En introduktion til

Microsofts

Component **O**bject **M**odel

Agenda

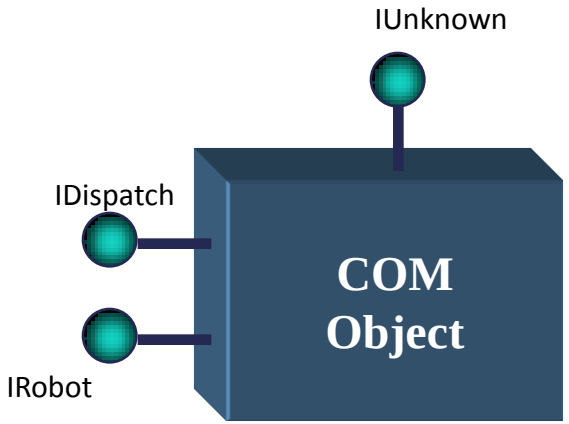
- Why COM?
- COM architecture
- Programming COM components
- Programming COM clients

Goals of COM

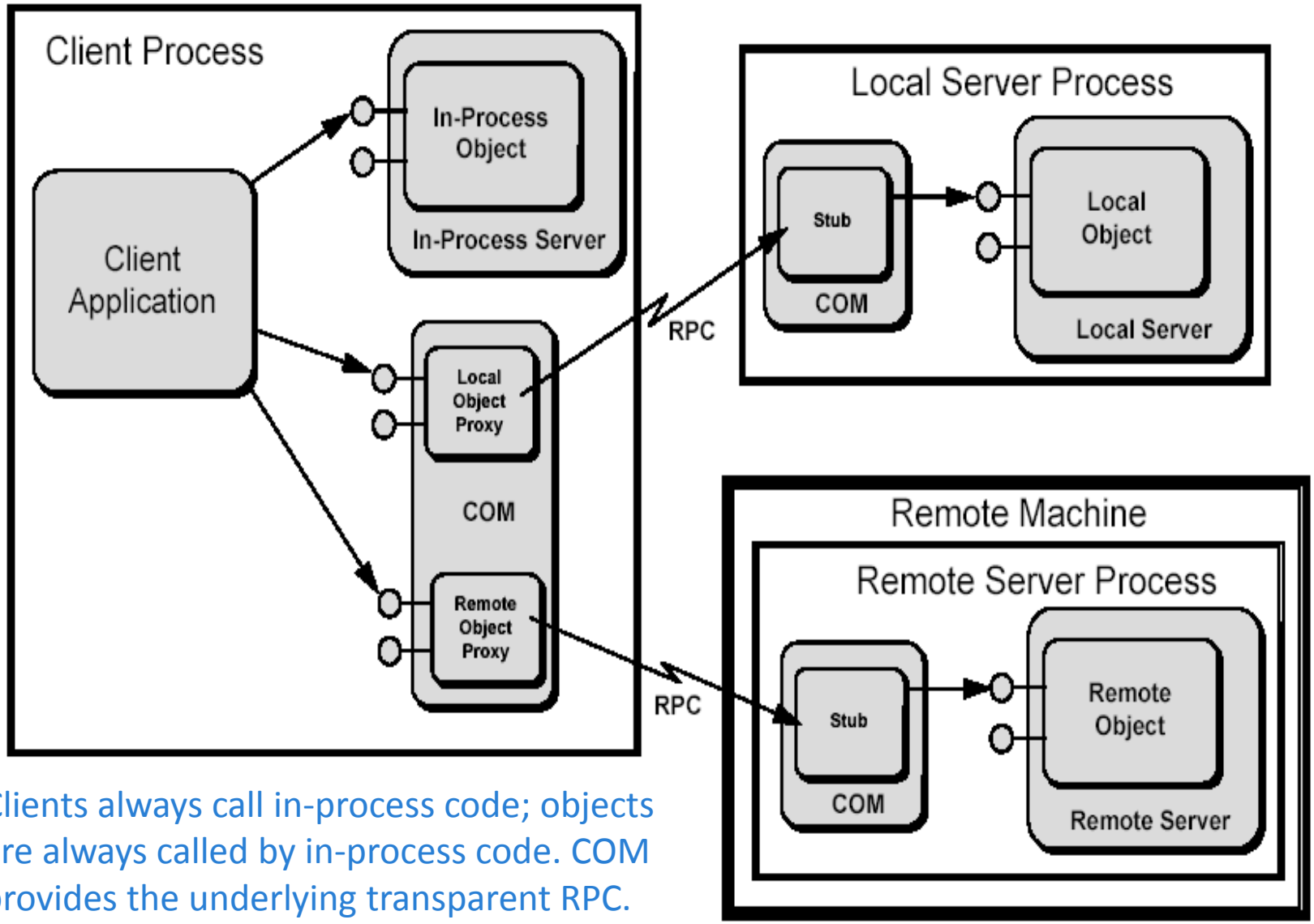
- Provide a component object model that facilitates binary encapsulation and binary compatibility
  - Binary encapsulation: Clients do not have to be re-compiled if server objects change
  - Binary compatibility: Client and server objects can be developed with different development environments and in different languages
- COM is proprietary de-facto standard

COM Principles

- Rigorous Encapsulation
  - Black box -- no leakage of implementation details
  - All object manipulation through strict interfaces
- Polymorphism
  - via multiple interfaces per class
  - “Discoverable”: QueryInterface
- Interfaces are immutable after publish…
  - Get them right the first time

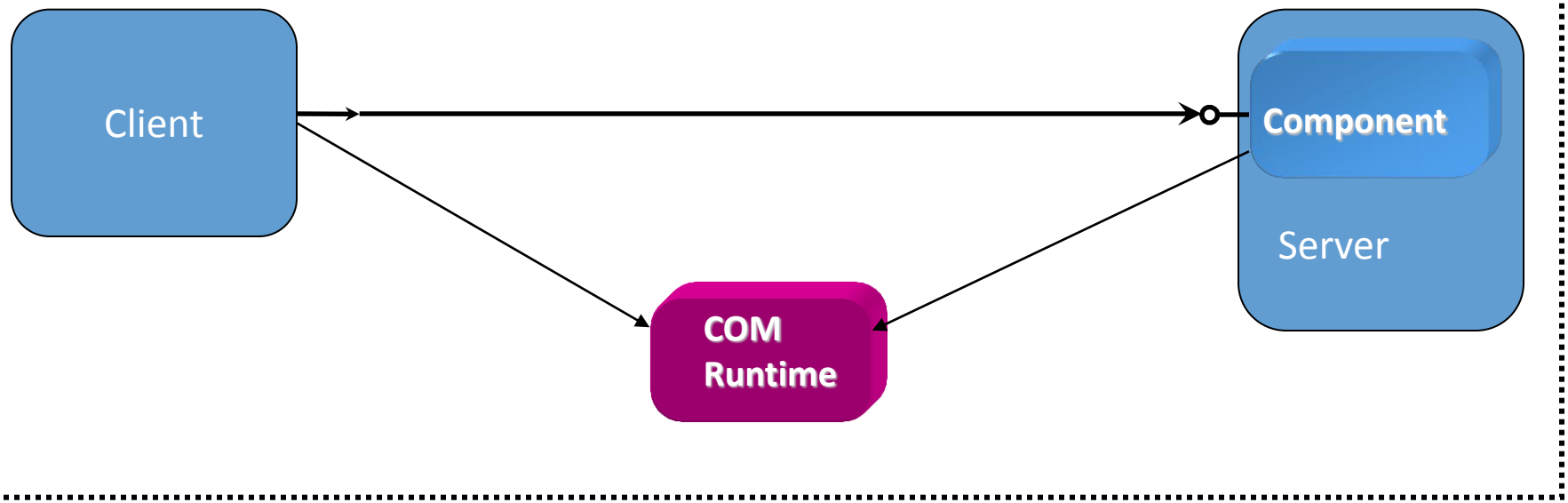


COM Architecture



Clients always call in-process code; objects are always called by in-process code. COM provides the underlying transparent RPC.

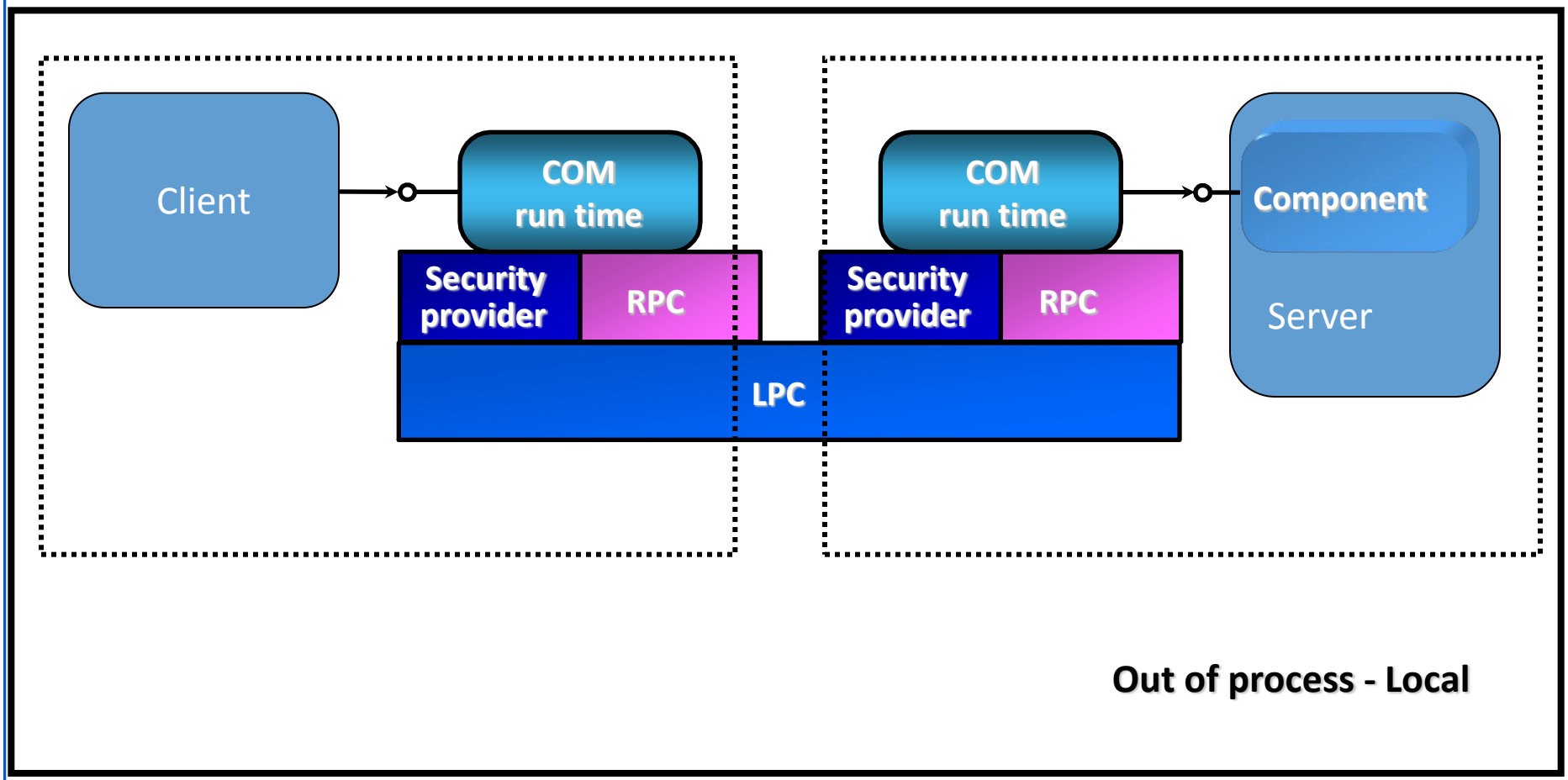
COM Architecture: In process



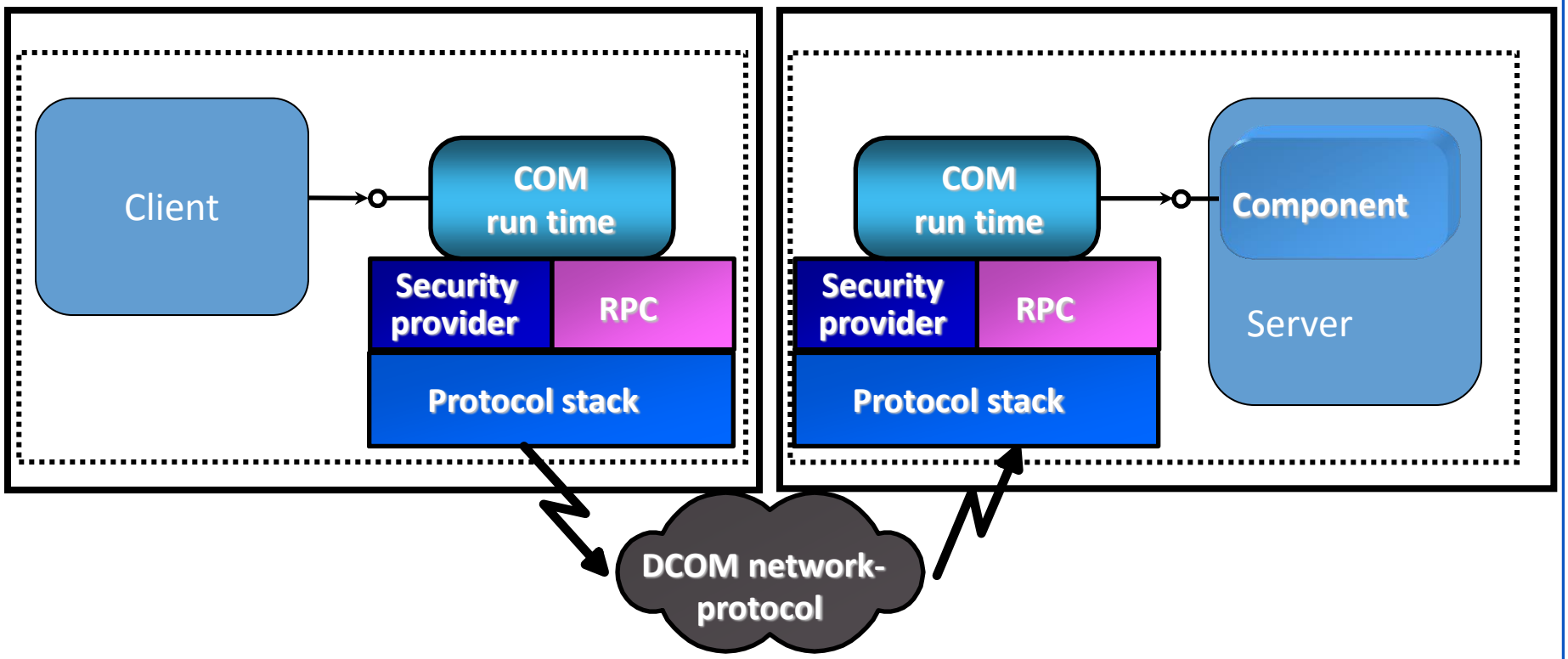
Once COM connects client and server object, the client and server object communicate directly without added overhead

In process

COM Architecture: Local

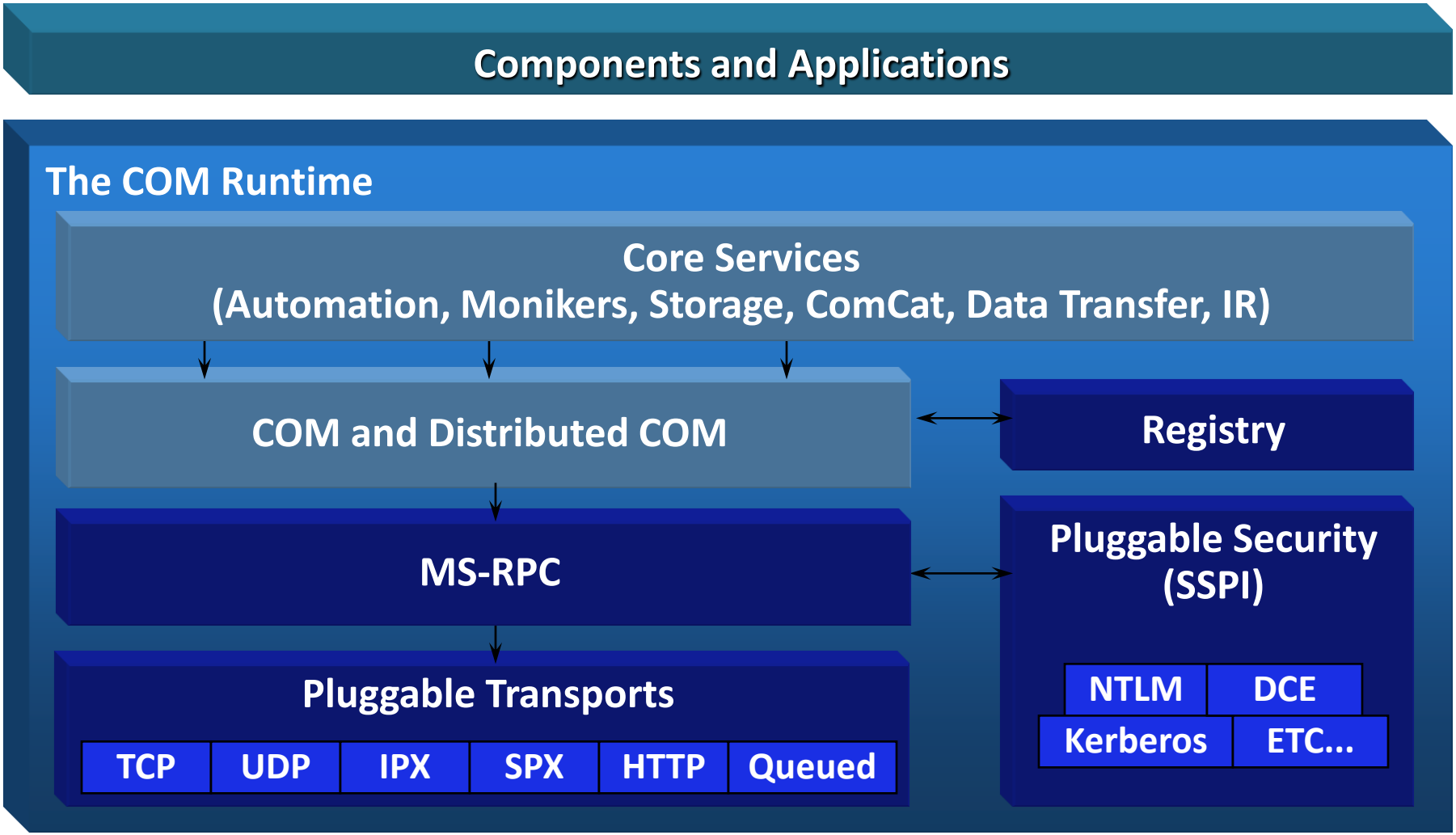


COM Architecture: Out of process - Remote

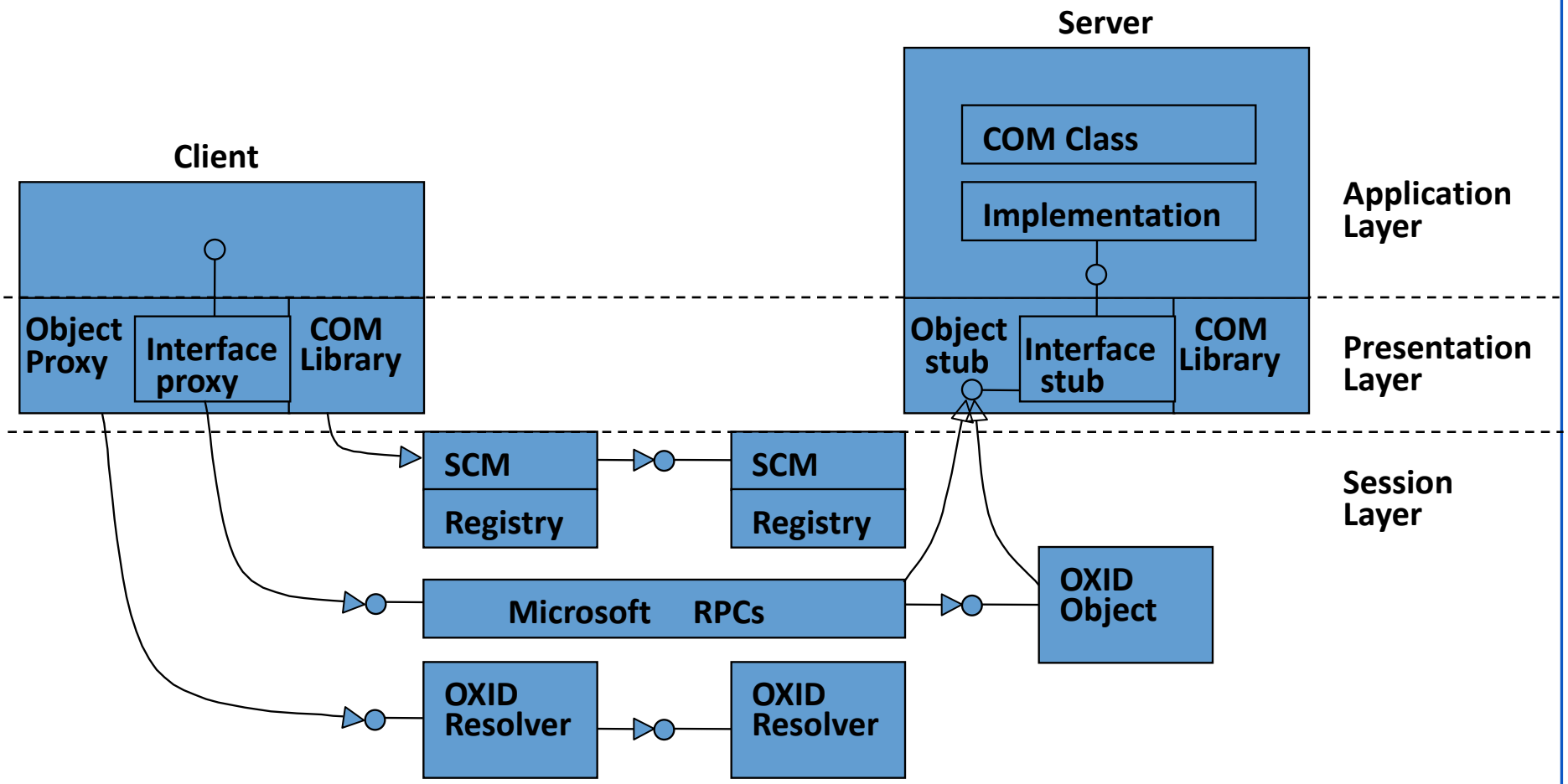


Out of process - Remote

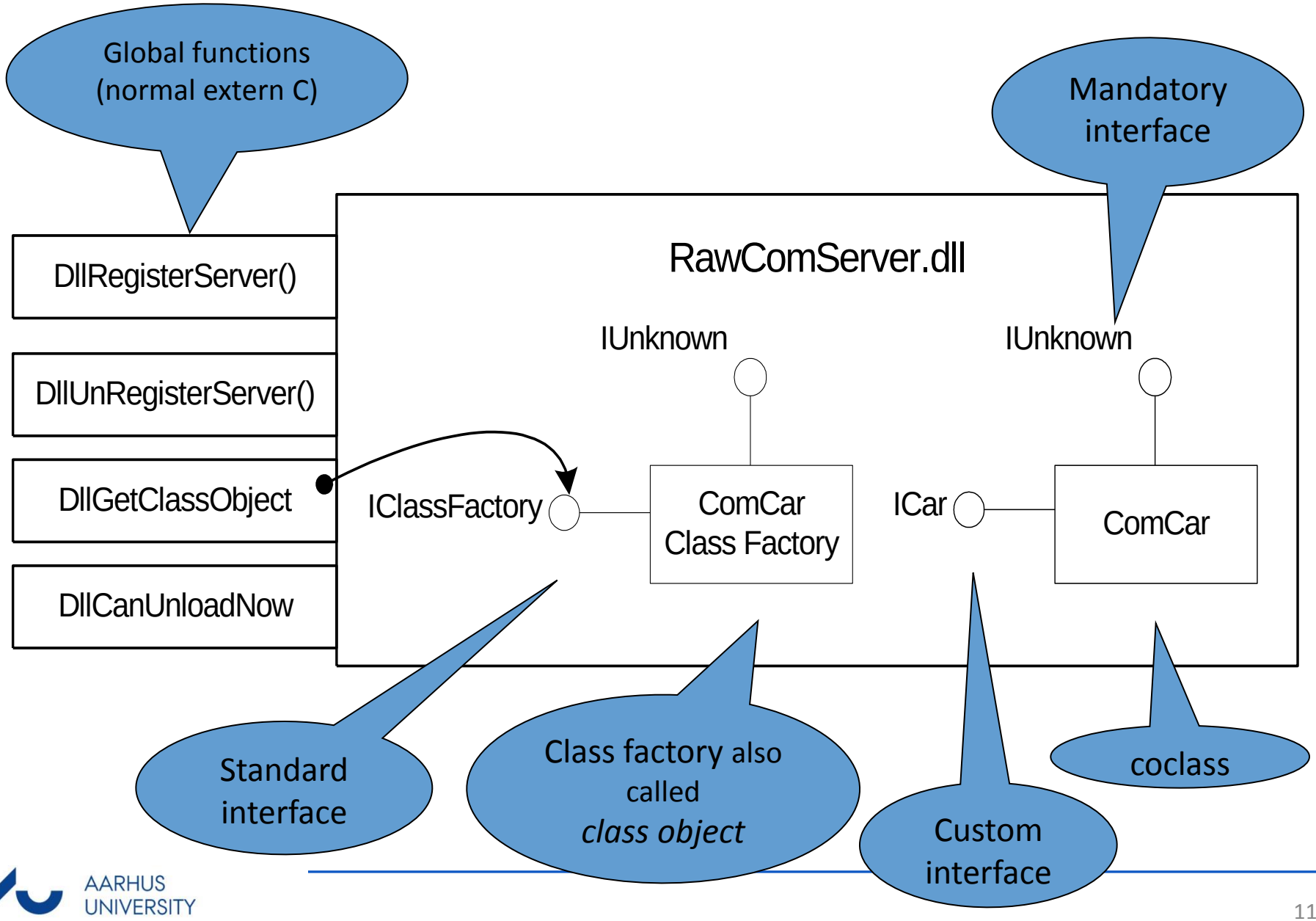
Structure of the COM “ORB”



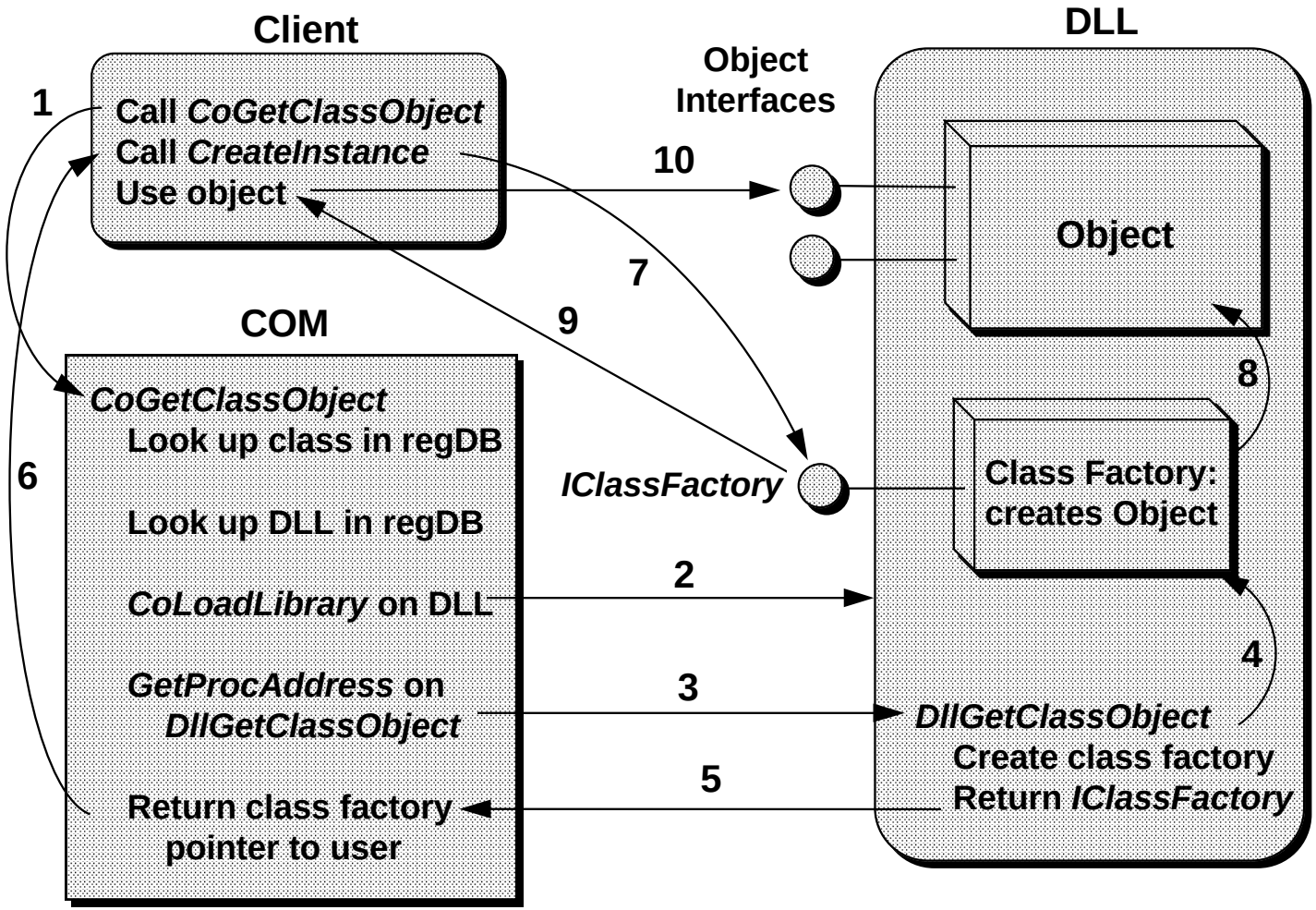
Architecture



The Composition of a COM DLL



Creation sequence of an object from a COM DLL server



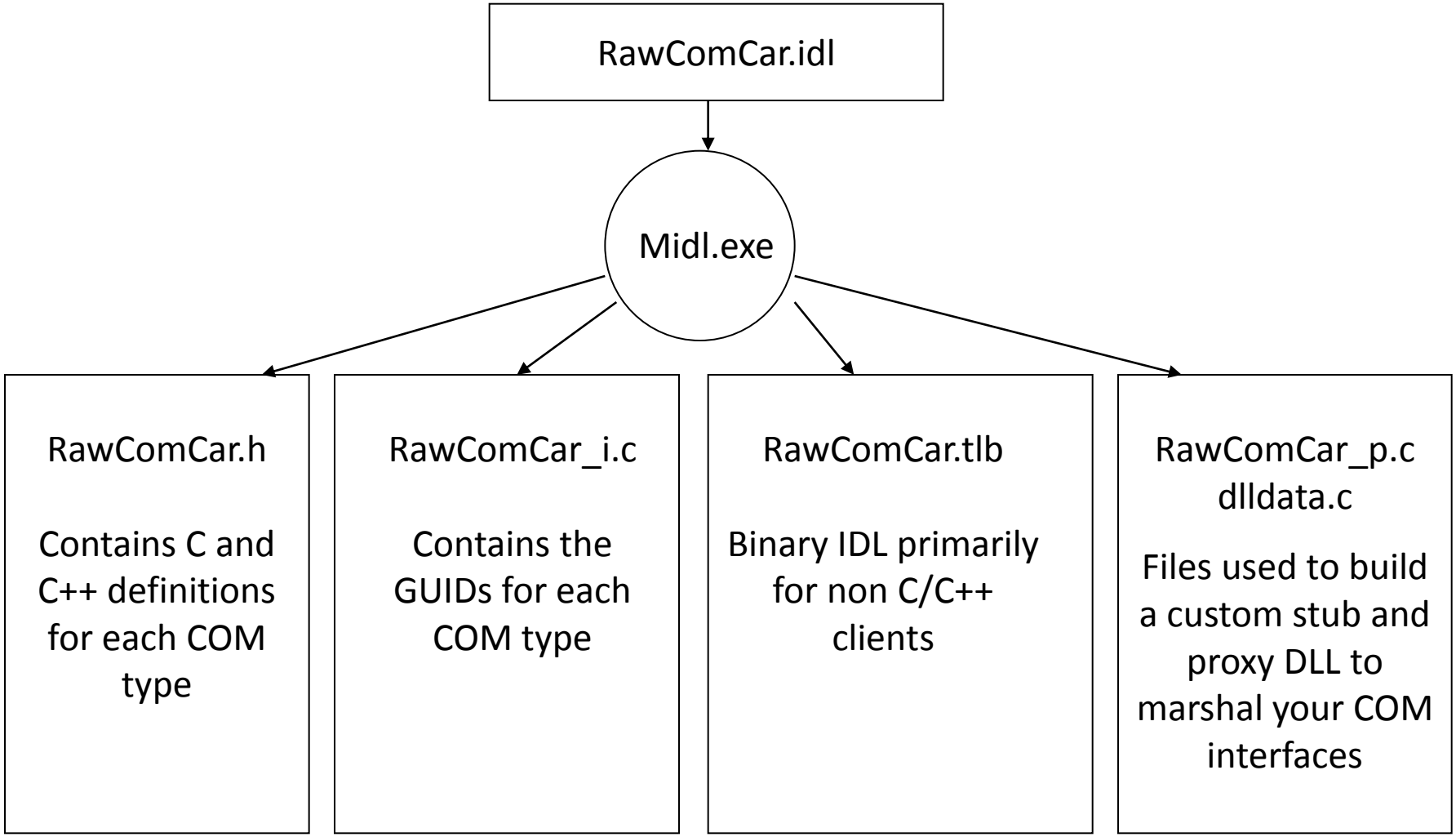
Interface

- A group of related functions that specifies a contract
  - name, interface signature, semantics, and marshaling buffer format
- Denotes behavior only, not state
- Identified by an GUID – Global Unique Identifier
  - Called IID – Interface ID
- Must derive from IUnknown
  - either directly or through other interfaces

Microsoft IDL (MIDL)

- **I**nterface **D**efinition **L**anguage
  - Language for expressing all COM concepts
- MIDL is
  - programming-language independent
  - evolved from OSF/RPC IDL
  - not computationally complete but better than C++ header files
- In COM IDL files are compiled with a MIDL compiler

MIDL output


RawComCar.idl
Midl.exe
RawComCar.h
Contains C and C++ definitions for each COM type
RawComCar_i.c
Contains the GUIDs for each COM type
RawComCar.tlb
Binary IDL primarily for non C/C++ clients
RawComCar_p.c
dlldata.c
Files used to build a custom stub and proxy DLL to marshal your COM interfaces

COM Interfaces in MIDL

```
// RawComCar.idl
import "oaidl.idl";

// The ICar interface
[uuid(710D2F54-9289-4f66-9F64-201D56FB66C7), object]
interface ICar : IUnknown
{
  HRESULT SpeedUp([in] long delta);
  HRESULT CurrentSpeed([out, retval] long* currSp);
};
```

IID

Interface

Items in square brackets are called attributes.
(*attributes in COM are not related to attributes in UML)*

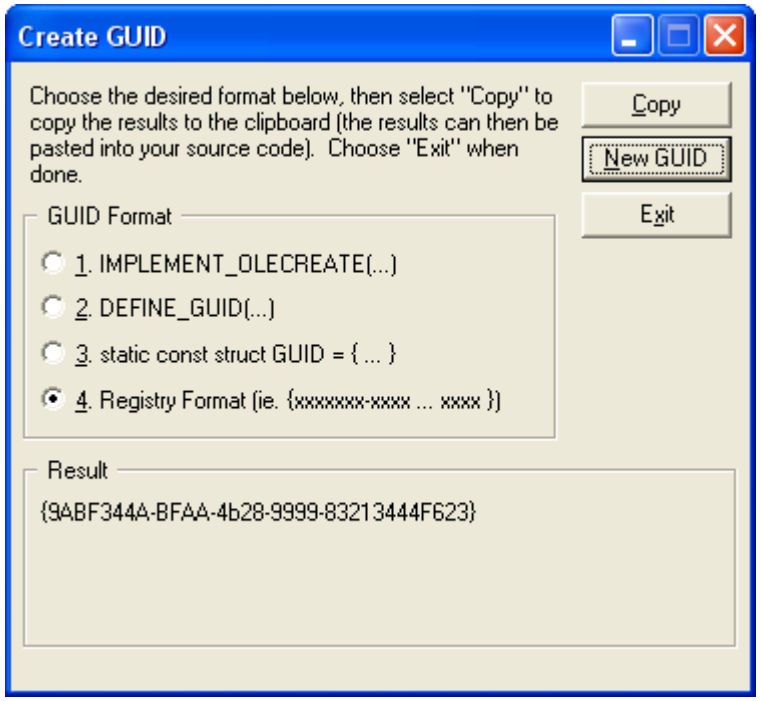
Root Interface

IDL Keywords

Attribute	Context	Meaning
uuid	interface	The IID of the interface for use by QueryInterface
object	interface	Indicates that the interface is a COM interface and not a DCE RPC function-based interface
pointer_default(ptype)	interface	Sets the default pointer attribute for pointers used in the interface
local	interface, method	A prototype should be produced for the generated header file, but does not produce a proxy/stub implementation
in	parameter	The parameter value must be sent from the caller to the object; can be combined with out
out	parameter	The parameter value must be sent from the object to the caller; can be combined with in
ref	parameter	The pointer parameter always points to valid memory (cannot be null) and never points to the same memory as any other method parameter
unique	parameter	The pointer parameter points to valid memory or is null, and never points to the same memory as any other method parameter
ptr	parameter	The pointer parameter points to valid memory or is null, and may point to the same memory as any other method parameter
size_is(cElems)	parameter	The pointer parameter is an array that has the capacity for cElems elements
length_is(cElems)	parameter	The pointer parameter is an array and the first cElems elements are valid and should be marshaled
string	parameter	The pointer parameter points to a null-terminated string, and the result of strlen or wcslen should be used to determine the number of valid bytes to be marshaled

GUID

- A GUID is a 128-bit number that is statically unique
- GUID's are used to identify severel different COM subjects, e.g. interface IID, coclass CLSID …
- Can be created by a tool or programmatically



Wtypes.h has many defines ease working with GUIDs, Ex:

#define REFGUID const GUID * const
#define REFIID comst IID * const

And objbase.h has some helper functions and operator overloads

COM Class ~ coclass

- Named, concrete implementation of one or more interfaces
- Identified by a CLSID (~ GUID)
  - sometimes ProgID too
- Type libraries may describe incoming, outgoing interfaces

COM Class in MIDL

```
// RawComCar.idl
// The Raw Car Library.
[uuid(D679F136-19C9-4868-B229-F338AE163656), // Library ID (LIBID)
version(1.0),]
library RawComCarLib
{
        importlib("stdole32.tlb");
        // Our COM class.
        [uuid(096AC71D-3EB6-4974-A071-A3B1C0B7FC8D)]
        // Class ID (CLSID)
        coclass ComCar
        {
                [default] interface ICar;
                interface IRadio;
        };
        // an other COM class.
        [uuid(7AD9AFC9-771C-495c-A330-006D54A23650)]
        // Class ID (CLSID)
        coclass ScriptableCar
        {
                [default] interface IScriptableCar;
        };
};
```

Implementing the COMCar

```
// ComCar.h: interface for the ComCar class.
#include <windows.h>
// MIDL generated file!
#include "rawcomcar.h"
// ComCar implements IUnknown, ICar and IRadio.
class ComCar : public ICar, public IRadio
{
public:
        ComCar();
        virtual ~ComCar();
        // IUnknown impl.
        STDMETHOD_(ULONG,AddRef)();
        STDMETHOD_(ULONG,Release)();
        STDMETHOD (QueryInterface)(REFIID riid, void**);
        // ICar impl.
        STDMETHOD (SpeedUp)(long delta);
        STDMETHOD (CurrentSpeed)(long* currSp);
        // IRadio impl.
        STDMETHOD (CrankTunes)();
        // data members
        ULONG m_refCount; // Ref counter.
        long m_currSpeed; // Current speed!
};
```

ComCar's implementation of IUnknown

```
STDMETHODIMP_(ULONG) ComCar::AddRef() {
        return ++m_refCount;
}

STDMETHODIMP_(ULONG) ComCar::Release() {
        if(--m_refCount == 0)
                delete this;
        return m_refCount;
}

STDMETHODIMP ComCar::QueryInterface(REFIID riid, void** ppInterface)
{
        // Remember!  Always AddRef() when handing out an interface.
        if(riid == IID_ICar)   {
                *ppInterface = (ICar*)this;
                ((IUnknown*)(*ppInterface ))->AddRef();
                return S_OK;
        }
        …
}
```

Objects vs. Components

- COM Object
  - an instance of a COM Class
- COM Component
  - binary unit of code that creates COM objects
  - includes packaging code , registration code, class factory, etc.

COM: Objects

- Instances of COM Implementations
- References to COM objects are called interface pointers
- Interface pointers refer to main memory locations
- References support location transparency
- Object references are persistent

COM Classes

- Named implementations
- Have one or several interfaces
- Are the principal mechanism to create COM objects
- Can return interface pointers to specific COM objects

UML: Attributes

- COM does support UML attributes (state information)
- Attributes must be represented as set and get operations by the designer
- COM has a keyword to designate this: **Property**
  - propget
  - propset
- Example:

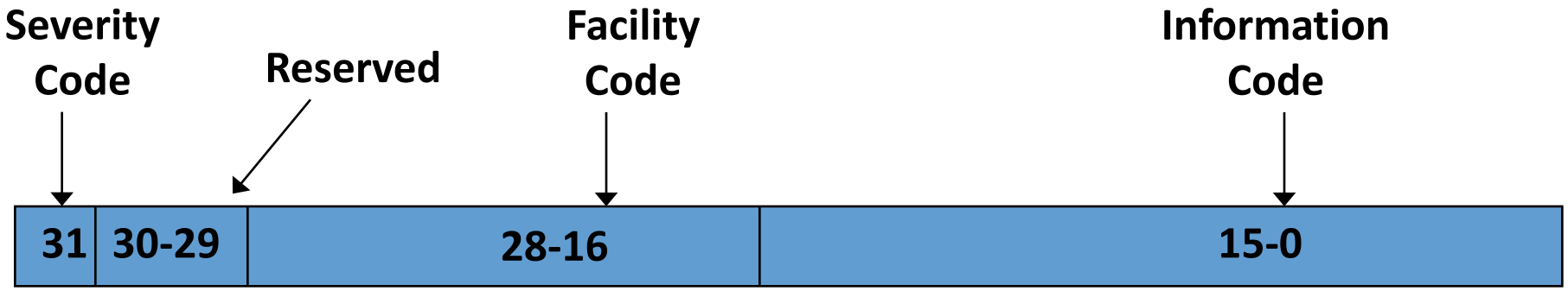
```
interface ICarStats : IUnknown {
 [propget] HRESULT Name([out, retval] BSTR* val);
 [propput] HRESULT Name([in] BSTR val);
};
```

COM Operation Invocations

- Invocation is defined by client objects
- Invocation determines
  - Interface pointer of server object
  - Name of invoked operation
  - Actual parameters
- Invocation is executed synchronously
- Invocation can be defined
  - statically
  - dynamically
- Clients have to interpret HRESULTS (return error code)!

COM: HRESULTS

- All COM API functions returns an error code of type **`HRESULT`**
- **`HRESULTS`** are 32-bit integers
- Structured into four fields



Programming
COM
Clients

COM Clients

- Any program can be a COM client.
- The basic steps in any COM client are:
  1. Initialize COM
  2. Get a pointer to the wanted COM server object
     1. Get a pointer to the Class Object
     2. Get a pointer to the relevant COM server object.
  3. Use the COM server object
  4. Call release on the COM server object
  5. Uninitialize COM

A C++ COM Client

```
int _tmain(int argc, _TCHAR* argv[])
{
  // Declare variables (not shown here)
  CoInitialize(NULL);   // Bootstrap COM.

  // Using CoGetClassObject().
  hr = CoGetClassObject(CLSID_ComCar, CLSCTX_INPROC_SERVER,
    NULL, IID_IClassFactory, (void**)&pCF);
  hr = pCF->CreateInstance(NULL, IID_ICar, (void**)&pCar);

  // You may replace the 2 statements above with
  // hr = CoCreateInstance(CLSID_ComCar, NULL, CLSCTX_INPROC,
  // IID_ICar, (void**)&pCar);

  // Use the COM server
  pCar->SpeedUp(10);
  hr = pCar->QueryInterface(IID_IRadio, (void**)&pRadio);
  pRadio->CrankTunes();

  // Clean up
  if(pCar != NULL) pCar->Release();
  if(pCF!= NULL) pCF->Release();
  if(pRadio!= NULL) pRadio->Release();
  CoUninitialize();
```

Demo

- Study the demo project **`COMdemo`** for details regarding coding a COM server by hand.

References & Links

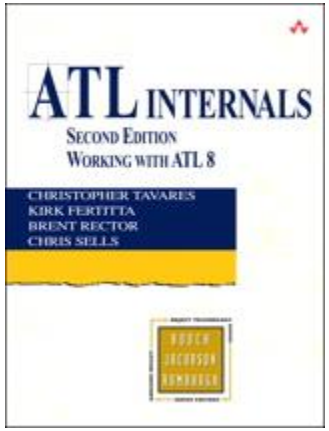
- Wikipedia
  http://en.wikipedia.org/wiki/Component_Object_Model

- The COM specification
  http://www.daimi.au.dk/~datpete/COT/COM_SPEC/pdf/com_spec.pdf

- The COM Programmer's Cookbook
  http://msdn.microsoft.com/en-us/library/ms809982.aspx

- MSDN COM Guide
  http://msdn.microsoft.com/en-us/library/ms690156(v=vs.85).aspx

Literature



- **ATL Internals**

  By Christopher Tavares, Kirk Fertitta, Brent E. Rector, Chris Sells
  Published 2006 by Addison-Wesley Professional.
  ISBN-13: 978-0-321-46793-5