Development of CAPE-OPEN Compliant Process Modeling Components in Microsoft .NET

William M. Barrett, Jr PhD, PE

Engineer

06'AIChE – 3rd US CAPE-OPEN conference – Paper 489e

16 November 2006

Current CAPE-OPEN MiddleWare Specifications

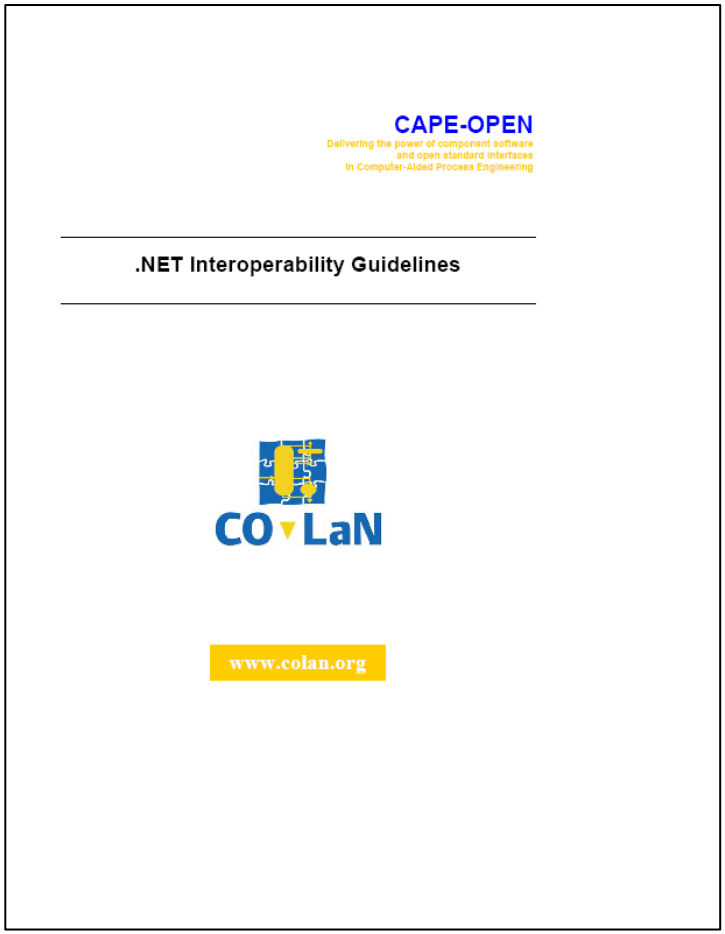
- Component Object Model (COM)
  - Primarily Windows-Based
  - An outgrowth of Object Linking and Embedding (OLE)
  - Active X Controls
- Common Object Request Broker Architecture (CORBA)
  - More Common on UNIX Systems
  - Different Object Brokers May Be Incompatible

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

CAPE-OPEN and .NET

Recently, a draft document related to CAPE-OPEN/.NET Interoperability was prepared and is available at the CO-Lan Web site: http://www.co-lan.org



RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

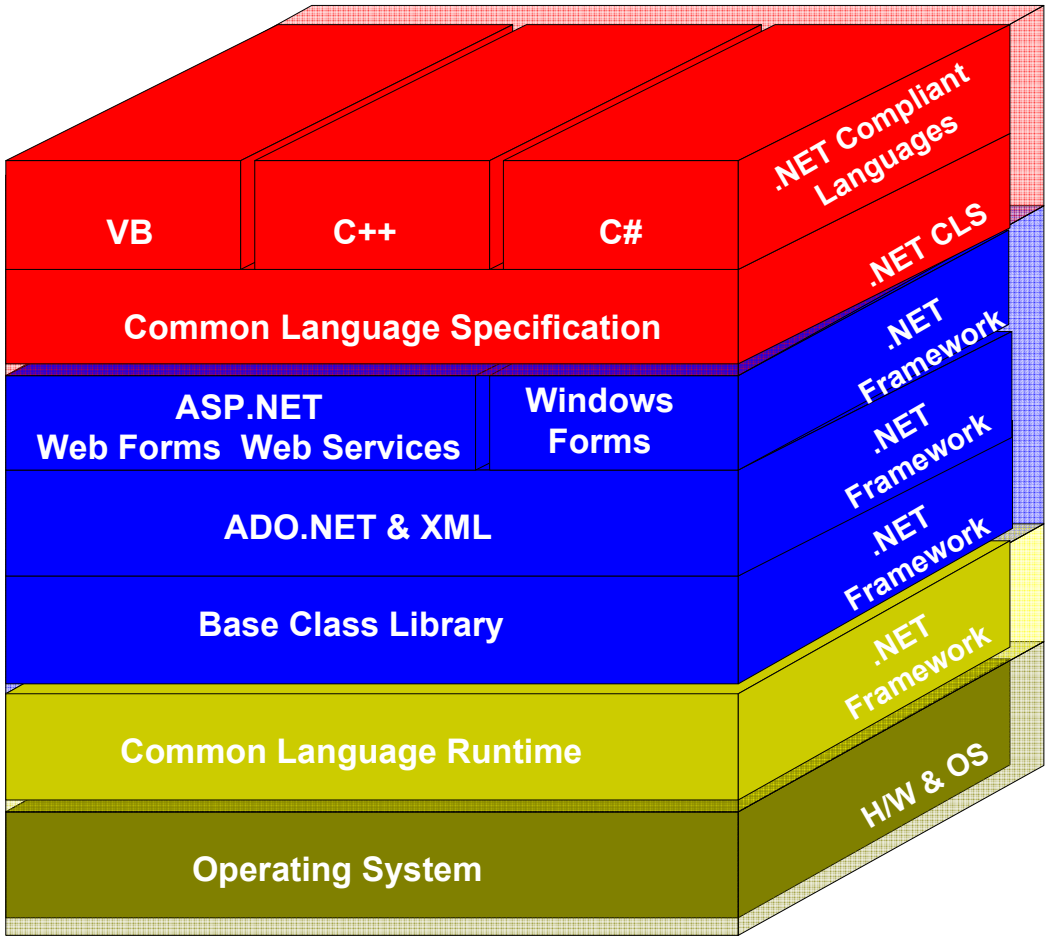
What is .NET

- .NET is an “update” of Microsoft’s COM
- Open Standards:
  - Ecma-334*: C# Specification
  - Ecma-335*: CLI Specification – Virtual Machine and XML-based class library
- Unix-based Open Source Implementation - Mono

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

.NET Framework Architecture



RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

How Does .NET Differ from COM?

COM

- DLL Hell
- Reliance on System Registry
- HRESULTS

.NET

- Versioning
- Side-by Side Execution
- XCopy Deployment
- Security
- Platform Neutrality
- Strongly Typed (CTS)
- XML/Web Services
- Garbage Collection
- Structured Exceptions

Native Code Interoperation Platform Invocation

- Creates a .NET wrapper that calls the function in the .dll
- Example (C#):

```
class ThermoWrapper
{
        [DllImport("thermo.dll")]
        internal static extern int calc_flash(double p, [In, Out] double[] z, double v, IntPtr user_data);
}
```

- A call of ThermoWrapper.calc_flash will invoke the .dll function.
- Allows use of any .dll file created in any language.

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

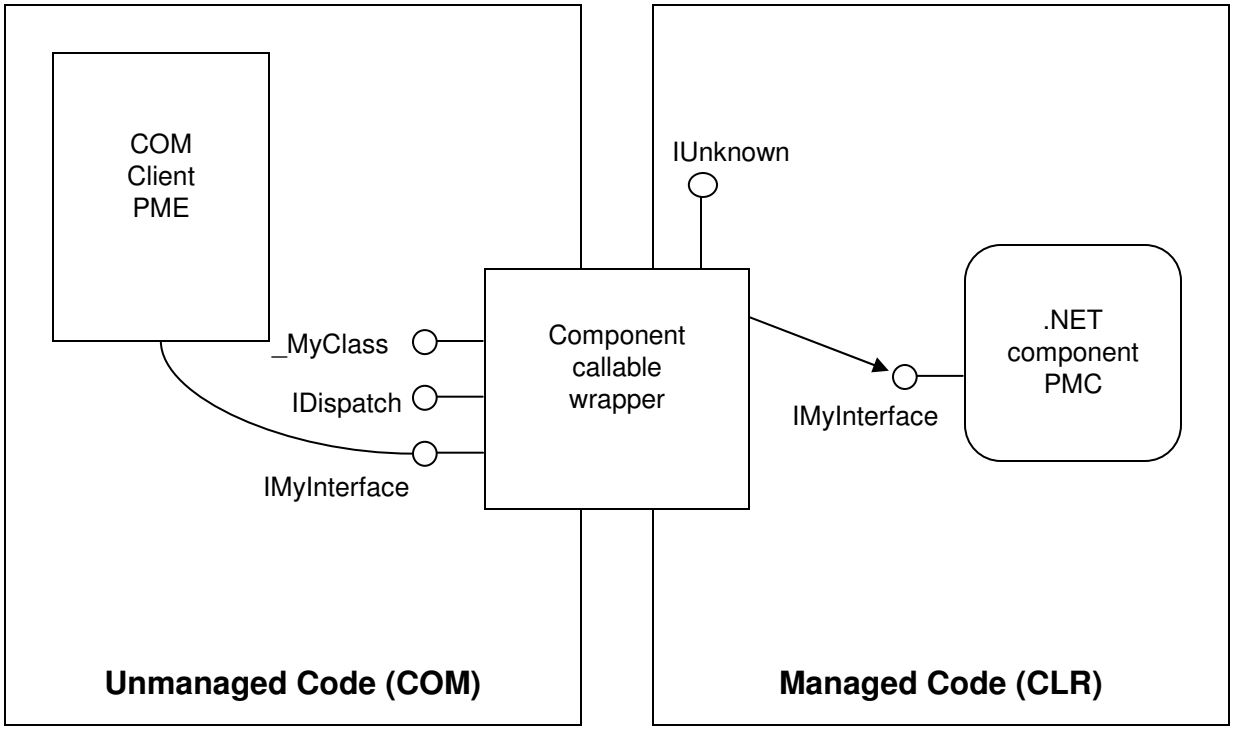
COM/.NET Interoperation

- "It Just Works" (In General, Mostly, But In Practice, CAPE-OPEN Does IJW)
- Creating A Primary Interop Assembly – Strongly Named Types In The PIA
- Using PIAs - Creates A "Native" .NET Version Of The CAPE-OPEN Interfaces That May Not Be Quite "Good Enough"

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

Use of a .NET PMC in Cape-Open PME



RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

Interoperability Issues

- Data Type Translation
- Error Handling/Exceptions
- Collections
- Persistence
- Object Registration

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

Comparison Of Data Types

CAPE-OPEN Data Type	Description of Data Type	COM Data Type	.NET Data Type
CapeLong	long	long	long
CapeShort	short	short	short
CapeDouble	double	double	double
CapeFloat	float	float	float
CapeBoolean	boolean	VARIANT_BOOL	bool
CapeChar	char	BYTE	char
CapeString	String	BSTR	String
CapeDate	string date	VT_DATE	DateTime
CapeURL	URL string	BSTR	String
CapeVariant	Container of any other type	VARIANT	Object
CapeInterface	CO Interface	LPDISPATCH	Object UnmanagedType::IDispatch
CapeArray(TYPE)	Array of Type	VARIANT containing Safearray(Type)	Array<Type> downcast to an Object

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

CAPE-OPEN Types

- CAPE-OPEN defines a number of enumeration types such as the CapeValidationStatus enum
- Interfaces Are Also Types
- Creating a PIA results in creating a namespace (*e.g.* CAPEOPEN093 from CAPE-OPENv0-9-3.tlb)
- CAPEOPEN093.CapeValidationStatus *IS NOT THE SAME AS* CAPEOPEN100.CapeValidationStatus
- CAPEOPEN093.ICapeParameter *IS NOT THE SAME AS* CAPEOPEN100.ICapeParameter

  Even Though the GUIDs Are The Same (Same COM Type)

Comparison of Interfaces

C++/CLI Interface Class

```
//This interface provides the basic functionality for a Unit
//Operation component
// CAPE-OPEN v1.0
[
   ComVisibleAttribute(true),
   Guid("678c0998-0100-11d2-a67d-00105a42887f"),//ICapeUnit_IID),
    System::ComponentModel::DescriptionAttribute("ICapeUnit Interface")
]
public interface class ICapeUnit
{
// Get the collection of unit operation ports
//
//  CAPE-OPEN exceptions:
//  ECapeUnknown,  ECapeFailedInitialisation,  ECapeBadInvOrder
[DispIdAttribute(1), System::ComponentModel::DescriptionAttribute("Gets the whole list of ports")]
property Object^ ports
{
    [returnvalue: MarshalAs(UnmanagedType::IDispatch)]
    Object^ get();
};

// Gets the flag to indicate unit's validation status
//· notValidated(0),invalid(1) or valid(2)
//
// CAPE-OPEN exceptions
// ECapeUnknown, ECapeInvalidArgument
[DispIdAttribute(2), System::ComponentModel::DescriptionAttribute("Get the unit's validation status")]
property CapeValidationStatus ValStatus
{
    CapeValidationStatus get();
};

//  Executes the necessary calculations involved in the unit
// operation model
//
//  CAPE-OPEN exceptions raised:
//  ECapeUnknown,  ECapeBadInvOrder,  ECapeOutOfResources,  ECapeTimeOut,
//  ECapeSolvingError,  ECapeLicenceError
[DispIdAttribute(3), System::ComponentModel::DescriptionAttribute("Performs unit calculations")]
    void Calculate();
// Validate that the parameters and ports are all valid
//
//  CAPE-OPEN exceptions:
//  ECapeUnknown,  ECapeBadCOParameter,  ECapeBadInvOrder
    [DispIdAttribute(4), System::ComponentModel::DescriptionAttribute("Validate the Unit"), returnvalue: MarshalAs(UnmanagedType::VariantBool)]
    bool Validate(String^ %message); //  C# bool Validate (ref string message)
};
```

COM IDL

```
[
    object,
    uuid(ICapeUnit_IID),
    dual,
    helpstring("ICapeUnit Interface"),
    pointer_default(unique)
]
interface ICapeUnit : IDispatch
{
// Get the collection of unit operation ports
//
//  CAPE-OPEN exceptions:
//  ECapeUnknown,  ECapeFailedInitialisation,  ECapeBadInvOrder
    [propget, id(1), helpstring("Gets the whole list of ports")]
    HRESULT ports([out, retval] CapeInterface* ports);

// Gets the flag to indicate unit's validation status
//· notValidated(0),invalid(1) or valid(2)
//
// CAPE-OPEN exceptions
// ECapeUnknown, ECapeInvalidArgument
   [propget, id(2), helpstring("Get the unit's validation status")]
    HRESULT ValStatus([out, retval] CapeValidationStatus *valStatus);

// Executes the necessary calculations involved in the unit
// operation model
//
//  CAPE-OPEN exceptions raised:
//  ECapeUnknown,  ECapeBadInvOrder,  ECapeOutOfResources,  ECapeTimeOut,
//  ECapeSolvingError,  ECapeLicenceError
   [id(3), helpstring("Performs unit calculations")]
   HRESULT Calculate();

// Validate that the parameters and ports are all valid
//
//  CAPE-OPEN exceptions:
//  ECapeUnknown,  ECapeBadCOParameter,  ECapeBadInvOrder
    [id(4), helpstring("Validate the Unit")]
    HRESULT Validate([ACTUALLYout] CapeString* message, [out, retval] CapeBoolean* isValid);
};
```

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

Error Handling

- CAPE-OPEN/COM Issue:
  - Does not use COM's GetErrorInfo API
- .NET Exceptions are translated to COM ErrorInfo Objects
- CAPE-OPEN requires Objects to expose error interfaces that they support.
- Interoperation requires Objects to implement appropriate error interfaces

Collections

- Use a generic .NET collection such as ArrayList
- Expose CAPE-OPEN collection interface(s)
- Remember that CAPE-OPEN collections are 1-based
- Test Item method argument to see if it is a 16-bit integer, 32-bit integer or String
- You can access the collection using COM enumerators

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

Persistence

- .NET uses “Serialization” to save an object
- All objects must be marked “Serializable”
- To be COM Persistable, one of the COM Persistence interfaces must be implemented.

Registration

- .NET compiler option to register object for COM
- The object must include a “RegisterFunction” to insert it into the CAPE-OPEN categories
- Registration-free COM not tested

RESEARCH & DEVELOPMENT

Building a scientific foundation for sound environmental decisions

Status of CAPE-OPEN and .NET

- Open Issue
- Few CAPE Software Vendors Have Near-Term Plans to Move to .NET
- Microsoft recommends that developers use the .NET Framework rather than COM for new development.
- Draft Interoperability Guidelines