWorldClient_TLB()
  Dim hw As COMHelloWorld
  Dim gw As COMGoodbyeWorld

  Set hw = New COMHelloWorld
  hw.ShowMessage

  Set gw = New COMGoodbyeWorld
  gw.ShowMessage
End Sub
```

```
'TestHelloWorldClient_ProgID access the COM component using the ProgID
Sub TestHelloWorldClient_ProgID()
  Dim obj As Object

  Set obj = CreateObject("COMHelloWorld.COMHelloWorld")
  obj.ShowMessage

  Set obj = CreateObject("COMHelloWorld.COMGoodbyeWorld")
  obj.ShowMessage
End Sub
```



13. Limitations of COM

COM is still widely used by many applications to provide programmatic access for automation purposes.
However, due to many technological limitations (see below) COM was technologically superseded by .Net.

- **No true inheritance (may be emulated with aggregation and containment).**
- **No exceptions (only return codes).**
- **Inconsistent use of IDL (COM uses IDL, but VB or scripting languages like VBA use binary representation (type library)).**
- **No OO-features (static modifier, virtual functions, overloaded methods).**