



Unit Operation Prototypes for the Development of CAPE-OPEN Refinery Reactors Interfaces

Ignasi Palou-Rivera, BP Refining Technology
Jeffrey Pribich, SAIC

4th CAPE-OPEN US Conference
2007 AIChE Annual Meeting, Salt Lake City UT
November 7 2007

Context



- March 2006 CO-LaN AGM: proposal for creation of Refinery Reactors Special Interest Group (RR SIG)
- June 2006 RR SIG kick-off meeting:
 - Agreement for creation of two prototypes for testing and demonstration of interfaces

Prototype	PME	PMC
#1	SimSci – Proll	Shell
#2	Aspentech – Refsys Honeywell – Unisim	BP

Prototype Characteristics



- BP Refining Technology to develop a PMC (Process Modelling Component) prototype
 - Unit Operation prototype
 - Property Setter/Calculator (RON)
 - Multiple input streams
 - One output stream
- } Mixer
- Architecture:
 - Standard CO UO shell (wizard) in VB6
 - Property calculation code from literature
 - Addition of petroleum fraction interfaces (RR SIG) by hand
 - Final prototype to have freely distributable source code

Development Process



- SAIC contracted to do the basic unit operation software development:
 - High software development expertise
 - New to CAPE-OPEN world
 - Dual purpose prototype:
 - Testing and development of RR SIG interfaces
 - SAIC to develop CAPE-OPEN expertise for future development

Development Results



- Results were much worse than expected:
 - Severely delayed.
 - Questionable code quality from wizard
 - Manual re-write from “some” code examples
- Probable re-write using .NET and EPA-created wrapping class

Code examples



- Several Associate members have sample code available
 - Part of PME installation
 - University/research institution code
- Issue of code ownership reusability from examples (COLaN or vendors)
 - Unclear (changing) licensing
- Is there a need for *canonical* code samples for several “common” modules?
 - Difficult
 - VERY useful
 - Breaking with CO tradition.

Observations from SAIC



- Since the current set of wizards produce template code that still requires a significant manual code effort to enable the CAPE-OPEN interface, a more simplified wizard utilizing .NET as a platform provides a good alternative. .NET contains embedded wizard to enable various languages to interface with MS products such as excel, word and access. A comprehensive interface template written in .NET would allow a modeler to extract the interface calls straight out of an object (written in some chosen language) that will glue together the I/O for the CAPE-OPEN template.
- One of the major areas in the code developed by the wizard that requires substantial hand reworking is in the call interfaces where the input and output streams to a unit are defined. The actual pieces require investigation into the mechanism for passing each parameter. Essentially, the inner working of the interface are not hidden and require the modeler to expose these inner workings of the interface.
- An object based interface would hide all of the CAPE-OPEN defined interface code and process and provide a layer to the modeler where a unit can be defined in simple terms (not just code terms). One of the advantages to using the object would be the ability to automatically generate an interface diagram/picture describing data flows into and out of any custom unit that utilizes CAPE-OPEN to interface with HYSYS (as an example). From that point forward, the modeler could describe the interface from the diagram.
- What this leads to is an object based CAPE-OPEN interface where a modeler could tie his custom unit together using a graphical/pictorial representation of the interface to tie with a third party software tool (HYSIS). The ability to tie code together using diagrams is available today in software tool.
- Today, the wizard does not provide a very comprehensible interface in the actual code - which is a general problem when using COM.

Conclusions



- Wizards have many intrinsic limitations:
 - Too many options
 - Support or not for optional interfaces/properties
 - Strong tie to specific language/version
- Most software developers prefer to have good source code examples than wizards
- CO-LaN should emphasize creation of complete example prototypes with source code available to new developers
- Move to .NET architecture (or at least current COM interfaces implemented using .NET) a very positive step
 - Wrapping classes