Using .NET CAPE-OPEN Unit Operations in an Applied Process Simulation Course

Peyton C. Richmond

Department of Chemical Engineering, Lamar University, Beaumont, Texas, USA

Abstract
This paper describes the author's experience teaching the CAPE-OPEN (CO) unit operation standard [1] in a graduate level "Applied Process Simulation" course at Lamar University during the summer 2008 semester. The purpose of this course was to educate students on process simulation skills required to model industrial systems. The students were asked to consider the case where a desired unit operation must be developed by a user because it is not available for their process simulator package. Rather than teaching a proprietary process simulator extension method that would only apply to a particular vendor's product, this instruction was based on the CO standard so that the lessons learned would be applicable to a variety of process simulation packages.

Students were instructed in the CO standard and related technology and then required to complete a laboratory assignment to copy and modify an existing unit operation process modeling component (PMC) project. The CO standard defines rules and interfaces that allow process simulator packages and other CAPE (Computer Aided Process Engineering) applications to interoperate with third-party components. Students were provided with a functional PMC project as a basis for building their CO unit operation. After they registered their version of the CO unit operation project, the students demonstrated its functionality in a commercial steady-state process simulation package licensed by Lamar University.

1. Introduction
The Lamar Chemical Engineering Department has incorporated a number of commercial process simulators into their curriculum to provide a realistic context for undergraduate and graduate students to learn about process engineering. Although students must learn the fundamental concepts in order to make higher level judgments and comparisons of process simulation results, after learning those concepts we feel that it is important for them to apply them using commercial process simulators similar to those used by operating companies and contractors to solve industrial problems.

In some cases the process simulator package does not include a unit operation appropriate to model the process under consideration. Say that a company has a proprietary model for such a unit operation and would like to use their model with a particular process simulator package. This unit operation model could be delivered as a self contained black box with a custom user interface that can be installed on the employee's computer and manually interfaced with the process simulator package. However, a better alternative is to develop an extension to the process simulator package so that the interface between the unit operation and the simulator package is handled automatically. The CO unit operation standard defines an interface that allows a CO extension to be seamlessly interfaced with any CO compliant process simulation package.

Using a CO extension method also allows a company to share its modeling technology without giving away its proprietary information unlike some other extension methods. A technology or operating company may also want to provide their modeling capability to a licensee as part of a license agreement or to a contractor for developing a design. It is even possible to incorporate a separate licensing strategy for a proprietary unit operation, thus providing access while still maintaining control of the technology. Certainly if the company wants to license its technology to users of various process simulation products, developing a CO standard based extension makes more sense than using a proprietary extension method. This technology also benefits CO compliant process simulation users because they can obtain the benefits of a third-party extension while using their process simulator of choice.

This standard is supported by the CAPE-OPEN Laboratories Network (CO-LaN) organization, a user group formed to maintain and promote CO usage and support ongoing development on the standard. The CO standard is currently based on the Microsoft Component Object Model (COM) which defines a binary interface between software components so that third-party components can be written in any COM compatible language [2]. The list of languages available to write user components also now includes any Microsoft .NET compatible languages including Visual Basic, C++, C#, and others. However, for a third-party CO component to be accessible in a process modeling environment, it must be registered as a COM object in the local Windows registry.

There are other examples where it is preferable to create a process simulator extension using the CO standard instead of the proprietary process simulator extension method. However, in a corporate setting the choice between using a proprietary method and creating a CO process simulator extension may also depend on the available in-house expertise. Some companies chose to use proprietary extension methods because they typically require less expertise initially; however, there remains a very convincing argument to invest in CO technology. In terms of the education of students in these types of technology, we favor the open standard of CO versus the proprietary nature of any given process simulator package.

2. Problem Description

At Lamar we are committed to the application of active learning technologies and have been updating our curriculum with problem based learning (PBL) and other cooperative learning strategies designed to improve student learning. In this vein several active learning activities are provided for students before the actual CO laboratory programming experience. The first short activity is based on the Microsoft COM technology which is used by a process modeling environment (PME) to locate and create instances of a CO PMC (e.g., a CO unit operation extension). This is followed by a second short activity involving Microsoft .NET technology. The final warm-up activity concerns CO background and its terminology.

In order to learn about and apply the CO standard the students were asked to develop a basic stream splitter unit operation and demonstrate that unit operation using two commercial process simulator packages licensed by Lamar University. The stream splitter unit operation was required to have one inlet stream connection and two outlet stream connections and include one real parameter to specify the split fraction of the inlet stream which should go to the first outlet stream with the remaining material going to the second outlet stream. Recognizing that a broad range of programming skills exists among graduate students and wanting to emphasize the process simulation skills rather than programming skills, a template-based approach was chosen to simplify the programming requirements for this exercise and allow more time to be spent on course objectives [3].

The template developed for this course was based on a class library created using the C++/CLI unit operation .NET framework example distributed by CO-LaN [1,4] as modified by the author for use in the AspenPlus or HYSYS [5] PME. Those framework example codes demonstrate the use of Microsoft Visual Studio 2005 .NET (VS2005) to develop a CO unit operation PMC. Some small modifications of this code were required to make these components more compatible with AspenPlus and HYSYS. The professional version of VS2005 was made available to each of the students on laboratory computers based on the University's site license. However, the author notes that the .NET examples used in this class could also have been demonstrated using the freely available VS2005 Express edition.

To demonstrate their modified CO PMC on a university laboratory computer each student must register their PMC in accordance with the CO standard. Typically, a modification to the Windows registry requires administrative access. However, due to security concerns, we were unable to provide student users with administrative access to academic lab computers. Also, students could not demonstrate their PMC's functionality using their personal computers because our process simulation software is restricted to University computers. To resolve these issues, a template-based approach was developed that allows students to re-register a CO PMC project without administrative rights. Each laboratory computer was first prepared under an administrative account as follows:

- First the template CO PMC project and its .NET executable were copied onto each lab computer.
- Then the executable was registered using *regasm* with the */codebase* option. This registration process created two new COM/CO registry entries for the template unit operation PMC.
- Finally, the permissions on only those two newly created registry entries were changed to provide students with Full Access.

These steps allowed each student to copy and then re-register the template CO PMC on a laboratory machine after modifying the project for this assignment. Note that they were only given permission to change the existing registry entries created by registering the template executable and were not given permission to create new registry entries.

3. Problem Implementation – 2008 Summer I Session

This section describes the CO learning activities used with this Applied Process Simulation course and the results of a survey following the programming lab exercise. For the first activity, the students were provided with simplified Microsoft COM documentation and each student was asked to generate their own Globally Unique Identifier (GUID). Then they were asked to answer the following questions:

- What does the COM acronym represent?
- What programming language must be used to create component objects?
- Describe at least four key features of COM.
- Describe at least five key attributes of interfaces and why they are important to COM.
- What is a GUID and how does this help COM?
- Name the interface and its three methods that are required for all component objects.
- What language is used to create interface definitions?

In the next activity the students were provided with simplified Microsoft .NET documentation and asked to answer the following questions:

- What is the .NET Framework and its CLR?
- What do the abbreviations CTS and CLS represent?
- What is the MSIL language?
- What is the difference between managed and unmanaged code?
- What is the “wrapper” that allows COM components to be accessed from the .NET Framework?
- What is the “wrapper” that allows .NET classes to be accessed from COM clients?
- How does the .NET Framework make it easier to create a COM object?

In the final activity the students were provided with the CO-LaN web URL (www.colan.org) and asked to review the General Information link and locate the CO documentation sets. The students were also instructed to review the EPA website located at www.epa.gov/nrmrl/std/mtb/p2tools/cape.htm because the underlying code of the library used in this course was based on that source. The students were pointed to specific tables in the “Thermodynamics and Physical Properties Interface Specification” CO documentation and asked the remaining questions:

- What is a Process Modeling Component (PMC)?
- What is a Process Modeling Environment (PME)?
- What is the CO-LaN?
- Locate the 1.0 Document set on the web site and note its size.

- Locate the ICapeThermoMaterialObject interface in the PDF
- List the methods for this interface and locate the Visual C++ prototype for its GetProp() method.
- Are the GetProp calls on page 85 of the PDF consistent with this prototype?
- Look at the Enumerations and determine the number of parameter types (CapeParamType), the possible port directions (CapePortDirection), and the number of port types (CapePortType).

In order to promote participation during these activities, random members of each team were asked to present their answers to the class following each activity. Each of these activities was also followed by a brief PowerPoint presentation to addresses any unanswered questions.

After the final activity the students were provided with the template CO PMC project which builds a stream mixer unit operation and asked to copy the project to their local VS2005 Project directory and, to demonstrate its initial functionality, change the name of one of the inlet ports and rebuild the project. The resulting modified PMC was re-registered by double clicking on the provided *register.bat* file followed by double clicking on the generated *.reg* file. Then they verified that their PMC functioned in both HYSYS and AspenPlus process simulation environments and that their PMC reflected the changes they had just made to the inlet port names. The students were then released to complete the CO lab by continuing to modify their team's PMC project into a stream splitter operation. Each team successfully completed the lab although some additional help was provided as requested by team members.

Afterwards, an anonymous survey was taken to gauge how the students felt about using CO technology in this course; the survey was completed by eleven of the twelve students taking this course. Only five of the students responding to this survey rated themselves as extremely proficient with MS Visual Studio 2005 and C++/CLI applications (see Figure 1); therefore, it is not surprising that five of the eleven respondents were unsure of their ability to create their own CO unit operation (see Figure 2) in the absence of their team members. Ten of the eleven respondents believed that the time spent on CO unit operations should be expanded to allow them to create a unit operation from scratch (see Figure 3). And it was gratifying to see that all respondents believed that this subject would be useful to their future chemical engineering careers (see Figure 4).

4. Conclusions

A CO exercise was incorporated into a graduate level Applied Process Simulation course to demonstrate how to implement a CO compliant unit operation extension for a commercial process simulator. The students responded well to the exercise and were able to modify a template CO unit operation project and validate its interoperability with the AspenPlus or HYSYS commercial process simulator environments. A template based approach was developed to allow our students to modify and register their own CO based unit operation without having administrative access to laboratory computers. We believe that the CO exercise described above is a valuable learning experience for our students and plan to continue to use it in this course.

5. Acknowledgments

The author wishes to express his appreciation to Michel Pons and the CO-LaN staff and to Dr. William Barrett for the code examples used as a basis for this exercise. The code examples were developed originally as training materials for a course titled "CAPE-OPEN with .Net" that was held on March 7, 2007 in Heidelberg, Germany. That course was organized and financed by CO-LaN.

6. References

1. URL: http://www.colan.org/; The CAPE-OPEN Laboratories network.
2. URL: http://www.microsoft.com/; Microsoft Corporation
3. Silverstein, D.L. (2003). Increasing Time Spent on Course Objectives when Using Computer Programming to Teach Numerical Methods. *Chemical Engineering Education*, **37**, No. 3, 214-218.
4. URL: www.epa.gov/nrmrl/std/mtb/p2tools/cape.htm; US Environmental Protection Agency Pollution Prevention Research Tools.
5. URL: http://www.aspentech.com/; Aspen Technology, Inc.

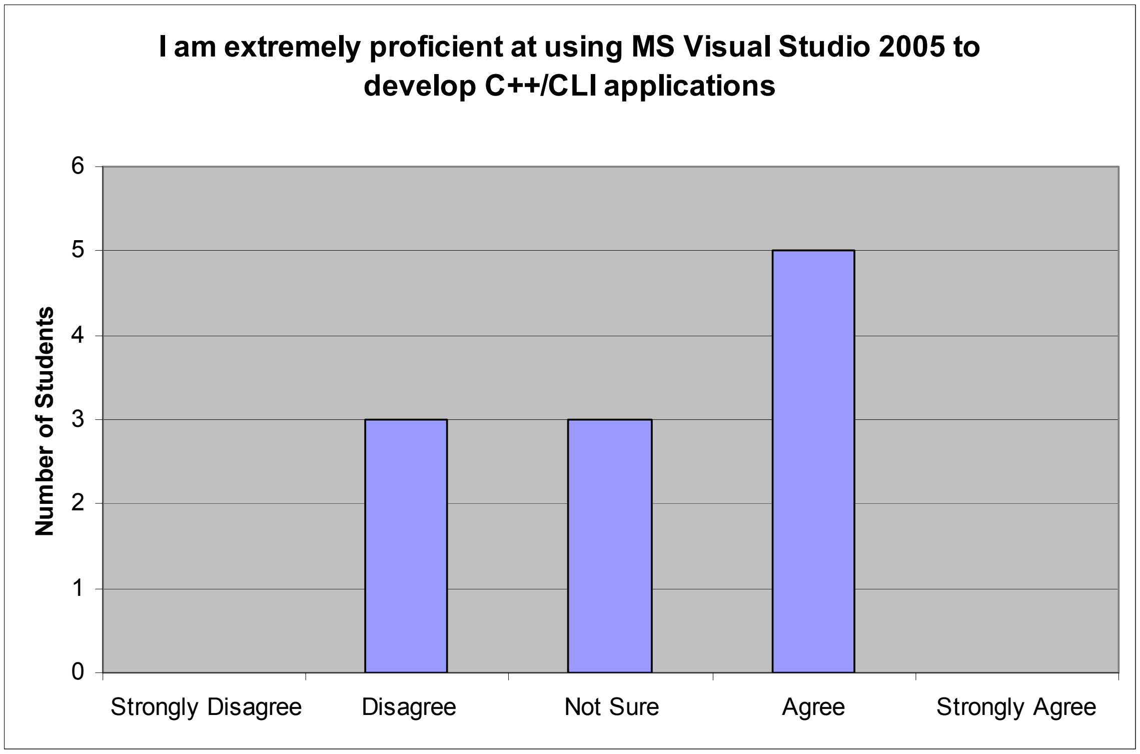

Figure 1. Student's perceived C++/CLI ability with VS2005.

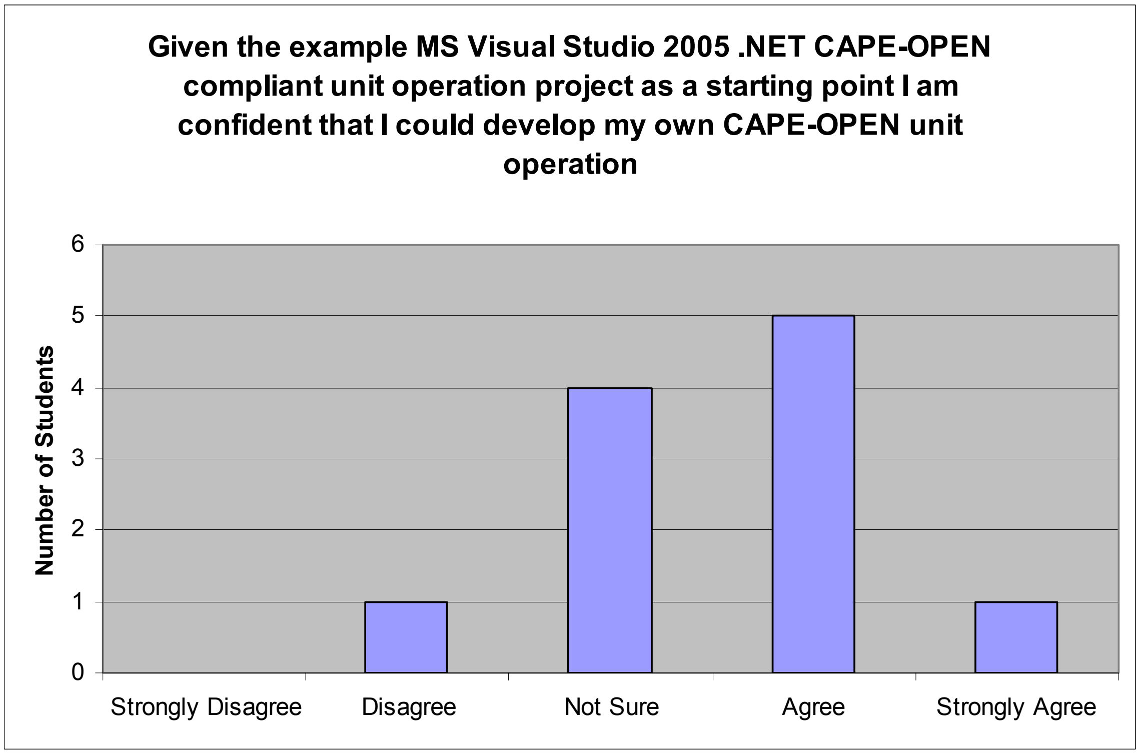

Figure 2. Perceived ability to re-create a CAPE-OPEN PMC.

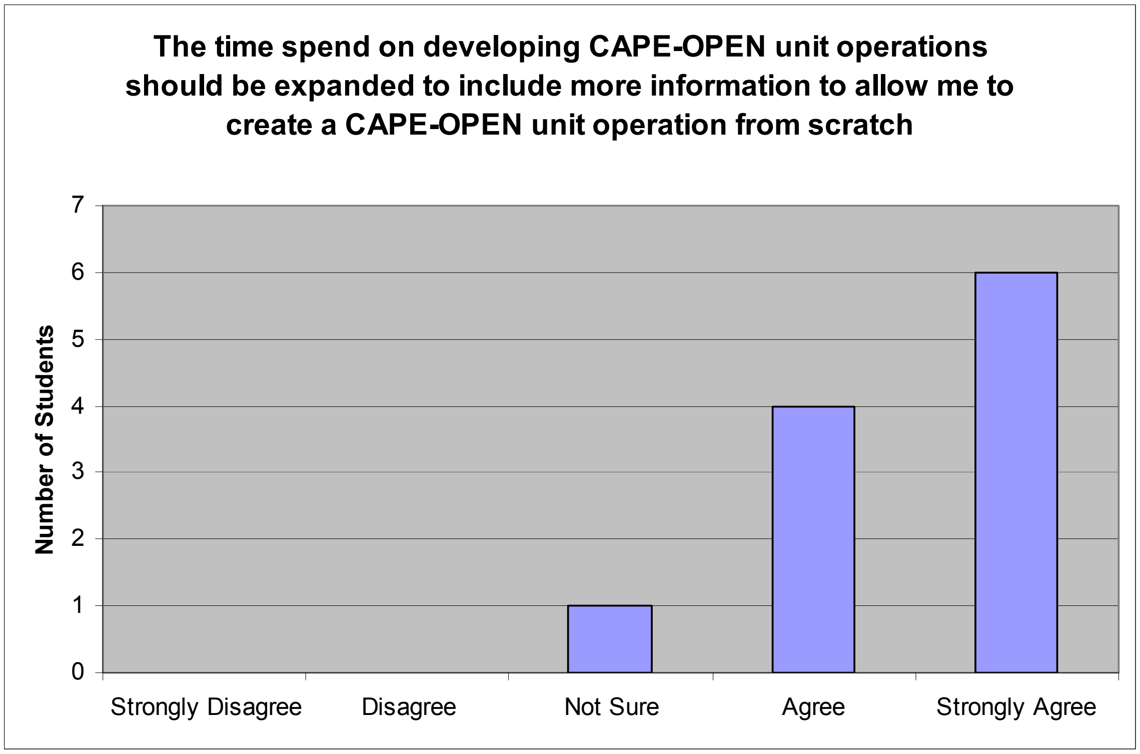

Figure 3. Attitude concerning expanding CAPE-OPEN PMC instruction.

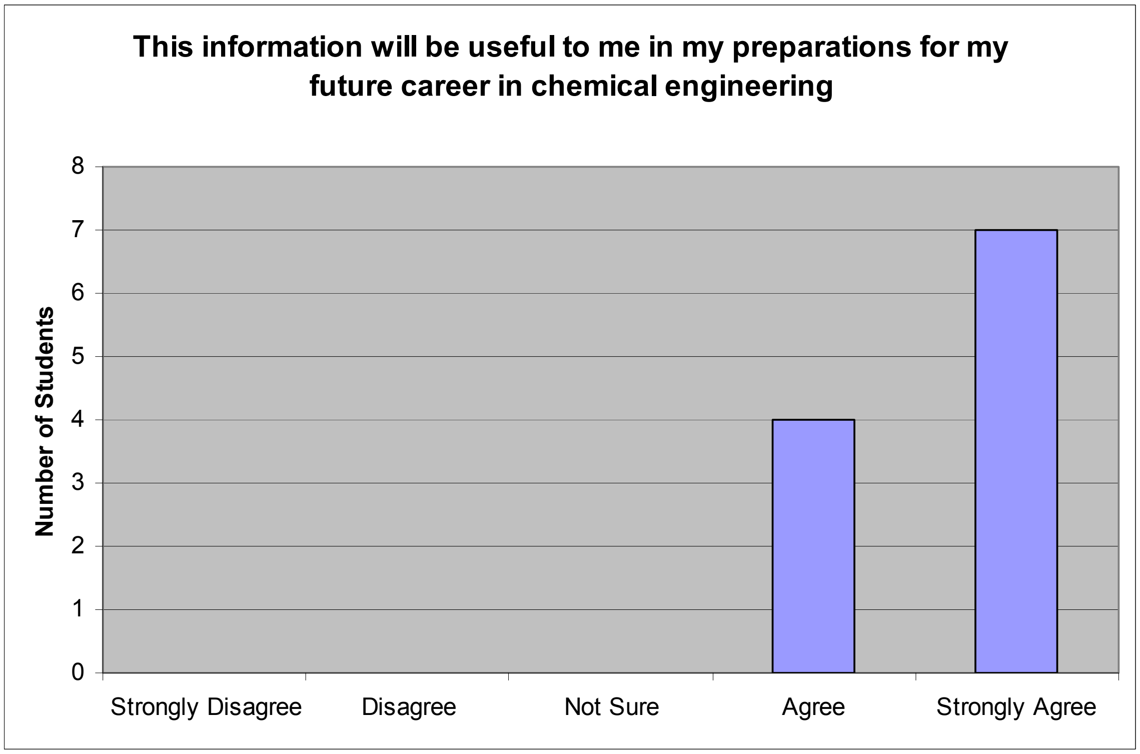

Figure 4. Attitude concerning CAPE-OPEN applicability to future career.