What is COM?
- COM = Common Object Model.
- Platform-independent, distributed OO system for client-server implementations.
- COM objects can be created in a variety of languages (like CORBA).
- Not an object-oriented language but a standard (like CORBA).
- COM specifies an object model and programming requirements that enable COM objects (also called COM components) to interact with other objects.

Principles 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 Design Principles
- **Encapsulation**:
  - black box - no leakage of implementation details
  - all object manipulation through strict interfaces
- **Polymorphism**:
  - via multiple interfaces per class
  - “discoverable”: QueryInterface

Microsoft IDL (MIDL)
- Language for expressing all COM concepts.
- MIDL is:
  - programming-language independent
  - evolved from OSF/RPC IDL
  - not computationally complete
  - Different programming language bindings are available.

Example (UML Diagram)
Interfaces

- IUnknown
- AddRef
- Release
- QueryInterface
- IDispatch
- GetDispIDs
- GetTypeInfo
- GetTypeInfoCount
- Invoke
- Custom Interfaces

COM Interfaces

- UUID
- Objects
- References to COM objects are called interface pointers.
- Interface pointers refer to main memory locations.
- The references are persistent.

COM Interface Impl. in C++

#include "soccer.h"

class TrainerPlayer : public Itrainer, public IPlayer {
private:
    char* name; // name is the same in Itrainer & IPlayer
    short Number; // for IPlayer
protected:
    virtual ~TrainerPlayer(void);
public:
    TrainerPlayer(void);
    IMPLEMENT_UNKNOWN(TrainerPlayer)
    BEGIN_INTERFACE_TABLE(TrainerPlayer)
    IMPLEMENT_INTERFACE(ITrainer)
    IMPLEMENT_INTERFACE(IPlayer)
    END_INTERFACE_TABLE(TrainerPlayer)
    void train(); // Itrainer method
    void book(); // IPlayer method
};

COM Attributes

- COM does not support direct attribute specification.
- Attributes must be represented as set and get operations by the designer.
- COM has a keyword to designate this.
- Example:

```
interface IOrganization : IUnknown {
    [propget] HRESULT Name([out] BSTR val);
}
```

COM Operations

- Parameter kind
- Parameter list
- Parameter, e.g. Interface pointer
- Return value indicating success/failure
- Operation name
- Interface, e.g. IClass

Instances of COM implementations.
- References to COM objects are called interface pointers.
- Interface pointers refer to main memory locations.
- The references are persistent.
COM Hresult
- HRESULTS are 32-bit integers
- Structured into four fields

<table>
<thead>
<tr>
<th>Severity Code</th>
<th>Reserved</th>
<th>Facility Code</th>
<th>Information Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30-29</td>
<td>28-16</td>
<td>15-0</td>
<td></td>
</tr>
</tbody>
</table>

COM Operation Invocations
- Invocation provides:
  - 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!

Three Implementations of Requests
- Inter-process call with light-weight RPC
- Method call
- Remote call with real RPC

DCOM Architecture
- Communication details handled by COM runtime.
**Proxy & Stub**

- Client process
- Co-Process Object
- Interface Adder
- Local Server Process
- Local Object
- Local Class
- Remote Machine
- Remote Server Process
- Remote Object
- Remote Class

**DCOM Wire Protocol**

- Client Machine
- Server Machine
- Component
- COM Runtime
- TCP, UDP
- SPX, IPX
- NetBEUI
- HTTP

**DCOM Security**

- Secure - security is designed and built in, not an option.
- DCOM uses the extensible security framework provided by Windows NT.
- Security configurable
  - DCOM stores Access Control Lists for components
  - ACLs can be configured using the DCOM configuration tool (DCOMCNFG) or programmatically using the Windows NT registry and Win32 security functions.

**COM Security Architecture**

- Client Machine
- Server Machine
- Component
- COM Runtime
- NTLM
- SSL, Certs.
- NT Kerberos
- DCE

**Components & Reuse**

- Use existing tools and components.
- Reduce development time and cost.
- COM components easily configured as DCOM components.
- COM can use many other components.
- COM components are usable by many technologies.

**Location Transparency**

- COM object locations are stored in registry.
- Applications make calls using the interface proxy.
- Path to COM server or remote computer to run DCOM server is not needed by the application.
Connection Management

- Low bandwidth:
  - header is 28 bytes over DCE-RPC
  - keep-Alive Messages bundled for all connections between machines.
- COM employs an efficient pinging protocol to detect if clients are active.
- COM uses reference counting mechanism to do garbage collection.

Efficient & Scalable

- Multiplexing - single port per-protocol, per server process, regardless of # of objects
- Scalable - connection-less protocols like UDP preferred
- Established connection-oriented (TCP) sessions reused by same client

Platform Neutrality

- DCOM run-time is available for various platforms:
  - Win32 platforms, Solaris, DEC UNIX, HPUX, Linux, MVS, VMS, Mac
  - cross-platform interoperability standard
- Per-platform binary standard:
  - unlike Java, DCOM can utilize powerful platform-specific services and optimizations
  - less abstraction layers prevents additional overheads

Basic Principles