COM from C++, history and now

June 6, 2015

Chapter 1

Introduction

1.1 Introduction

- Disclaimer
- Initial situation
- Problem
- Solution
- COM currently

1.2 Goal

- Review COM's history
- Review COM from a modern C++ perspective
- Remove most unfamiliarity with COM
- Not a COM tutorial

Chapter 2

Disclaimer

2.1 Disclaimer

- I took great care in telling the truth in all of the following
- Yet, all of the following may be false
- Please correct me if I am wrong
- MSDN appears to be an excellent reference, but I could not find a single online COM tutorial that was correct C or C++

Chapter 3

Initial situation

3.1 1993

- Win16 API used in Windows 3.1
- Win32 API in development (would ship with Windows 95)
- Dynamic Data Exchange (DDE) used for interprocess communication, for example copy and paste
- C dominates in both API's
- C++ young: available since 1983, but without Standard

Chapter 4

Problem

4.1 Problem

- Complexity barrier was getting closer
  - C coding standards matured
  - Could C++ help?
- Language neutral communication between programs desired
  - Programmer A calls COM object B from Visual Basic
  - COM object B was written in C[1]
- Windows code was big, re-use of possibly-suboptimal interface preferred over rewrite

[1] pun intended

Chapter 5

Solution

5.1 COM

- Component Object Model
- Allows compiling of functions to a DLL from any language
- Makes these functions in DLL callable from any language
- Still in use, predecessor of, among others, DCOM, COM+, .NET

5.2 Common interface

Any language	DLL functions callable from any language
Any language	IDL[1] interface between any language and DLL
C	C interface between IDL and C++
C++	C++ implementation of C interface

5.3 IDL

```
import "myclass.h","unknwn.idl";
[
  object,
  uuid(12345678-90ab-cdef-1234-567890abcdef),
] interface IMyClass : IUnknown
{
  HRESULT SayHello();
  HRESULT Swap([in,out] int * x, [in,out] int * y);
  HRESULT IsPassword(
    [in] char * s, [out] int * is_password);
};
```

5.4 C interface declaration

```
#include <windows.h>
interface IMyClass : public IUnknown
{
  public:
  HRESULT STDMETHODCALLTYPE QueryInterface(
    REFIID riid, PVOID * object);
  ULONG STDMETHODCALLTYPE AddRef();
  ULONG STDMETHODCALLTYPE Release();
  HRESULT SayHello();
  HRESULT Swap(int * x, int * y);
  HRESULT IsPassword(char * s, int * is_password);
};
```

5.5 C-like interface: reference counting

```
ULONG STDMETHODCALLTYPE IMyClass::AddRef()
{
  return reference_count++;
}

ULONG STDMETHODCALLTYPE IMyClass::Release()
{
  --reference_count;
  if (reference_count == 0) delete this;
  return reference_count;
}
```

5.6 C-like interface: dynamic_cast

```
HRESULT STDMETHODCALLTYPE IMyClass::QueryInterface(
  REFIID riid, PVOID * object)
{
  if ( riid == IID_IMyClass)
  {
    *object = this;
    this->AddRef();
    return NO_ERROR;
  }
  else
  {
    *object = 0;
    return E_NOINTERFACE;
  }
}
```

5.7 C-like interface: custom functions

```
HRESULT IMyClass::SayHello() {
  std::cout << "Hello\n";
}

HRESULT IMyClass::Swap(int * x, int * y) {
  const int tmp = *x;
  *x = *y;
  *y = tmp;
}

HRESULT IMyClass::IsPassword(
  char * s, int * is_password) {
  *is_password
    = MyClass::IsPassword(std::string(s));
}
```

5.8 C you need

- Win32 API uses Hungarian notation heavily
- Hungarian notation is useful in C, but to be avoided in C++[2][3]

HRESULT	Result (error message)	Unsigned long
PVOID	Pointer to void	void *
PVOID *	Pointer to pointer to void	void **
REFIID	Reference to IID	IID *
IID	Interface ID	GUID
GUID	Globally Unique Identifier	a C-style struct
ULONG	Unsigned long	unsigned long

[2] Bjarne Stroustrup's C++ glossary: 'Hungarian notation - [...] It is totally unsuitable for C++ where it complicates maintenance and gets in the way of abstraction'

[3] Herb Sutter, Andrei Alexandrescu. C++ coding standards. Item 0, example 3: 'Therefore, no C++ coding standard should require Hungarian notation, though a C++ coding standard might legitimately choose to ban it

5.9 C you need

- Instead of std::string: 'char*' or 'wchar_t*'
- 'wchar_t*' is used often as a return type
- Instead of (derived) class: 'void*'
- Work with std::wstring, std::wcout, wcsstrlen, etc. instead

```
std::wcout << L"my_class says: "
  << my_coclass->SayHello() << L"\n";
```

Chapter 6

EOF