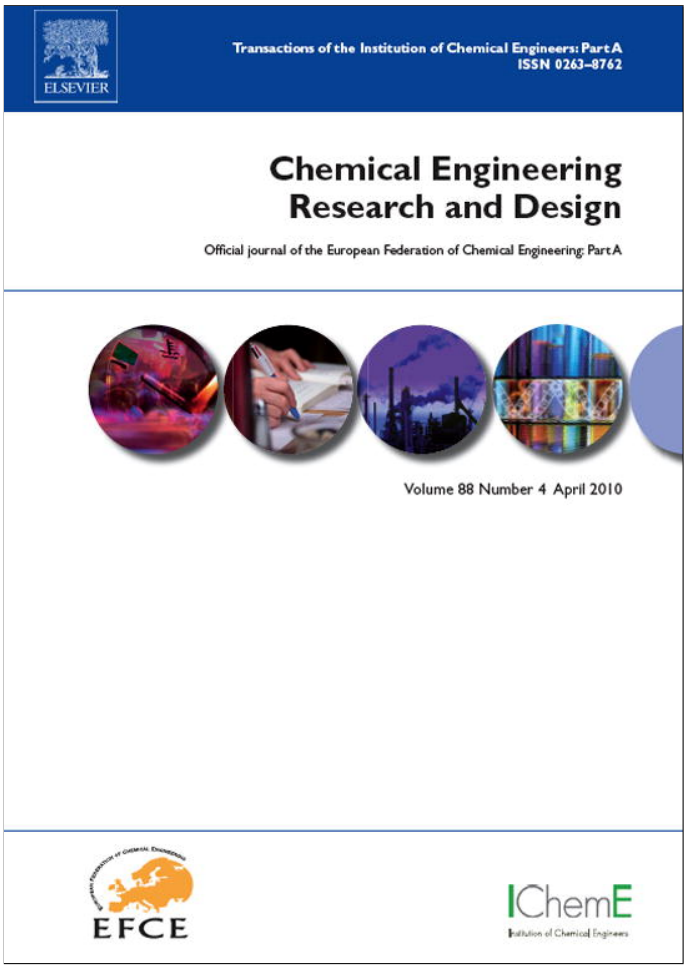

CHEMICAL ENGINEERING RESEARCH AND DESIGN 88 (2010) 421–429

Contents lists available at ScienceDirect

Chemical Engineering Research and Design

journal homepage: www.elsevier.com/locate/cherd

Systematic generation of a CAPE-OPEN compliant simulation module from GAMS and FORTRAN models

Alejandro O. Domancich[a,b,c,*], *Virginia Perez*[a], *Patricia M. Hoch*[b,c], *Nélida B. Brignole*[a,c]

[a] *Laboratorio de Investigación y Desarrollo en Computación Científica (LIDECC), UNS, Departamento de Ciencias e Ingeniería de la Computación, Av Alem 1253, Bahía Blanca 8000, Argentina*

[b] *UNS, Departamento de Ingeniería Química, Av Alem 1253, Bahía Blanca 8000, Argentina*

[c] *Planta Piloto de Ingeniería Química, CONICET Complejo CCT-UAT, Camino La Carrindanga km. 7 CC 717, Bahía Blanca 8000, Argentina*

A B S T R A C T

The adaptation of complex existing modules so that they meet the CAPE-OPEN (CO) standard is important in order to take advantage of previous work done for the simulation of processes. In this work we present the design and implementation of two different Operation-Unit (OU) modules that support the CO interface. Each resulting OU satisfies the definitions and guidelines established by the standard. The first module consists of a CO-compatible component for the steady-state simulation of a multi-bed ammonia reactor, whose original code had been written in FORTRAN. The second one, which also supports the CO interface, it is a GAMS-compatible module of a reactive distillation column. In order to develop these modules, an existing CO wizard that targets the mechanical generation of the OU Package was used. This wizard generates a Visual Basic project that contains the OU's source code and its installation package. In particular, the calculation routine and the Graphical-Unit Interface (GUI) were implemented. Before implementing the interfaces, an analysis of requirements for the correct interaction between the OU and the other simulator components was carried out. Both modules embedded in HYSYS take advantage of its complete property-package, which complements GAMS and FORTRAN in order to carry out the accurate computation of the chemical and equilibrium properties.

Keywords: CAPE-OPEN; Process simulation; HYSYS; GAMS; Process-modeling components

1. Introduction

In the area of process engineering, there are a lot of general mathematical models developed in commonly used software (e.g. GAMS, Brooke et al., 2004, FORTRAN, FORTRAN Company, etc.), that represent different items of chemical process equipment. In many cases, most of these individual models have turned out to be unnecessary for simulation purposes with the emergence of modular sequential simulators (e.g. HYSYS, Hyprotech, 2002 and Aspen, Aspentech), due to the complete equipment libraries that they include, thus allowing the users to simulate almost any process plant.

There is a disadvantage that arises when the plant includes a specific piece of equipment (e.g. a specialized chemical reactor, or a complex distillation column), whose behavior cannot be accurately represented by any of these general models provided by the existing commercial modular sequential simulators. A tailor-made model should be developed in order to solve these problems. Then, it is desirable to define a general mechanism for the conversion of specific equipment models into modules that can be installed or plugged in a sequential modular simulator. In this way, the user is able to take advantage of the benefits of such programs by using their complete library of equipment models and thermodynamic packages, but without losing the possibility of including in any plant simulation some self-modeled equipment. In short, it is undoubtedly convenient to be able to couple the tailor-made chemical process equipment with the other standard unit models, in order to simulate the whole plant under the same environment.

* *Corresponding author at*: Planta Piloto de Ingeniería Química, CONICET Complejo CCT-UAT, Camino La Carrindanga km. 7 CC 717, Bahía Blanca 8000, Argentina.

E-mail address: adomancich@plapiqui.edu.ar (A.O. Domancich).

Received 2 September 2008; Received in revised form 10 July 2009; Accepted 29 July 2009

0263-8762/$ – see front matter © 2009 The Institution of Chemical Engineers. Published by Elsevier B.V. All rights reserved.

doi:10.1016/j.cherd.2009.07.022

The CAPE-OPEN (CO) standard provides a suitable environment for this purpose because it enables the generation of modules with plug-and-play facilities, but it lacks a well-established methodology to aid the users to incorporate tailor-made modules. CAPE is becoming a commonly used integration tool applied in modern industry. From a commercial point of view, CO permits competitive product development. Moreover, it is attractive for software-developers that build components for process industries, who are prone to expand the number of applications where their products can be executed. In addition, software companies and academic institutions will be able to develop components and to transfer them to the process industry field afterwards, in a straightforward manner.

In view of the above-mentioned considerations, it is desirable to develop some migration strategies that facilitate the embedding of new process-modeling components within different simulation environments. In this work we present two different migration strategies. In the first place, we developed a strategy to embed a FORTRAN model in HYSYS. For this methodology, we present the design and development of a CO-compatible component for the steady-state simulation of a multi-bed ammonia reactor. In order to develop this module, the objects to be invoked were defined and the reactor model was coded. The original code, which had been written in FORTRAN, was encapsulated in a Dynamic Link Library (DLL) and employed as an OU calculating routine. Both the CO-compliant OU and the interfaces were implemented in Visual Basic (VB), and then tested in the environment of a commercial simulator.

In the second place, we describe a migration strategy to design and implement a user-friendly GAMS-compatible module. More specifically, a CO-compatible simulation component for a reactive distillation column was developed. This module was developed modifying the GAMS model in order to read input values from a GDX file. An Excel file, which contains the user-defined parameters and the incoming stream properties, was created in the Graphical User Interface (GUI) written in VB programming language. Afterwards, the GDX file was created from the Excel file using the GDXXRW utility of GAMS. Briefly speaking, we have attained the technical know-how about the method to incorporate a user-defined item of equipment that was both previously modeled and simulated via GAMS.

The commercial simulator HYSYS was used in order to test the developed components and both modules became available extensions on the list of operation units. Following both strategies presented in this paper, the user will be able to incorporate new process-modeling components within existing process-modeling environments. It will also be possible to make the embedded models accessible to other applications. Both modules embedded in HYSYS profit from its complete property-package, which complements GAMS and FORTRAN in order to carry out the accurate computation of the chemical and equilibrium properties.

This paper begins with a description of the interoperability CO standard, its main goals and its working environment. From this point onwards, the possible migration strategies and software tools available for this process follow. Then, the main features of two cases of study (a reactor and a distillation column) are described. In each case, the VB code is reported and the results obtained with the implemented modules are then discussed. The final part of the document deals with the main conclusions achieved.

2. CAPE-OPEN project

The CAPE-OPEN (CO) standards are the uniform rules to interface process-modeling software components specifically developed for the design and operation of chemical processes. They are based on universally recognized software technologies such as Component Object Model (COM) and Common Object Request Broker Architecture (CORBA). The CO standards are open, multiplatform, uniform and available free of charge. They are described in a formal documentation set (CAPE-OPEN Standards and Supporting Documents), which is put forward by the CAPE-OPEN Laboratories Network (Co-LaN).

The primary purpose of CO-LaN is to promote the use and development of the CO standards in Computer-Aided Process Engineering (CAPE) software, and more generally to encourage all actions aimed at facilitating the use of CAPE software tools in industry, government agencies and academia. Some of CO-LaN's main activities are the following:

- the definition of user-friendly CO standards by working with software vendors in order to clarify user priorities for the development of process-modeling software component;
- the international promotion of the CO standards in order to create commercially valuable modules;
- the supply of compliance testers in order to support component development.

2.1. CAPE-OPEN vendors

In order to obtain better results when trying to solve a particular problem, it should be possible to gain access to both programs from several simulator providers and in-house software that contains specific methods or data. IK CAPE, CAPE-OPEN and GLOBAL CAPE-OPEN (Mayer and Shoenmakers, 1998) are the major projects working on this area. Their main features are the following:

- IK CAPE is a cooperation agreement of chemical companies. The main activities are the formulation of common standards for either programming or interfaces.
- CAPE-OPEN (CO) is a project funded by European Community. Its partners are chemical and petrochemicals companies (BASF, Bayer, BP, DuPont, Shell, Total, ICI, IFP, etc.), universities (Imperial College, Institut National du Petrole, RWTH Technische Hochschule Aachen, etc.) and vendors (Aspentech, SinSci, Quantisci, etc.). The project promotes an open environment for simulation with common interfaces that should be independent from the vendors. The main objective is to enable native components of a simulator to be replaced with minimal effort by those from another independent source, or by those coming from another simulator (CAPE-OPEN Synthesis Report, 1999). Then, the CO project aims at describing and publishing agreed standards for the software-component development of a process simulator.
- GLOBAL CAPE-OPEN is the extension of the CO concept to a worldwide project that aims at the acceptance of those standards for process–simulator interfaces and software.

2.2. Scope

The CO-Interface System's architecture is based on an object-oriented technology. It allows software systems to be

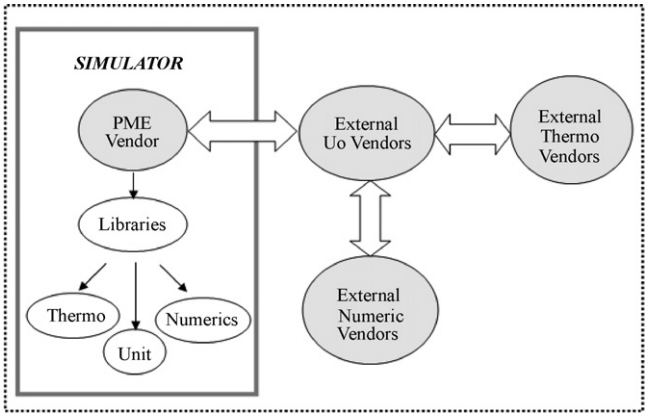

Fig. 1 – Integration of various CO tools.

constructed from various components that can relate to each other via previously defined interfaces. These components may be supplied by different vendors, and they may reside on the same or different machines connected through a network.

The basic structure in an architecture for a modular process-modeling tools contains a process modeling executive (PME), which is responsible for both the model construction and the computations necessary to solve it. Fig. 1 shows the interaction that takes place between the PME and any other module. In this scheme, the PME is supplied by one vendor, whereas all the process-modeling components (PMC) come from different suppliers. It is contemplated that the CO system requires the following kinds of objects:

- *a Unit Object* that represents the CO unit and provides methods for initialization, calculation and result reporting;
- *a Simulator Object* that provides services that an Operation Unit needs;
- *a Thermo Object* that provides physical-property services;
- *a Numerical Object* that has numerical solvers.

The CO-Interface System is a standard means of connecting an external software component, e.g. an Operation Unit (OU), to any compliant simulator. The internal coding is like a black box for both the simulator and the OU. The system's main function is to translate requests for information or action. These interfaces could be implemented in various different ways. CAPE-OPEN has chosen to adopt an object-oriented approach that views each PMC as a separate object. Communication between objects is handled by a middleware, such as the Object Management Group's (OMG) CORBA (CORBA, 2005) and Microsoft's COM (Microsoft COM, 2005). Thanks to the availability of these technologies, each software object is able to interact with others that are based on a formal interface definition, which is expressed in standard languages. The communicating objects may be run as either a part of the same process or in different processes. Besides, the software may reside on either the same device or different computers connected through a network, thus providing local/remote transparency.

2.3. *Operation-Unit interfaces*

A modular-oriented unit operation has the following facilities (CAPE-OPEN, 2001):

- *Ports*: They allow the OU module to be connected to other modules and to exchange material or energy data, or any other kind of information with them. Each material port has a thermo material object associated to it. By using appropriate methods of this object, the OU is able to get the thermo and physical properties of the inlet and outlet flows. These properties are a part of the property-package system.
- *Parameters*: They represent information that is not associated to the ports. Nevertheless, they are necessary because the modules wish to expose it to their clients.
- *User interface*: It allows the user to configure each instance of the module in an appropriate fashion. This interface should be provided by the developer and it is out of the scope of the standard.
- *Reports*: They present the results of the module's computations. This feature does not form part of the standard.

The computation of an OU module is triggered explicitly by its clients via the invocation of a method provided by the unit operation object, which is called Calculate. There are no agreed standards for this method, but a validation test should be performed at this point. For example, the validity status of the material objects connected to each port should be checked.

Equation-oriented OU objects also have the same configuration. The key difference lies in the PME task. Instead of carrying out any computation, the PME's main responsibility is to form and expose a set of mathematical equations and to solve it by interacting with one or more numerical solvers.

3. Development of a CAPE-OPEN compliant simulation module from a FORTRAN model

In this part of the paper, the design and implementation of a CAPE-OPEN compliant simulation module, which was previously developed in FORTRAN, are described. The module corresponds to a multi-bed ammonia reactor and, once it was developed, it was tested in HYSYS.

3.1. *Migration strategy*

Since CO standards are a recent development, a lot of academic, commercial, or in-house CAPE software naturally do not still comply with the CO standards (Köller and Töbermann, 2002). The main problem that may arise is that source code could have been written in a procedural programming language, such as FORTRAN, where there is only a collection of subroutines. This viewpoint of a complete bulky system is different from the one of the software component to be created. This latter approach is focused on object-oriented techniques. Therefore, a migration process is needed. There are mainly two ways to do this: reimplementing from scratch or wrapping existing code (Köller, 1999).

- *Reimplement-from-scratch technique*: This strategy seems to be the simplest one. But special care must be taken when creating the new system. Although the original system may have been used and tested for a long time, the software created from scratch may have a lot of errors resulting in system crashes or, what is even worse, in inconsistencies between the behavior of the old and the new implementation. Finding and fixing these bugs will surely take a lot of resources. Therefore, this strategy seems to be very dangerous and expensive.

- *Wrap-existing-code technique*: Applying this technique the system can be migrated without changing the source code. In this case, the FORTRAN code is treated as a black box whose functionality is used, but it is ignored how this functionality is implemented. An object-oriented shell around this black box is provided to integrate the system into the component-based framework. This shell is implemented using an object-oriented language, such as Java or C++. Based on this shell, a CORBA layer is provided to make the functionality available to the outside. This layer is also implemented in Java or C++.

Leaving source code untouched is a way to avoid introducing new bugs to the system's core functionality. Another advantage of this strategy is that code optimization is no longer necessary because it was already done by the original developers.

Taking all its benefits into account, the Wrap-existing-code strategy was chosen for this project. Some specialized tools become necessary for the implementation of this technique.

In the GLOBAL CAPE-OPEN project a number of tools have been developed to support the implementation of CAPE-OPEN components (Köller and Töbermann, 2002). All these tools are wizards that can produce source code skeletons for CAPE-OPEN components, thereby relieving the developer from the implementation of everything that is conceptually repeated in almost every CAPE-OPEN component. One of these wizards that targets the mechanical generation of Operation-Unit Package, was used in the present project. This wizard generates a Visual Basic project containing the source code of the OU and the installation package for other machines. In particular, the calculation routine and the Graphical-Unit Interface are not generated. It is the developer's responsibility to provide this code. The OU generated will be compatible with the 0-9-3 version of the CO standard.

As we have seen so far, migration requires the integration of code pieces that have been implemented in different programming languages. Each of these languages has its own data structures and data types and each one has different calling conventions for procedures, subroutines or functions (Köller and Töbermann, 2002). To overcome these differences a bridging technology is needed. A piece of software that may be an executable, a library, a piece of source code, or macros, is used to call legacy routines that reside in a compiled library in the new source code. The bridging approach clearly separates old and new code, thereby facilitating the maintenance of both subsystems.

One of these technologies is the Dynamic Link Libraries (DLL), which is employed in this project. Creating a DLL allowed us to call a FORTRAN subroutine from Visual Basic (Fig. 2) without using additional code.

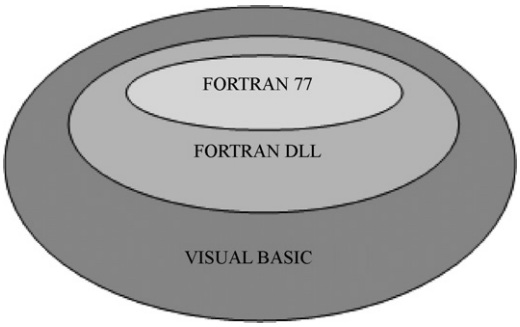

Fig. 2 – Wrapping FORTRAN code.

3.2. *Study case: A simple chemical reactor*

3.2.1. *Description*

The reactor used as case of study is a plug-flow unit and is assumed to be isothermal with fixed conversion (Byke and Grossman, 1985). The kinetics of the ammonia synthesis reaction over a double-promoted iron catalyst is described by the following rate equation:

$$N_2 + 3H_2 \Leftrightarrow 2NH_3$$

The reactor must be cooled to prevent the temperature of the catalyst from rising above 800 K, its deactivation temperature. An inlet temperature of 703 K and a heat transfer coefficient of 19,644 kJ/h K are sufficient to maintain the catalyst temperature below 800 K.

Conversion varies directly with catalyst volume inside the unit. For a given catalyst volume there is a single heat transfer coefficient (UA) whose value corresponds to a maximum conversion. UA dictates the rate that the heat can be withdrawn at from the reactor. For low values of UA, the heat generated by the reaction builds up rapidly in the reactor, a hot spot occurs near the feed point and most of the catalyst volume becomes inactive. As a result, the conversion is low. For high values of UA, the rate of heat removal is large enough to slow the reaction rate, thus decreasing the conversion.

```
Private Sub ICapeUnit_Calculate()
'Declare private variables
...
'Gets connected ports
Set PORTIN = GetPort("PORTIN")
Set PORTOUT = GetPort("PORTOUT")
'Gets material objects associated to the ports
Set materialIn = PORTIN.connectedObject
Set materialOut = PORTOUT.connectedObject
'Gets components ids and number of components
Components = materialIn.ComponentIds
NumComponents = materialIn.GetNumComponents
'Gets the total flow and partial flows of the input
flowIN = materialIn.GetProp("flow","Overall",Components,vbNullString, "mole")
totalIN = materialIn.GetProp("totalFlow","Overall",Empty,vbNullString, "mole")
Gets the temperature and pressure of the input
T = materialIn.GetProp("temperature", "Overall", Empty, vbNullString, Empty)
P = materialIn.GetProp("pressure", "Overall", Empty, vbNullString, Empty)
'Gets the values of the parameters
...
'Calculates input parameters for the DLL routine
...
'Calls the DLL routine
Call DLL_QBED(XIN, XOUT)
'Sets outport temperature and pressure from the output parameter XOUT
...
materialOut.SetProp "temperature", "Overall", Empty, vbNullString, Empty, TOUT
materialOut.SetProp "pressure", "Overall", Empty, vbNullString, Empty, POUT
'Sets total and partial flow of outport from the output parameter XOUT
...
materialOut.SetProp "flow","Overall",Componentes,vbNullString,"mole", flowOUT
materialOut.SetProp "totalFlow","Overall",Empty,vbNullString,"mole", totalOUT
End Sub
```

Fig. 3 – Visual Basic code implementing a reactor Unit Operation.

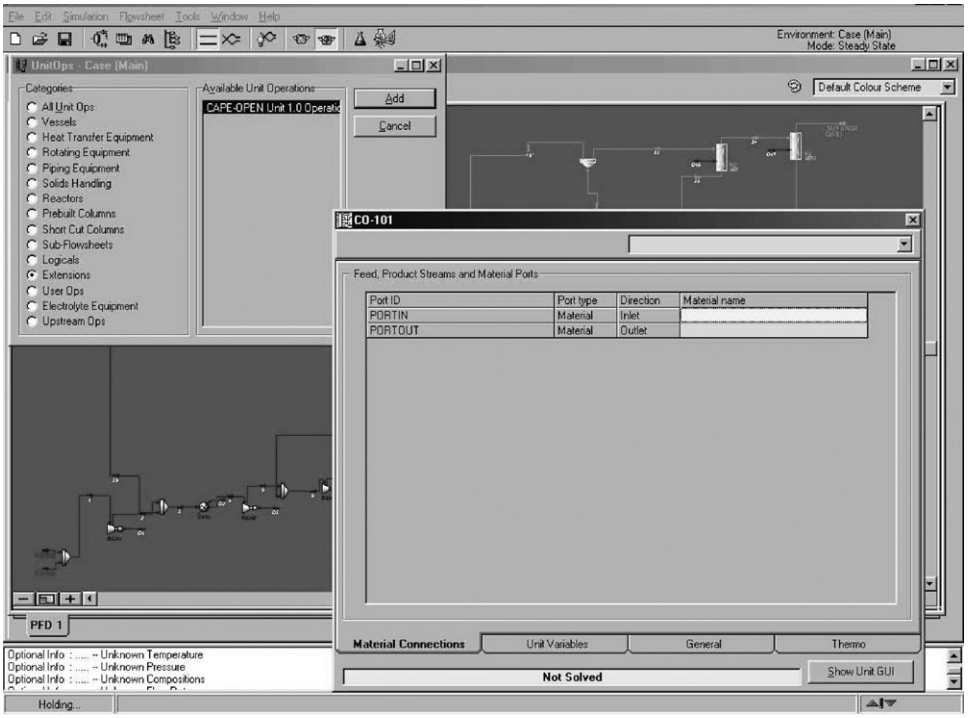

Fig. 4 – CO Operation Unit available at HYSYS.

3.2.2. *Source code considerations*

The original code for the simulation of the reactor is written in FORTRAN 77 for the VAX 11/780 (Byke and Grossman, 1985). It invokes an IMSL library routine, called Dgear, in order to integrate the model's differential equations. The new version of the IMSL library includes this routine under a different name and with some additional features. This new routine is called Ivpag and it was the one employed in this project.

Another modification added to the original code involved the interaction with the user. The source code already had a user interface that was removed because it overlapped with the OU functionalities. It became the OU responsibility to obtain and check the input parameters and to display the results in an appropriate way.

3.3. *Results*

Once the FORTRAN code had been tested and wrapped into a DLL, the OU was created. This OU presents the following features:

- There are two available material ports, INPORT and OUPORT. The material object connected to the INPORT carries the initial flow rates of hydrogen, ammonia, nitrogen, water, methane and argon. It also provides the feed temperature and pressure. The OUTPORT flow rates, temperature and pressure are specified by the unit.
- A set of parameters specify the remaining data that must be provided to the calculating routine.

3.3.1. *Visual Basic code of the reactor module*

A pseudo-code's core of the calculation routine for the reactor's OU that relates FORTRAN with HYSYS is shown in Fig. 3.

Unit conversions were added in this OU calculation routine. This was necessary because CO specifies certain units strictly for temperature, pressure, flows and so on. Since these units differed from the ones used in the FORTRAN code, the appropriate transformation was included. In addition, the names of the components had to be properly written in order to agree with those from the standard. All this information is available at the Thermodynamics and Physical Properties document (CAPE-OPEN, 2002).

3.3.2. *Reactor module tested in HYSYS*

The CO-compliant OU developed was successfully installed and it became available in HYSYS as an extension to the OU list (Fig. 4).

In order to test this new OU, it was installed in an ammonia plant simulated with HYSYS. This plant used a reactor provided by the mentioned simulator that implements a model different from the one presented in the study case (Byke and Grossman, 1985). This is the reason why differences in the values between the simulation runs and the case study results arose. Then, the HYSYS' reactor was replaced by the CO component developed and simulation runs were analyzed. The obtained results were similar to those from the case study, thus demonstrating that the whole simulation works successfully using the CO unit. Table 1 shows a comparison between the results obtained before incorporating the CO UO and after doing so (Fig. 5).

Table 1 – Comparison between the results obtained for the ammonia plant before incorporating the CO UO and after doing so.

Variable	Without CO UO	With CO UO
Reactor conversion (%)	24.41	27.91
Reactor temperature (K)	702.9	697.6
Reactor heat exchanged (Kcal/h)	4.181e7	4.208e7
Total plant production (kg NH_3/h)	54,630	55,170
Product purity (specified variable)	0.9995	0.9995
Plant purge gas (kg N_2/h)	4944	6385
Plant extracted water (kg/h)	909.6	846.0

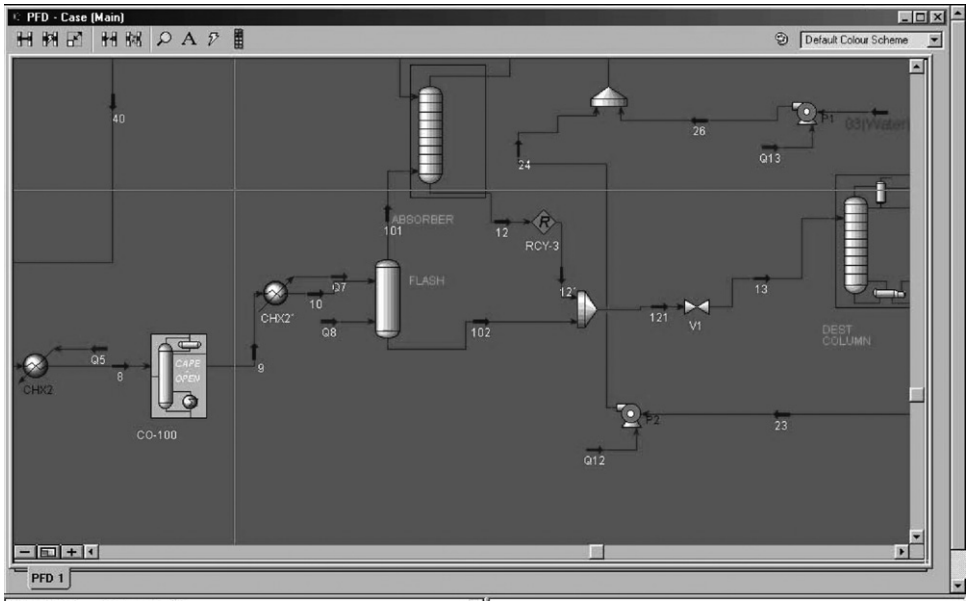

Fig. 5 – CO Unit Operation in a HYSYS simulation case.

4. Development of a CAPE-OPEN compliant simulation module from a GAMS model

In this section, the design and implementation of a user-friendly GAMS-compatible module that supports the CAPE-OPEN (CO) interface were carried out. A CO-compatible simulation component for a reactive distillation column was developed. HYSYS was employed in order to test the developed component, which became an available extension on the list of operation units.

4.1. *Migration strategy*

4.1.1. *Software embedment*

In order to obtain a CO-compliant module of the reactive distillation column we used the CO Unit Operation Wizard described in Section 3.1.

In the first place, the distillation column model was developed and tested in the General Algebraic Modeling System (GAMS), which is a high-level modeling system for mathematical programming and optimization. As the wizard-generated code was written in Visual Basic, a code migration was required to integrate the pieces of code that had been implemented in different programming languages.

The original GAMS code was modified, in order to obtain simulation data from an external source. For this purpose the GDX (GAMS Data Exchange) was used (GAMS, GDX). This facility provides a utility called GDXXRW that allows reading and writing data of an Excel spreadsheet. The OU code generated creates an Excel spreadsheet with all the necessary input data and then calls GDXXRW before running the GAMS model. Once the model is solved, the output data is written into another Excel spreadsheet so it can be read from OU Visual Basic code (Fig. 6).

4.1.2. *Steps on how to embed existing software*

In order to develop a CO Operation Unit from an existing GAMS model, some small modifications should be done in the original GAMS code. For instance, all the user-defined data must be provided from an external file. Our approach for data input consists in creating an Excel file (i.e., datain.xls) with all the user-defined data, and then creating a GDX file (i.e., datain.gdx) using the GDXXRW tool.

GAMS code should include the following:

$GDXIN datain.gdx (load the GDX file)

SET VAR1 (*) (define a variable)

$LOAD VAR1 (load the value from the external file)

Once the GAMS code has been tested in GAMS IDE, we developed a CO Operation Unit using the Wizard tool provided by CO project.

The Visual Basic code generated by the wizard was modified in order to create the Excel file with input data and to call GAMS IDE. Visual Basic code should include the following:

x = CreateProcessA(0&, "gams.exe model.gms", 0&, 0&, 1&,

NORMAL_PRIORITY_CLASS, 0&, 0&, NameStart, NameOfProc)

(call to model.gms)

x = WaitForSingleObject(NameOfProc.hProcess, INFINITE)

x = CloseHandle(NameOfProc.hProcess)

In summary, the steps to embed an existing GAMS model in HYSYS are the following:

(A) In GAMS, develop and test the unit operation model.
(B) Use the CO Unit Operation Wizard in order to generate a Visual Basic project containing:

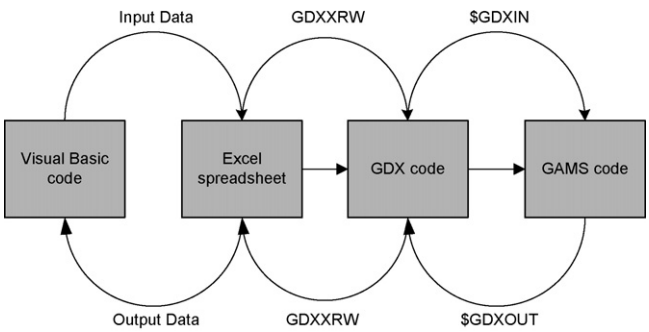

Fig. 6 – Bridging approach.

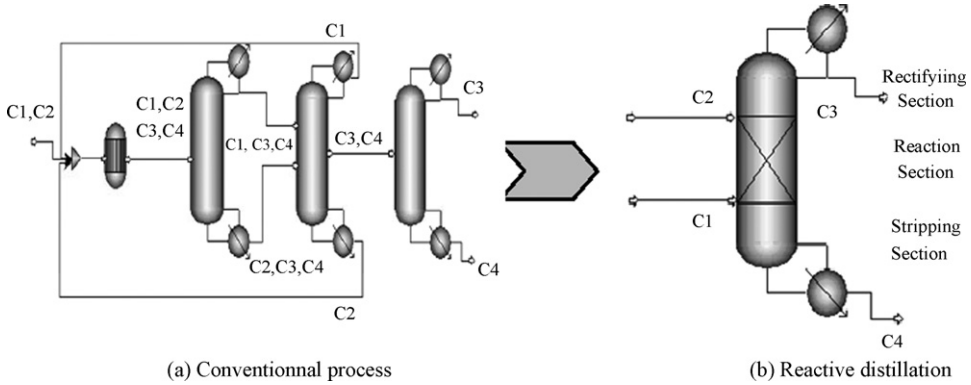

Fig. 7 – (a) Conventional process and (b) reactive distillation.

- the source code of the Operation Unit (OU);
- the installation package. This enables the unit to be installed in other machines.

(C) Modify the original GAMS code in order to obtain simulation data from an external source. For this purpose the GDX (GAMS Data Exchange) is used.

(D) The GDX facility provides a utility called GDXXRW that allows reading and writing data of an Excel spreadsheet.

4.2. *Case study: A reactive distillation column*

4.2.1. *Description*

To take advantage of the existing software, we concentrated on a complex model for a reactive column that had been built and tested in advance (Domancich et al., 2007). Reactive distillation is a hybrid operation that combines two key tasks in chemical engineering, i.e., reaction and separation processes (Almeida-Rivera et al., 2004). Considering the next reversible reaction scheme:

$$C_1 + C_2 \leftrightarrow C_3 + C_4$$

where the boiling points of the components follow the sequence C_1, C_3, C_4 and C_2, the traditional flow-sheet for this process consists of a reactor and a sequence of distillation columns (Fig. 7a). Components C_1 and C_2 are fed to the reactor, where the equilibrium reaction takes place in the presence of a catalyst. A distillation train is required to produce C_3 and C_4, which are obtained as pure products. An alternative for this process is the use of a reactive distillation column.

The reactive distillation column, see Fig. 7b, integrates reaction and separation and offers the possibility to overcome constraints given by chemical and phase equilibrium. It consists of a reactive section in the middle with non-reactive rectifying and stripping sections at the top and bottom. The task of the rectifying section is to recover reactant C_2 from the product stream C_3. In the stripping section, the reactant C_1 is stripped from the product stream C_4. In the reactive section the products are separated, driving the equilibrium to the right and preventing any undesired side reactions between the reactants with any of the products. For a properly designed column, virtually 100% conversion can be achieved.

4.3. *Results*

Once the Visual Basic code was tested, the OU was installed in HYSYS and it became available as an External Unit. Fig. 8 shows how the data are transferred between HYSYS and GAMS.

In summary, the HYSYS OU, which was naturally programmed in Visual Basic, calls GAMS, where the OU mathematical model should be previously introduced. It is

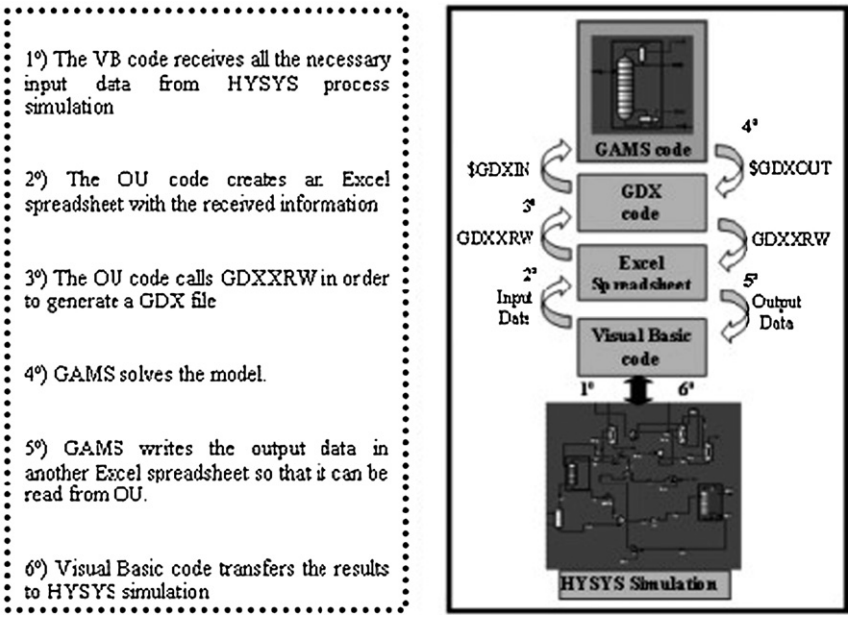

Fig. 8 – Data-transfer cycle between HYSYS and GAMS.

```
//GAMS process initialization
NameStart.lpTitle = "VB GAMS"
NameStart.dwFlags = STARTF_USESHOWWINDOW
NameStart.wShowWindow = SW_SHOWDEFAULT
NameStart.cb = Len(NameStart)
// datain.gdx file creation
x = CreateProcessA(0&, "C:\Program Files\GAMS21.5\GDXXRW datain.xls set=i
rng=etapas!B1:R1 cdim=1 par=etapas rng=etapas!A1:R5 cdim=1 rdim=1 set=val
rng=valores!A1:R1 cdim=1 par=valores rng=valores!A1:R2 cdim=1 par=portin
rng=portin!A1:J5 rdim=1 cdim=1 set=j rng=portin!A2:A5 rdim=1 par=aw
rng=a!A1:E5 cdim=1 rdim=1 par=bw rng=b!A1:E5 cdim=1 rdim=1 par=antoine
rng=antoine!A1:D5 cdim=1 rdim=1 par=cp rng=cp!A1:E5 cdim=1 rdim=1", 0&, 0&,
1&, NORMAL_PRIORITY_CLASS, 0&, 0&, NameStart, NameOfProc)
If x <> 1 Then
MsgBox "CreateProcess GDXXRW in failed. Error: " &
Err.LastDllError
End If
x = WaitForSingleObject(NameOfProc.hProcess, INFINITE)
x = CloseHandle(NameOfProc.hProcess)
//GAMS call
x = CreateProcessA(0&, "C:\Program Files\GAMS21.5\gams.exe modelo.gms", 0&,
0&, 1&, NORMAL_PRIORITY_CLASS, 0&, 0&, NameStart, NameOfProc)
If x <> 1 Then
MsgBox "CreateProcess GAMS failed. Error: " & Err.LastDllError
End If
x = WaitForSingleObject(NameOfProc.hProcess, INFINITE)
x = CloseHandle(NameOfProc.hProcess)
// dataout.xls file creation
x = CreateProcessA(0&, "C:\Program Files\GAMS21.5\GDXXRW dataout.gdx var=x
rng=x! var=y rng=y! var=I rng=I! var=v rng=v! var=FD rng=FD! var=T rng=T!
var=rx rng=rx! var=Qcond rng=Qcond! var=Qreb rng=Qreb! var=conversion
rng=conversion! ", 0&, 0&, 1&, NORMAL_PRIORITY_CLASS, 0&, 0&, NameStart,
NameOfProc)
If x <> 1 Then
MsgBox "CreateProcess GDXXRW out failed. Error: " &
Err.LastDllError
End If
x = WaitForSingleObject(NameOfProc.hProcess, INFINITE)
x = CloseHandle(NameOfProc.hProcess)
```

Fig. 9 – Visual Basic code implementing a reactive distillation Unit Operation.

important to note that GAMS solves that model, and sends the results back to HYSYS. These results may be used by HYSYS as the entry to any other equipment entries in order to continue with the complete simulation of the process plant.

4.3.1. *OU Visual Basic code*

Fig. 9 shows a relevant part of the pseudo-code of the calculation routine that relates GAMS with HYSYS for the column's OU.

5. Conclusions

The present document aimed at identifying the primary concepts and tools that should be used when building a CO-compliant module. The methods for software embedding described in this paper are useful for the industry, where HYSYS has become widespread. For simulation purposes, HYSYS provides a lot of general models. In contrast, there are some dedicated, accurate, specific models in any plant, which the process engineers need to include in their simulation. Thanks to our methodology, the engineers can take advantage of their accurate models, which surely demanded hard efforts for their development. We have developed methods to incorporate two different user-defined process units. The first one was both previously modeled and simulated via FORTRAN, whereas the second one was previously implemented via GAMS. In this way, it is possible to make any FORTRAN or GAMS model compatible with HYSYS. Moreover, there is access to the complete HYSYS property-package, which is lacking in these programming languages. This facility enables to carry out the accurate computation of the chemical and equilibrium properties. It is advantageous because the cumbersome procedure of adding thermodynamic equations, correlations and property data in the original mathematical models can be avoided.

From an academic point of view, CAPE-OPEN allows new theoretical developments to be implemented and tested together with qualified software components. New modules

are able to use external software components, such as data bases with thermodynamic properties of the simulator. These components may come from a different source, thus improving the final product quality and reducing implementation time and cost.

From a commercial point of view, CO permits competitive product development. Companies that develop software components for process industries will expand the number of applications where their products can be executed. In addition to this, software companies and academic institutions will be able to develop components and then transfer them to the process industry field in a straightforward manner.

The development of CO-compliant software involves interaction between two main disciplines: Computer Science and Process Engineering. This knowledge complementation is necessary because Software expertise (DLL, Object-Oriented Programming, etc.) is required and details of the process to be implemented should be deeply and properly understood. Then, it should be noted that the migration process is a complex task, while the generalized automation is practically unfeasible because there are a lot of diverse individual cases.

Acknowledgments

We gratefully acknowledge the economic support given by the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) through grant PIP 5930. We would like to thank the Agencia Nacional de Promoción Científica y Tecnológica, SeCyT, Argentina, for their Grants: PICTO UNS 917, and Grant No. 11-12778, which are part of the "Programa de Modernización Tecnológica", Loan Contract BID 1728/OC-AR.

References

Almeida-Rivera, C., Swinkels, P. and Grievink, J., (2004). Designing reactive distillation processes: present and future. Comput. Chem. Eng., 28: 10.

Aspentech, http://www.aspentech.com.

Brooke, A., Kendrick, D., Meeraus, A. and Raman, R., (2004). *GAMS: A User's Guide*.

Byke, S. and Grossman, I., (1985). *Design of an Ammonia Synthesis Plant Department of Chemical Engineering*. (Carnegie-Mellon University, Pittsburgh, PA).

CAPE-OPEN Synthesis Report, (1999). Available at http://www.colan.org.

CAPE-OPEN Open Interface Specifications—Unit Operations, (2001). Available at http://www.colan.org.

CAPE-OPEN Open Interface Specifications—Thermodynamics and Physical Properties, (2002). Available at http://www.colan.org.

CAPE-OPEN Standards and Supporting Documents. Available at http://www.colan.org.

CORBA Object Management Group, (2005). http://www.omg.org/corba.

Domancich, A., Brignole, N. and Hoch, P., (2007). Optimal structure of reactive and non-reactive stages in reactive distillation processes, in Chemical Engineering Transactions, Jiri Klemes, (ed). (AIDIC Servizi S:R.L)

FORTRAN Company. http://www.fortran.com.

GAMS GDX facilities and tools. Available at http://www.gams.com.

Hyprotech., (2002). *HYSYS User Guide*.

Köller, J., (1999). *Migration Methodology Handbook*. Available at http://www.colan.org

Köller, J. and Töbermann, J., (2002). *Migration Cookbook*. Available at http://www.colan.org

Mayer, H. and Shoenmakers, H., (1998). Application of CAPE in industry—status and outlook. Comp. Chem. Eng., 22: 1061.

Microsoft COM, (2005). Available at http://www.microsoft.com/com.