17th European Symposium on Computer Aided Process Engineering – ESCAPE17
V. Plesu and P.S. Agachi (Editors)
© 2007 Elsevier B.V. All rights reserved.

An Overview of the Interoperability Roadmap for COM/.NET-Based CAPE-OPEN

William M. Barrett[a], Michel Pons[b], Lars von Wedel[c], and Bertrand Braunschweig[d]

[a] *US Environmental Protection Agency, 26 W Martin Luther King Drive, Cincinnati, OH 45268, USA, barrett.williamm@epa.gov*
[b] *CO-Lan, c/o Institut Français du Pétrole, 1&4 Avenue de Bois Preau, 92852 Rueil-Malmaison Cedex, France, cto.co-lan@tiscali.fr*
[c] *AixCAPE e.V. , Turmstrasse 46, 52064 AACHEN, GERMANY, vonwedel@aixcape.org*
[d] *Institut Français du Pétrole, 1 & 4 avenue de Bois Préau, F-92500 Rueil Malmaison, France, Bertrand.BRAUNSCHWEIG@ifp.fr*

Abstract

The CAPE-OPEN standard interfaces have been designed to permit flexibility and modularization of process simulation environments (PMEs) in order to use process modeling components such as unit operation or thermodynamic property models across a range of tools employed in the lifecycle of chemical process systems engineering. Technical foundations of interoperable software are constantly changing and Microsoft is nowadays declaring .NET and a successor to COM which has been the major platform for numerous CAPE-OPEN components so far. In order to ensure that the CAPE-OPEN idea will be applicable to recent technical changes, the COLaN has gathered experiences in the area of CAPE-OPEN implementations making use of .NET. This paper will demonstrate that CAPE-OPEN can be successfully implemented using .NET development tools and highlight how the CAPE-OPEN development can benefit from these new technologies.

Keywords

CAPE-OPEN, Component Object Model (COM), Microsoft .NET, Interoperability

1. Introduction

At present, there stands significant experience demonstrating the success of the CAPE-OPEN standards utilizing Microsoft's Component Object Model (COM)-based process modeling components (PMCs) within COM-based process modeling environments (PMEs). Indeed, today a number of major Microsoft Windows-based process simulation applications utilize CAPE-OPEN interfaces to enable plug-and-play use of third party PMCs. As a result of Microsoft's update of COM to the .NET Framework and the likelihood that COM will no longer be supported, the CAPE-OPEN Laboratories Network (CO-LaN) has evaluated the use of .NET within the CAPE-OPEN standardization process with the objective to explain new paths for integrating process modeling software to the CAPE community.

During the summer of 2006, the CAPE-OPEN Laboratories Network (CO-Lan) created a document entitled ".NET Interoperability Guidelines"[1], which is available through the CO-LaN website, www.colan.org. Preparation of this document benefited greatly from the experience of the authors in development work involved in creation of a process modeling environment (PME) based on .NET at the USEPA[2] and in migration of unit operation and thermodynamic models from academic research into CAPE-OPEN compliant process modeling components (PMC) at AixCAPE. The major result is that interoperability based upon .NET is a workable solution for the period of time in which COM and .NET implementations will exist besides each other.

This paper presents an overview of the CAPE-OPEN-specific interoperability guidelines prepared and published by CO-LaN that provide developers with insight into how to implement various CAPE-OPEN functionality in .NET. It briefly introduces .NET as a jump start into this new technology, explains general interoperability between COM and .NET as well as particular issues discovered during the above mentioned development in interoperating CAPE-OPEN software modules across COM and .NET.

In order to assess the need for developing native .NET interface specifications for the CAPE-OPEN standards, the CO-LaN has launched a survey among the CAPE community. The results of this survey will influence the future technical basis of the CAPE-OPEN standard and are discussed further below. The contribution concludes with a discussion of future aspects of CAPE-OPEN standardization development and the relevance of .NET.

2. .NET Background

The Microsoft .NET framework was created during the late 1990s by Microsoft with several goals in mind, which includes the unification of the various development technologies being used to date (such as COM, Active Server pages (ASP), etc.); bringing an opponent to Sun's Java technology on the market; better coverage of mobile devices; simplifying application deployment of (fighting so-called DLL hell); and better response to security issues.

The major difference between .NET and previous object models was the use of managed code which is not executed by a physical processor in hardware, but by a virtual processor emulated by a virtual machine. The code to be executed by virtual machines resides in assemblies which resemble dynamic linked libraries (DLLs) but are equipped with metadata describing their identity, locale, version number, content, and many other things. The virtual machine of the Common Language Runtime (CLR) provides a type system which permits data and classes to be shared across software written in a variety of several programming languages. Additionally, using platform invocation services (P/Invoke), .NET can also interoperate with legacy DLLs.

The architecture of the .NET framework is based upon the open specification of the Common Language Infrastructure (CLI) that was ratified by the European Computer Manufacturer's Association (ECMA)[3] and has been submitted to the International Organization for Standardization (ISO). This standard has not only been used as a basis for implementing the Microsoft .NET framework, but also in other projects such as the Mono Project (http://www.mono-project.com/Main_Page) development platform (for various Unix operating systems variants) or Portable .NET (http://www.dotgnu.org/pnet.html).

The advantages of the .NET architecture is that security is improved because assemblies are signed and their identity is verifiable. Further, because assembly versions are unique, multiple assemblies can exist side-by-side, allowing the application to use the appropriate component version.

3. Interoperability Issues

As the majority of commercial implementations of CAPE-OPEN rely on Microsoft's COM as a supporting middleware platform, the ability to successfully utilize the existing COM interfaces to interact with existing PMEs must be demonstrated as well as the interoperability between .NET-based PMEs with existing libraries of PMCs. Testing revealed the primary issues data type conversions, error handling, collections, persistence, and object registration.

In general, the creation of a primary interop assembly from the COM-based CAPE-OPEN type library was sufficient to convert the existing CAPE-OPEN interfaces to .NET based interfaces. Of the COM data types used by CAPE-OPEN, Boolean and Variants were most problematic. The Boolean data type in COM uses 1 for true and 0 for false, whereas the .NET Boolean complies with C++ style, where true is non-zero and false is zero-valued. COM variants are converted to the Object data type in .NET. Problems were encountered with the variant-wrapped arrays used by CAPE-OPEN and in assuring the class objects were returned as IDispatch-based class objects. These issues are easily resolved through the use of .NET's *Marshall* class attributes.

CAPE-OPEN error handling uses COM-style HRESULT function returns, with additional error information obtained through error interfaces supported by the PMC returning the error HRESULT. This differs from the COM *GetErrorInfo* API supported by .NET's COM interop. In order to comply with CAPE-OPEN's error handling, when .NET exceptions are thrown by a PMC, the exception needs to indicate the appropriate error HRESULT and the PMC needs to implement the CAPE-OPEN error interface. Further, a .NET-based PME would need to obtain the CAPE-OPEN error interfaces from the PMC and use that information to create and re-throw an exception that can be caught and processed in the PME's .NET-based exception handling infrastructure.

Generic .NET collections in general support COM-based collection methods, and must support the CAPE-OPEN collection interfaces. The main issue is that the CAPE-OPEN collection index is 1-based (first item is index 1) where .NET collections are 0-based (first item index 0). Further, the CAPE-OPEN collection's Item method uses a COM variant argument that Visual Basic treats as either a 16-bit or 32-bit integer, so the .NET collection's implementation of the CAPE-OPEN Item method must test integer indices for both these types.

Persistence mechanisms were drastically changed between COM and .NET. .NET-based PMCs must support one of the COM-based persistence mechanisms indicated in the CAPE-OPEN persistence specifications. Further, a .NET-based PME should wrap a COM-based PMC in a serializable class and persist the PMC to a serializable stream.

Object registration is complicated by the need to place the object in the appropriate CAPE-OPEN component categories. This can be accomplished by instructing Visual Studio to register the class library for COM interoperation. In order to expose the object as a CAPE-OPEN-based PMC, the component must also be registered in the appropriate CAPE-OPEN categories, which is accomplished using a COM registration function that creates the appropriate CAPE-OPEN categories and adds the object to the categories.

4. Summary of Survey Results

Following preparation and publication of the draft interoperability guidelines, the CO-Lan conducted a survey of software vendors, end users and academics to determine their intentions regarding the use of .NET. The survey, sent to more than 300 organizations, probed for the interfacing technology used with other CAPE software, for the programming languages and development tools used, for the use of .NET framework or plan to use it. Current status and plans towards implementation of CAPE-OPEN interfaces constituted a second part of the survey. Most of the answers were from CO-LaN members.

The answers are related to 24 different software products, either PMEs or PMCs. The .NET framework is already used by only a few CAPE software developers having answered the survey. C++ is the preferred programming language with Visual Basic and FORTRAN also in use. Microsoft Visual

Studio is the only development tool for all software targeted at Windows platforms. Only two organizations listed Unix/Linux/Solaris as either their sole or alternate platform supported by their product. While most of the software products listed implement CAPE-OPEN interfaces, mostly COM based ones, proprietary interfaces to other CAPE software are also often implemented.

One software vendor, one already using the .NET framework indeed, described the development of .NET native interfaces by CO-LaN as critical while it was marked as important half a dozen times. Otherwise organizations listed this as not important and once as not necessary.

While the survey results indicate that there is currently no strong push from the CAPE community for the CO-LaN to create .NET native interfaces, CO-LaN needs to proactively identify and prepare for changes in the development environment relevant to CAPE. As the COM object model is being deprecated, CO-LaN's proactive evaluation of the .NET environment will allow a decision to be made regarding a roadmap for transitioning from COM to .NET based not only on the survey answers, but an analysis of how the discontinuation of COM and a transition to .NET will effect CAPE user and allow them to take advantage of adavances in information technology in the years to come.

5. CAPE-OPEN Road Map

The CAPE-OPEN ".NET Interoperability Guidelines" document provides a more detailed discussion of the added features associated with .NET, and shows that .NET objects can be readily used in a COM environment. Further, a .NET environment can also readily use objects created in COM. This demonstrated that interoperability is the first step in moving from one object model to another – ensuring legacy objects are supported. Clearly, interoperability, legacy support, and added features are important in evaluating the use of the .NET object model, but other issues remain, such as whether there will be a requirement for future changes in object model and will the new object model be robust enough to evolve as new technologies are developed and brought to bear on future problems.

As a starting point for this discussion, it should be recalled that "The first objective of the [CAPE-OPEN] partnership was to understand how software for designing and optimising process plants could be modified to make use more cost-effective by integrating software pieces one into another."[4] At present, there stands significant experience demonstrating the success of this endeavor as COM-based PMCs can now readily be utilized in a wide range of PMEs through the use of the CAPE-OPEN interface set. While this effort is not complete, some interface packages require little more than fine tuning of tested interface models while other interface packages are still in their infancy, future efforts should build upon past accomplishment. A clear consideration is that any changes to the object model build upon this experience, and .NET meets this criterion.

One key issue related to the continued use of COM is that COM is a proprietary technology created by Microsoft, and Microsoft is in the process of phasing it out. Microsoft's COM web page (http://www.microsoft.com /com/default.mspx) clearly states: "Microsoft recommends that developers use the .NET Framework rather than COM for new development." At this point, it should be noted that the need to consider a new object model is due to the reliance on a previous proprietary model and that the .NET Framework and the Common Language Runtime (CLR) are Microsoft proprietary models. The risk that the .NET object model will be deprecated or made obsolete is reduced by the fact that the Common Language Infrastructure (CLI), the C# programming language, and the C++/CLI programming language are open standards that have been accepted by the European Computer Manufacturer's Association (ECMA). Third-party implementations of both the CLI and C# language exist, such as the Mono Project (http://www.mono-project.com/Main_Page) and dotGNU (http://dotgnu.info/). These implementations can run not just in the Windows environment, but also on Linux/UNIX and Apple's Macintosh Operating System. The fact that open-source, shared source, and third-party implementations of the CLI (and therefore, the .NET Framework) exist reduces the risk that a new object model will be designed that will supplant this effort.

Another area that must be considered is the ability of the object model to evolve as new technologies are brought to bear on CAPE-related problems. Recently, CAPE-OPEN-based process simulation tools have been demonstrated on a parallel processing system. Technologies that one can readily expect process simulation applications to use, include advances in processor architecture and distributed applications. Microprocessor manufacturers and software developers are slowly moving away from 32-bit processors to 64-bit processors, which provide more addressable memory and faster computation.

At present, the .NET Framework, and ultimately the standardized CLI, appear to meet the needs of providing a relatively stable development platform for the foreseeable future. This architecture improves on issues related to COM development such as registration and security. Given the current state of the .NET Framework, and the third-party/cross platform implementations of the CLI, the .NET Framework is poised to be used as a replacement for COM.

Bibliography

1. CO-Lan (2006) .NET Interoperability Guidelines., CO-Lan. Accessible at www.co-lan.org.
2. Barrett, W. M.; Yang, J., Development of a chemical process modeling environment based on CAPE-OPEN interface standards and the Microsoft .NET framework. Computers & Chemical Engineering 2005, 30, (2), 191-201.
3. ECMA (2005). Standard Number 335, Common Language Infrastructure. Geneva, ECMA International.
4. Pons, M. (2003). "Industrial Implementations of the CAPE-OPEN Standard." AIDIC Conference Series 6: 253-262.