Petro-SIM Simulator and CAPE-OPEN: Experiences and Successes

Michael Aylott, KBC Advanced Technologies, Calgary, AB, Canada
Ben van der Merwe, KBC Advanced Technologies, Calgary, AB, Canada

Abstract

KBC Advanced Technologies is a leading independent consultant and software solutions provider to the energy industry. Our Petro-SIM simulator is a general purpose steady state graphical process simulator aimed at the Oil Refining and Petrochemical industries, offering a full suite of detailed reactor unit operations allowing rigorous simulation of refineries and petrochemical plants. It has long had extensibility mechanisms, allowing customers to plug in their own unit operations. KBC recently added support for the CAPE-OPEN Unit Operation interface to Petro-SIM.

This paper will discuss our experiences so far and share with you the successes we have had using third party heat exchanger modules. We will discuss details of the implementation technique and compare it against the traditional extension methods provided by the simulator.

Introduction

KBC Advanced Technologies (KBC) is a leading independent consultant and software solutions provider to the energy industry. Our process consulting practice has used simulation extensively since we started business in 1979, with our work taking us across the full spectrum of life cycle activities, from initial concept through to operating and revamp studies on existing plants. Our business core strengths are in the downstream oil refining sector, with much of our work historically focused on delivering profit improvements to refineries based on detailed modeling of their facilities.

As well as developing our own simulation programs (notably Petrofine and the Profimatics SIM Reactor Models), we also make use of commercially available simulation packages and specialized design tools. Our strategy has always been to develop and commercialize our own proprietary in-house technologies that differentiate us, using third-party tools to supplement this. We are probably unique among simulation software providers in making extensive use of our own simulation products within our consulting practice: the impact of this is that new features get adopted very quickly and we benefit from extensive feedback and expertise of our internal users.

We introduced our Petro-SIM simulator in 2005 as a successor to Petrofine and HYSYS.Refinery. Petro-SIM offers all the capabilities of PC-based graphical flowsheeting systems together with all of KBC's oil characterization technology and our comprehensive refinery reactor library.

Unit Operation Extension Methods in Petro-SIM

In common with other simulators, Petro-SIM has its own proprietary extension mechanisms for adding unit operation models. There are two techniques available today:

- Extension unit operation that exists as a separate COM dll registered on the users computer. Extension operations have a view or GUI created using Petro-SIM's Extension View Editor and have access to most of the native capabilities of the simulator.
- User unit operations that exist as an active script (typically VB but can be Java Script, Perl and so on). Here the solution method is part of the unit operation definition within the case, though users can export user unit operation types for re-use in other cases. The user unit operation can be used to wrap external programs, with this mechanism providing a quick prototyping route as a prelude to a fuller integration.

CAPE-OPEN Unit Operation support

We added support for the CAPE-OPEN (CO) Unit Operation 1.0 interface to Petro-SIM in 2007, largely driven by business needs to improve the range of heat exchanger design technologies available with Petro-SIM. We implemented a container class for CO Unit Operations natively inside the simulator, supporting all the interfaces of the 1.0 standard. The implementation allows for persistence within Petro-SIM cases using the IPersistStorage method where supported, as well as native storage of the ports and parameters. CO unit operations behave like native Petro-SIM unit operations and benefit from the XML, relational database and automation support of any Petro-SIM object.

Users add CO unit operations by selecting the CAPE-OPEN Unit Operation from the palette, where it appears alongside all other available options. After adding the operation, users have to select from a listed of registered operations on their computer. They then connect streams and provide values for unit operation parameters using either the Petro-SIM native unit operation view or the vendor-provided GUI.

CAPE-OPEN unit operation parameters can be used in the case study, optimization and spreadsheet tools of Petro-SIM in the same way as native operations, with their variables exposed to the navigator tools users employ.

Technical Details

We implemented native support for hosting CO unit operations inside Petro-SIM and using them as part of a larger simulation model. The native approach was chosen since it is the simplest and most efficient mechanism. There is no external wrapper or intermediate layer.

Our implementation was done in C++. We import the CAPE-OPEN type library which creates C++ smart pointer classes. These simplify the code and avoid reference counting problems. We have a main CO unit operation class in Petro-SIM which uses its own ParamInfo

and PortInfo classes to keep track of information from the unit operation. We do not keep a local copy of parameter values, since we prefer to avoid any duplication and also CO parameter values may change for reasons unknown to Petro-SIM.

Other classes inside Petro-SIM which already exposed OLE interfaces (such as the Fluid and Stream classes) were enhanced to also implement and expose a few CO interfaces such as ICapeThermoMaterialObject and ICapeIdentification. These correspond to CO ports. Most of the work here is done by one method which recognizes the CO property name ("pressure", "temperature") and sets or retrieves the corresponding information. Some unit conversion between internal units and internal CO units is done, and the results are packaged as appropriate (into a SAFEARRAY for example).

Petro-SIM unit operations and thermo are not exposed via CAPE-OPEN interfaces. CAPE-OPEN unit operations generally interact with the host simulator via its implementation of the ports and hence work fine even if there are no explicitly exposed CAPE-OPEN thermodynamics interfaces.

After implementing CO support, we wanted to test as extensively as possible. The general rule in a situation like this is to code against the generic standard interface, but test against several specific implementations because they are always some differences or deviations. Testing unit operations from different vendors turned out to be well worth the effort. For example, we found:

- One vendor was assuming that an IDISPATCH pointer could simply be cast to a thermo material pointer without explicitly querying first.
- Unit operations implemented in Visual Basic can be very forgiving because VB transparently handles differences between arrays of variants and arrays of raw types etc. But unit operations implemented in C++ require strict adherence to the letter of the standard and sometimes do not clearly report mismatch. So it matters very much if you return a SAFEARRAY containing VARIANTS with doubles, or a SAFEARRAY containing raw doubles. This presented a challenge for us, mostly because if a CO unit operation fails, it is not always clear why it is failing.

Petro-SIM attempts to make CO unit operations appear and function like native unit operations:

- We provide a Petro-SIM like standard view which shows information that the CO exposes,
- We optionally mimic Petro-SIM behavior, such as not solving if feed streams or certain parameters are unknown,
- We show native Petro-SIM units (with unit conversion) for parameters, so that the CO unit operation can interact with other Petro-SIM unit operations such as adjusts, spreadsheets, and optimisers.
- Mixing and propagating stream assays.
- Solving when a change has been made.

The CO interface does not fully cover refinery/assay properties, and Petro-SIM has the option to mix and propagate assays through CO unit operations. Ports do expose all of the Petro-SIM OLE interfaces as well as the CO ones, so assay properties and such could potentially be manipulated by a CO unit operation in the current implementation.

Comparing the methods

The three mechanisms now available in Petro-SIM each have their advantages, where the choice of which to adopt depends on the purpose and starting point. The User unit operation offers rapid prototyping and is suited to simple integrations with limited numbers of parameters. We have deployed it in our own activities to provide a hook into methods written in Excel, where we operate it in conjunction with Petro-SIM's internal spreadsheet to move large numbers of values around. It is not suited to commercial delivery.

The Extension unit operation has been the traditional route for external vendors looking to provide technology modules into simulators. The proprietary nature of these methods means of course that vendors have to re-implement their extension for each new simulator.

The CAPE-OPEN unit operation offers an independent way forward, allowing component providers to make their extensions available to many platforms simultaneously. As our experience shows however, the CAPE-OPEN unit operation does not provide the same level of user experience that is available with an extension operation, since it is more limited in its GUI.

There is an overhead in building CAPE-OPEN unit operations around legacy systems. One must consider cost, purpose and intended audience before deciding which route to take. Shortly after adding CAPE-OPEN support to Petro-SIM, we were asked to wrap a third party DOS program as a unit operation in Petro-SIM. We considered using CAPE-OPEN as a way of testing and advertising the interface: we elected instead to use our built-in extension technology.

Experiences using CO Unit Operations

As we said earlier, the primary driver for us in adding CO Unit Operation support was to allow Petro-SIM users access to the heat exchanger design packages, particularly the HTRI Xchanger Suite modules. Our initial expectation was that our software customers would make use of the mechanism but we soon found much greater internal needs. KBC was not initially a member of the HTRI consortium and has since joined, with users in many company offices adopting the technology to support our process design and energy analysis activities.

Feedback is generally positive, with people benefiting from the increase in functionality now available. There have been the usual teething problems and missteps common to the introduction of any new technology but results have been good, with users able to model complex exchanger trains and use exchangers in both rating and design modes. The primary remaining issues centre around usability, explored below.

Improving usability

The CO standard allows for a unit operation to have two views: the vendor-provided one and the native simulator one. Feedback from our user community suggests that this is the weakest area of the standard. There are several issues:

- Native simulator view has no knowledge of the parameter structure and is limited to providing a flat view. This gets cumbersome where the operation exposes many parameters.
- Vendor view can provide a richer, more structured environment that is limited by being divorced from the native simulator. Things like unit of measure support and drag and drop mechanisms will behave differently from the native simulator.
- Vendor may not expose all parameters of the operation through its CO interface
- Port description is limited, particularly for heat exchangers where different thermodynamic properties can be used for each side of the exchanger.

We have in the past integrated some of KBC's reactor technology into other simulation platforms and have experienced first hand the challenges that arise where the unit operation has many hundreds of parameters available. The end-goal is surely to make the CO unit operation fully look and feel like a native unit operation, with the same richness of user interface the simulator provides for more deeply embedded operations. This minimizes user learning time and reduces mistakes.

Stripped to their essentials, all simulators are very similar in how they describe unit operations and in the types of view they provide. We all use the same view building blocks, with input boxes, dropdown boxes, group boxes, radio buttons and grids organized into pages of information. One solution to the issues we identify is to extend the standard to include support for a view definition described in XML that each platform vendor can render using their native tools. This approach would resolve the disjoint between the native view and the vendor-provided view, making the integration more complete.