European Symposium on Computer Aided Process Engineering – 15
L. Puigjaner and A. Espuña (Editors)

A CAPE-OPEN based framework for process simulation solutions integration

Laurent Testard[a] and Jean-Pierre Belaud[b]

[a] RSI

Parc Technologique de Pré Milliet, F-38330 Montbonnot, France

Laurent.Testard@rsi-france.com

[b] Laboratoire de Génie Chimique, CNRS UMR 5503, INPT-ENSIACET

118 Route de Narbonne, F-31077 Toulouse Cedex 4, France

JeanPierre.Belaud@ensiacet.fr

Abstract

This paper deals with the challenge of knowledge integration within process modelling and simulation applications. We expose the particular problem of unit operations integration in a flowsheet taking benefit from the standard CAPE-OPEN technology. Such approach can ease to get an application specific to each domain, project and context adding inner software components for modelling thermodynamic, kinetic, unit operations... In addition to the coding task problem, engineers are faced with the difficulties of such integration, from a software and process engineering point of view. A framework from RSI for components integration supports the development task by providing abstraction of several domain related concepts and produces CAPE-OPEN compliant unit operation. As a way of validation of this approach, it was successfully used in process engineering and computing education.

Keywords: Dynamic simulation, Unit Operation, CAPE-OPEN, Framework

1. Introduction and context

Due to recent developments in software applications in the past years, the problematic has evolved from the development of monolithic multi-physics software applications to the integration of software components from different domains. The process engineering field does not escape to this problematic of information exchange. CAPE-OPEN (CO) standard (from CO-LaN organisation discussed in Pons et al., 2003) is a significant technology for interoperability and integration of process engineering software components allowing a *components off the shelves* engineering. Such approach suggests to get applications from *best-in-classes* and *inner* software components. Belaud and Pons (2002) introduce the standard and Braunschweig et al. (2000) deal with the needs, motivations and challenges of such standardization process. CO consists in a *technical architecture*, *interface specifications* and *implementation specifications*. The *technical architecture* relies on modern development tools and up-to-date information technologies (object-oriented paradigm, component-based approach, web-

enabled architecture, middleware technology) and uses the UML language. The *interface specifications* identify a conceptual model and the *implementation specifications* give the corresponding Microsoft COM and OMG CORBA platform specific model. The specifications cover major application areas i.e. unit operation, thermodynamic and physical properties, numerical solvers, chemical reactions systems etc. As example Roux et al. (2003) present the unit operation interfaces and show compliant applications for a fixed bed reactor for butane isomerization.

This move from monolithic to componentized software did not suppress the complexity inherent to the application design. A unit operation (UO) that simulates a device has generally to take into account several kinds of equations. A reactor UO is a good illustration of this complexity, because it mixes kinetics, thermodynamics and other equation types to be used in a flowsheet. Also recent developments in the field of programming languages and application frameworks tends to be more complex and powerful, so that it has become much harder for a software engineer to understand perfectly the techniques involved in software development. In the particular field of process simulation, the CAPE-OPEN standard defines some natural concepts for the developer of unit operations and thermodynamic properties. The drawback is a higher complexity in the development, installation, and management of software involving components middleware technology i.e. CORBA and COM. Also, even if the CAPE-OPEN standard defines the properties that a *thermodynamic property package* should propose, a given *property package* does not have to make all these properties available for computations. The many implementations that can exist of such a component can make the development of UO able to use these components really difficult. This is particularly true if a CO compliant UO has to be inserted into several different process simulation environments. The solution is the genericity. The code that can handle such various *property packages* can be quite complex and should be reused between different projects.

Thus we have raised three major difficulties for inner CO compliant UO components integration; (i) the modelling task inherent to process engineering domain, (ii) the software technologies inherent to component based approach and (iii) the CO standard design and training inherent to any standardisation process lifecycle. The framework discussed in the next section suggests a way for the complexity management of item (ii) and (iii).

These difficulties naturally apply to the problematic of *process engineering and computing* teaching. The students do not know every aspect of the domain, do not apprehend every aspect of software development and can understand CO to some degree. So we experiment the use of framework during a “lab” session presented in section 3 for an education on *process software solutions integration*.

2. A framework for unit operation integration

2.1 Definition and objectives

RSI develops simulators in the field of process simulation, from the upstream domain to chemical process. RSI provides *INDISS*™ (INDustrial and Integrated Simulation Software, see RSI, 2004, for details and contact), a simulation environment for the

whole process life cycle. The CAPE-OPEN standard 1.0 does not cover the dynamic simulation aspects. We use a draft version that proposes interfaces for dynamic unit operations (CO-LaN, 2004, provides further details), INDISS being used for test.
In addition a CAPE-OPEN based framework is developing to support the development of CO compliant unit operation component. A framework is a set of classes that embodies an abstract design for solutions to a family of related problems. It provides a set of class libraries encompassing functions arranged in an inheritance hierarchy. Here *RSI framework* is a *set of C++ primitives and classes* that aims at the development of CO compliant (static or dynamic) unit operations components for COM platform. It enables the reuse of generic code, as well as providing a complete control over the run-time behaviour of UO. In this way, the framework approach is opposed to a "wizard" approach, in the sense that the developer can choose to use or not a feature of the framework, instead of having to configure (or even modify) the wizard to serve his requirements. He can provide equivalent functionality e.g. an internal optimization or an external one that better targets his needs.
The main goal is to support the process engineer on his business code development. He "only" needs to focus on the modelling task. Hence the framework suggests:

- To hide in some way the modern software technology;
- To manage COM components communication;
- To provide the generic behaviour occurred in any UO development;
- To give implementation of common CO services (parameter, identification, errors management, ports, material objects);
- To hide the details of CO architecture and design in a way that process engineers do not need to be an expert on CO standard;
- To help for component deployment within CO compliant environment e.g. INDISS.

The key part of UO development (the business code that contains the core of UO behaviour) is implemented behind a CO interface layer. This part uses the elements of the framework, by using the available objects and by subclassing generic elements according to object-oriented benefits. For example, a UO can inherit from the *GenericUnitOperation* class, and thus gets the run-time functionality of retrieving data automatically from *CO ports* and *CO parameters* objects.

2.2 Framework architecture

The top-level architecture is represented on Figure 1. The framework is divided in four packages. *Unit Operation Utils* gathers generic UO support of dynamic draft version, automation of streams data retrieval from CO structures (parameter and port) and, handling of UO identification. *Thermo Mgr* provides generic thermo wrapper and optimisation call for properties calculations. *CAPE-OPEN Utils* encloses simulation context abstraction, factory of CO parameters specification and, standard CO objects (parameter, port ...). *Misc* includes persistence management and time management for dynamic simulation.

2.3 Unit operation design

The UML class diagram displayed on Figure 2 shows the modelling of a unit operation (in that case, a simple valve). We use the stereotype notation (i.e. the text between << and >>) to identify the various elements on the diagram. The classes stereotyped as <<framework>> are the classes of the framework that can be used as is to implement

the UO (*CGenericUnitOperation*, *CUODataMgr* ...). The <<coclass>> is automatically generated by the COM wizard of Microsoft Visual C++ tool, as well as the skeleton of the class stereotyped as <<implementation code>> which is necessary in order for the code to be used as a COM component. Finally, the class *CExampleValveImpl* stereotyped as <<business code>> contains the UO specific code.

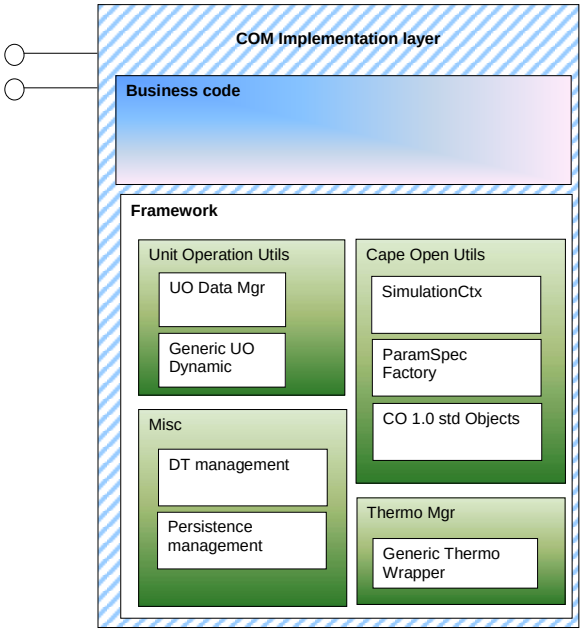

Figure 1 : Framework architecture

The class *CExampleValveImpl* looks familiar to a programmer because it uses standard types (double ...) and defines methods that are close to the methods of CO interfaces and that need a particular processing, specific to the business code in the *CExampleValveImpl* class such as *Calculate()* for static computations, *StartTimeStep()* and *Calculate()* for dynamic computations, *Validate()*...
The UML diagram can easily be derived to a valid Visual C++ project, and after a standard build step, installed on any computer by registering the component. It can be used in any CAPE-OPEN compliant simulation environment such as INDISS.

2.4 Unit operation development method

Because quality is a requirement the framework can be integrated in a software process development. RSI uses it in conjunction with classic development tools such as Rational Rose and SoDA, Parasoft test tools and Microsoft development tools (Visual C++). The integration with these tools enable the development of UO to be integrated in a software development process, thus allowing the software developed with the framework to be tested against software quality standards.

3. Use of framework in education

In addition to the common use in an industrial context that takes benefit from the framework, we have proposed an innovative education program. This teaching

experience takes place in the *ENSIACET School* of *INPT University* and includes industrial contributions. The main objective is to teach *information technologies*, especially open architecture and component based approach, and then to apply these concepts to *process engineering* using an industrial tool, INDISS, and a CAPE related standard, CO. Using the framework, the students from *"Process Engineering and Computing"* department develop and deploy their compliant unit operations in INDISS flowsheets with the aim of process control. Previous programs give the students a set of prerequisites. Then this program intends to study more advanced technologies and to work with industrial applications. The students practise (i) *information technologies* with software architecture, standards, middleware and interface concept, component based approach, UML/C++ component development and, (ii) *process engineering* with hybrid dynamic simulation, sequential modular approach, made-to-order simulator development, operator training system. The pedagogic link between these both topics is done by the *CAPE-OPEN technology* especially because of its application in CAPE domain, its component based architecture and its dynamic unit operation interface specification.

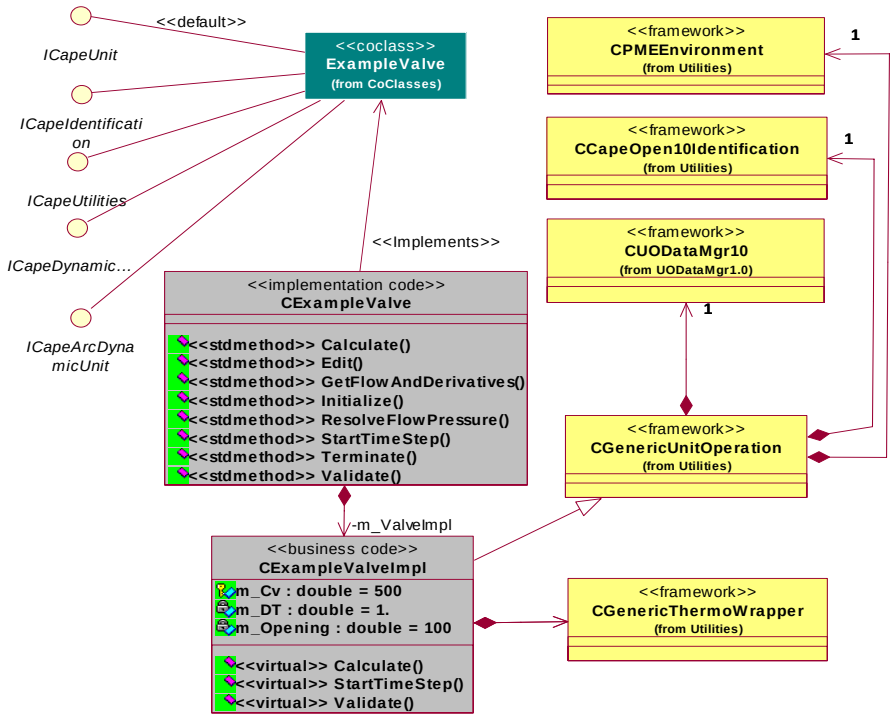

Figure 2: A class diagram

The students exercise in workshops three kinds of problems according to *a web solutions development view*, *a process engineering view* and *a process software solutions integrator view*. The latter allows students to develop their own CO compliant dynamic unit operation making use of framework. Then they plug the resulting

component into the INDISS environment. The framework provides a basis for the technical layer such as CO Parameters, constructors, destructors, etc. as explained in previous section. The students open a *Visual C++ workspace* specific to CO compliant dynamic UO development. Once the framework design understood through UML diagrams and C++ source files analysis, they search and implement the process engineering model following the design of figure 2 (through the business codes such as *CExampleValveImpl* class). The component is an element with pressure loss, a valve. Other actions are requested (for example to add CO parameters …). Finally they build the project and generate the associated COM component dll file. They register, deploy the compliant UO and integrate it in a *depropaniser* flowsheet designed with INDISS environment. They run the flowsheet and validate the dynamic behaviour of their components. Details, feedbacks and reactions of this teaching experience are proposed in Belaud (2004).

The real and main difficulty in the development of such ambitious program is how to manage the pedagogic complexity coming from the mix of modern computing, dynamic simulation, a CAPE standard and the use of industrial tools. As a first shot to solve partially this complexity we take advantage from the framework that provides technical and abstract classes and, from component inner concepts such as wrapping and interoperability of objects.

4. Conclusion

We present in this paper a framework that aims at easing the implementation of CAPE-OPEN compliant (static or dynamic) unit operation. It contains a set of C++ classes that provide high level abstractions of CO elements, as well as utility classes for the automation of repetitive tasks. This framework can be used not only for rapid prototyping of unit operations but also for training sessions. We validate the approach by submitting it to students in process engineering and computing. They use the framework to get operational CO compliant unit operation. The students act as process engineers when they design common dynamic simulation with INDISS tool. They are process solutions integrators when they develop using the framework their components enclosing specific process information.

References

Belaud J.P., An education program involving CAPE-OPEN standard, 2004, CO Update Volume 8, April 2004, Ed. CO-LaN

Belaud J.P. and Pons M., 2002, Open software architecture for process simulation, Computer-Aided Chemical Engineering, 10, May 2002, Elsevier, pp 847-852, ISBN: 0-444-51109-1

Braunschweig B. L., Britt H., Pantelides C. C. and Sama S., 2000, Process modelling: the promise of open software architectures, Chemical Engineering Progress, September issue, pp 65-76

CO-LaN, 2004, CAPE-OPEN Laboratory Network, http://www.colan.org

Pons M., Belaud J.P., Banks P., Irons K. and Merk W., 2003, Missions of the CAPE-OPEN Laboratories Network, Proceedings of foundations of computer-aided process operations, 2003, Coral Springs, Florida, United-States

Roux P., Belaud J.P. and Pons M., 2003, Opening unit operations for process engineering software solutions, AIDIC Conference Series, Vol. 6, AIDIC & Reed Business Information S.p.A., 2003, pp. 35-44, ISBN: 0390-2358

RSI, 2004, http://www.rsi-france.com