Conceptual Design Document (CDD2) for CAPE-OPEN Project

CAPE-OPEN

by
CAPE-OPEN Project Team

Version 4 January 2000

NOTE This is an interim report of the CAPE-OPEN project and gives a summary of the state of conceptual research and development reached by November 1997. We make it public in order to share it with the wider process systems technical community and other interested parties as early as possible. While every effort has been made to make it internally consistent, it is recognised that some aspects will have to be further refined and modified. Thus the document has been approved for release by both the Scientific and Technical Committee and the Steering Committee of the project on the basis that modifications may be carried out to conceptual design as experience and learning are gained through the project.

Project BE 3512 is funded by the European Community under the Industrial and Materials Technologies Programme (Brite-EuRam III), under contract BRPR-CT96-0293.

1.0 Executive Summary

Quick and effective process modelling is increasingly vital for the synthesis, design, monitoring and optimization of chemical and related processes. Different simulators have different strengths, and to obtain the best results for a particular problem, access to more than one vendor simulator and in-house software containing company specific methods or data is usually required. For this reason, the EC is sponsoring the CAPE-OPEN project, which aims to develop, test, describe and publish agreed standards for the interfaces of components of a process simulator. The main objective is to enable native components of a simulator to be replaced by those from another independent source or a part of another simulator with minimal effort in as seamless a manner as possible.

Since January 1997, the following partners have been collaborating on this project:

- Operating companies: BASF, Bayer, BP, DuPont, Elf, ICI, IFP
- Simulator vendor companies: AspenTech, Hyprotech, SimSci
- Academic institutions: RWTH Aachen, INP-Toulouse, Imperial College
- Software consultant: QuantiSci
- Administrative manager: Gerth

CAPE-OPEN will be funded by the EC until June 1999, and upon completion of the project, prototypes of CAPE-OPEN compliant components will have been developed. It will then be possible to assemble a process model from a set of software components encapsulating physical property methods, unit operation models and numerical algorithms, as pictured in Fig.1-1. Here, thermodynamic calculations, the numerical solver, the physical properties database, the simulation executive, and even the fine-grain Equation of State (EOS) originate from different vendors, whereas one unit operation is an in-house development (not an uncommon scenario). These components could be newly developed software or wrapped legacy code, communicating through the commonly agreed open standard interfaces represented by arrows.

The Conceptual Design Document CDD2 is the first public domain deliverable of the project. It presents the scope of our work, our architectural and technical choices and our relationship with other initiatives such as pdXi. We will continue to produce public-domain deliverables in the form of Standard Interface Specifications Drafts for simulation components, and we welcome suggestions from the CAPE (Computer-Aided Process Engineering) community on how to improve these proposed standards.

Feedback on this and other CAPE-OPEN related matters should be sent to:

- the Conceptual Workpackage Leader: Tom_Malik@ici.com
- the Project Coordinator: Bertrand.Braunschweig@ifp.fr
- or any CAPE-OPEN partner.

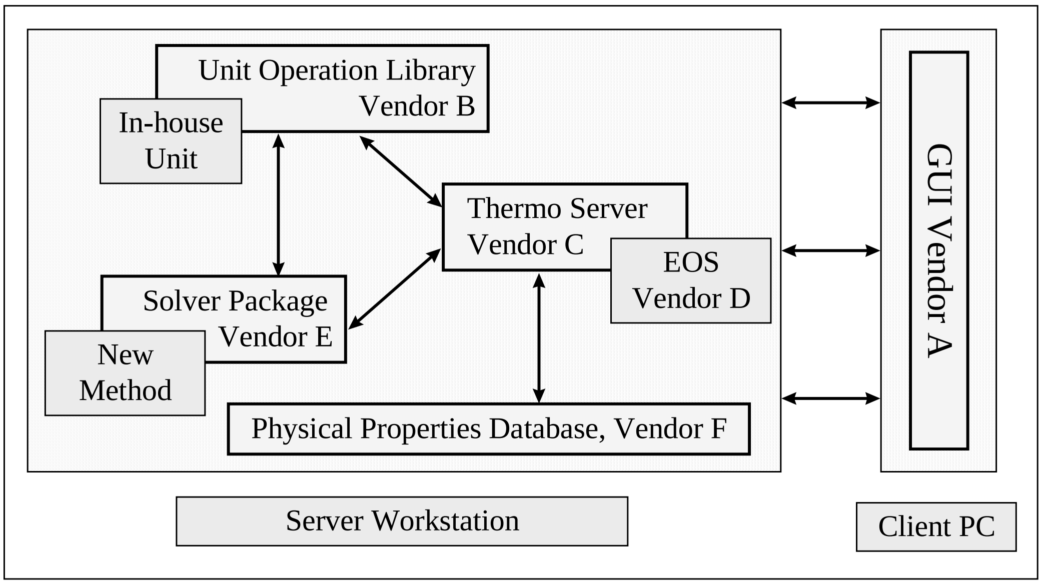


Figure 1-1: Simulator A Host Modified/Enhanced by CAPE-OPEN Compliant Components from In-house or Other Sources

2.0 Technical Executive Summary

This is the key document in the CAPE-OPEN project for **Conceptual Design** and this summary describes the key decisions made, both as a reminder for those continuing work within the project and for the wider technical community.

CAPE-OPEN is an EC sponsored project running from January 1997 to June 1999 with the aim of developing, testing, describing and publishing agreed standards for interfaces of components of a process simulator. Its objectives are to enable native components of a simulator to be replaced by those from another independent source or a part of another simulator with a minimum of effort in as seamless a manner as possible. Analogies with standardized interfaces for hi-fi systems or for electrical plugs and sockets can be offered to ease understanding of the concepts, but here we are concerned about the ability of software programs to work interactively in different host environments and yet retain the ability to make sensible responses. The project partners are BASF, Bayer, BP, DuPont, Elf, ICI, IFP among the operating companies, Aspen Technology, Hyprotech and Simulation Sciences among the vendor companies, RWTH Aachen, INPT-Toulouse and Imperial College London as the academic institutions, Quantisci as a software consultant and GERTH as administrative manager.

CDD2 has been written following an intense period of deliberations, debate, rationalisation and consensus seeking in conceptual matters after the project kick-off meeting on January 20, 1997 and particularly after the initial first draft CDD1 released three months later. The conceptual design has benefited from about 40 experts in process systems technologies, computer science and applied mathematics from the 14 organizations represented in the project. The work has been carried out through hundreds of technical discussion papers or technical correspondence, dedicated working group meetings (of Priorities, Framework, and Methods and Tools sub-groups established at the kick-off meeting), conferences of individual work packages, conferences of the Scientific and Technical Committee of the project (that particularly sought consensus from the key technical players), the Steering Committee (that maintained an overview) and also discussions and experience gained from the UML notation (see below) training work-shops that were organized in France and the UK.

Through the passage of this phase, clear evidence has emerged that the project has captured the imagination of all the groups represented. The **academic organizations** have contributed excellent papers particularly on addressing the fundamental issues afresh in light of the present day and future expected trends in technologies and some key concepts have been included in conceptual design as a result. The technology and tool **vendor organizations** have contributed vigorously to fundamental concepts as well as providing a realistic, practical outlook in the context of providing working prototypes of their tools complying with CAPE-OPEN standards. They are committed to providing working prototypes of compliant components and simulator executives of their present simulators. **Operating companies** have continued to express their needs, contribute to technical progress and lead many of the project activities. There is also evidence that this project addressing the concept of open standards has captured the imagination of the wider technical community in particular in the PSE’97, ESCAPE-7 meeting in Trondheim and the AIChE November 96 meeting. During a panel based discussion at PSE’97-ESCAPE 7 the general consensus of both the panel and the audience appeared to be highly supportive.

A subgroup helped to determine the combined **priorities** of the partners. Initially these were discussed in meetings, then written inputs were obtained from each partner on their perceived priorities. Subsequently a pre-designed proforma was sent out so each partner could respond to the same questions concerning different categories of priorities. The responses were analysed and a set of combined priorities worked out. If two or more partners considered a requirement essential with general support from the remainder then it became a combined priority. These are classified under various headings:

1. **Types of components** in simulators that will be interchangeable; this includes Thermodynamic/Physical Properties Packages, data for chemical species within the packages, individual physical properties methods within these packages, unit operation models within simulators of the same type and between different types (see next paragraph), numerical methods for solving dense or sparse sets of nonlinear algebraic and differential-algebraic (DAE) systems;
2. The **general types of simulators** that will be catered for e.g. sequential modular simulators and equation based simulators; **specific simulators** for which the interfaces will be applicable, e.g. Aspen Technology, Hyprotech and SimSci products and in-house company simulators - note that the interfaces are expected to be applicable to any process simulator;
3. The **material forms** that the simulators will be able to handle, e.g. uniform fluids defined as molar compositions of specific chemical species or of pseudo-components, electrolyte mixtures, reacting fluid mixtures, polymer solutions, and particulate systems;
4. The **physical properties methods** that will be catered for e.g. the properties required for the calculation of vapour-liquid-liquid-solid equilibria or subsets thereof, other thermodynamic and transport properties;
5. The run time performance of the CAPE-OPEN compliant simulator in comparison to its native version is an important issue
6. The standards will cater for both steady state and dynamic modelling applied to continuous, batch or mixed processes. **Off-line simulation**, **optimization**, **parameter estimation** and **data reconciliation** would be covered;
7. Other topics included **hardware and software systems** to be supported, **maintenance** and **error recognition capabilities**.

The conceptual discussions have largely been aided through conceptualisation of a process simulator in terms of the relatively coarse (traditional) components of thermodynamics package, unit operation models, numerical routines and the simulator executive.

Whereas the granularity and type of components will in itself be a subject of research within the PATH work package, the traditional view of a simulator has been invaluable in developing the practical concepts and work-plans for the project. The methods and tools sub-group has determined that the standard interfaces can then be defined, expressed and prototyped using an **object oriented** approach. The notation for the interfaces models is the **Unified Modelling Language (UML)**. The languages for the interface specifications are MIDL and IDL. Existing **middleware ActiveX/COM** and **CORBA** will both be used in order to make the standards durable. The interfaces should allow the encapsulation of **legacy code**. The specifications in UML, MIDL and IDL will be made available on a web server for the wider community.

There is consensus that a comprehensive way of dealing with materials in a simulator, the so-called **‘material concept’** is the right way forward for CAPE-OPEN. This concept has emerged through an object oriented approach that defines precisely all entities, their characterizing attributes, the relationship between them and the associated set of methods. A finite list of material types has been proposed, the most important of these have been determined through the prioritisation exercise and mentioned above.

A key part of any simulator is the **physical properties package**. It has been decided that the interfaces will be restricted to SI units (this does not prevent the internal units of measurements to be different). It is agreed that **standard names** will be required to unambiguously identify individual properties, state and other global variables. If possible, this list will be taken over from existing sources, e.g. IK-CAPE, DIPPR or pdXi/STEP. The properties themselves are organized into classes, those that are constant properties of pure components, those that depend upon the state variables of pure components and the mixture properties. A preliminary list of properties for each class has been written. Information from the OS-CAPE, IK-CAPE and pdXi sources has been used where appropriate and possible. The **physical and chemical equilibrium server** is part of the physical properties package and works out phase equilibria and chemical equilibria through a single generalized call. If required the physical and chemical equilibria can also be carried out by a unit operation with properties called from the physical properties package. For the **physical properties database interface** the options of a CAPE-OPEN defined neutral file as well as pdXi will be investigated.

For the **simulation databank interface**, that concerns the interface for the input data and the storage of results from a simulation run, the Bayer and pdXi options will be investigated. Such a standardized interface offers the possibility of being able to start a simulation run based upon one set of results possibly from another simulator, possibly carried out a long time ago.

It will be possible to use a compliant **Unit Operation** model component without modification, compilation or re-linking after installation in a compliant simulator. Forward compatibility of the compliant unit operations with future versions of simulators compliant with the CAPE-OPEN standard will be given very high priority. The unit operation interacts with the simulator executive, thermo and numeric parts during different phases of simulation. These are: 1) **flowsheet creation** when a unit operation is added to the flowsheet and the simulator interrogates the unit operation to check its usage is correct; 2) **initialisation** when the model parameters are read; 3) **solution** when the simulator evaluates each unit operation and the unit operations in turn invoke methods from the simulator executive, thermo or numeric objects; and 4) **reporting** results are requested by the simulator executive and reported back in various formats.

Within the numerical area, a number of ‘numerical objects’ have been identified. These are **Full or Sparse Linear Algebra Objects (FLAO/SLAO)**, **Nonlinear Algebra Objects (NLAO)**, **Differential Algebraic Equation Object (DAEO)** and **Optimization Object (OPTO).** The purpose of these objects is to represent and solve full or sparse sets of linear equations, nonlinear equations using residual and possibly Jacobian information, differential-algebraic equations and nonlinear programming respectively. It will be possible to use external compliant numerical objects within a compliant simulator.

It has been concluded that the internal simulator **streams** do not need to be standardized for CAPE-OPEN to succeed. Instead the concept of **ports** is introduced and will be standardized. A port provides the means for transferring quantities between the simulator executive and the various interchangeable components. The same consistent approach can be adopted for all information flow in and out of elements for material, energy or numerical data.

The role of a CAPE-OPEN compliant **simulator executive** has been examined in some detail and broadly agreed upon. The simulator executive provides the primary user interface for the simulator including its local components and for any foreign components, e.g. an imported unit operation model. However, specific interfaces for the components may also be invoked from the simulator executive. The unit operation specification can be via a **native user interface** provided by the unit operation component or it can be via the **generic interface** provided by the executive. Items that apply to the flowsheet as a whole (such as stream specifications, global constraints, design specifications that may span more than one unit operation, optimization problem specification such as the global variables and the objective function and variables that can be set or freed according to the problem specification) are to be handled within the executive. Each simulator and its executive are free to maintain their unique stream structures. However, there will be standard definitions for the materials that flow through the ports that connect to the unit operations.

Validation is considered to be an integral part of the software development process and it is intended to start the validation activity as soon as possible and to carry out testing before the final standards are settled. There will be **elementary testing** using a test harness on individual components such as unit operations, physical properties and numerical procedures.

The second level of testing will be **integration testing** where CAPE-OPEN compliant components will be tested together with a compliant simulator executive. Both component and executive will be provided by the same party. The resulting performance will be monitored on selected tests and will be compared to the performance when using both the component and the executive with their native (non CAPE-OPEN) interface. The last level of testing is the full **plug and play functionality** involving different component software authors and simulator executives from different simulator vendors.

The **PATH** work package will provide a window to advancing simulation and software technology in order to make the results as durable as possible and to take account of future trends in simulator technology. This work package will carry out **research activities** and look at alternative solutions not necessarily covered in the mainstream of the project. A minimal **testbed** environment will be made in order to assess new software technologies, evolving software standards and alternative design concepts. PATH will seek to assess the future architectures of process simulators without being constrained by the existing systems. It will also consider **distributed and concurrent computing technologies** covering subjects such as platform interoperability, component selection on the internet and component granularity.

It is intended to thoroughly consider the work of **related initiatives** and to coordinate activities to gain two-way synergies wherever possible. In particular, it is expected that discussions will take place with the Process Data Exchange Institute (**pdXi**), an initiative of the CAST (Computer and Systems Technology) Division of AIChE to promote the electronic exchange of process engineering data. pdXi has addressed the development of a generalized data model (the AP-231) and the development of a utility for accessing data that conforms to the defined model. Within the CAPE-OPEN project, the applicability of the pdXi data models will be considered particularly for **Physical Properties Database Interface** and **Simulation Databank Interface**. **CAPE-NET** is an EC sponsored network that is now being launched and there are likely to be synergies between that and CAPE-OPEN. In particular, the standards can be widely discussed and disseminated through the network among other means.

3.0 Introduction

3.1 Process Simulators and Process Models

Process simulators, often termed flowsheet simulators, are tools which are used in order to create models of manufacturing facilities used for processing and/or transforming materials. Chemical manufacturing through continuous or batch processing, polymer processing, and oil refining are examples of such processes. Process simulators are important tools for designing new processes and they are also used extensively to predict behaviour of existing or proposed processes. They are being used increasingly for process control, process optimization, and process operator training among other uses. Process simulators follow different internal architectures, e.g. block modular systems, equation based systems and simultaneous modular systems.

Process models may represent either the steady-state or dynamic behaviour of a process, and both are used extensively. Dynamic models are used to describe how a process behaves over time when its processing conditions are changed, e.g. following a disturbance to feed flow rates, or altering operating conditions of one or more unit operations. They may also be used to study process start-up and shutdown, or to investigate the effect of operator mistakes or modifying a process control scheme.

CAPE-OPEN is providing the overall conceptual design and interface specifications for simulators which consist of an assembly of relatively large software components. For example, the entire unit operation model library of a simulator may be replaced by that from another, or just one unit operation model may be substituted or added within the existing library. Similarly, for thermodynamic methods either the entire set may be replaced or one of the constituent methods is substituted or added. This approach will create a much more flexible environment than exists with conventional simulation systems. CAPE-OPEN will also enable simulator components that originate from different types of simulators to be mixed with each other. For example, an equation based unit operation model may be combined with a sequential modular host simulator. While this will lead to increased power, there are also some research issues that will need to be addressed. It will be intended that simulator integrity precedes the power in importance. Thus if mixing two types of components may lead to violation of simulator integrity then appropriate messages will be given to warn the user.

One example benefit will be the ability to develop and test a working component which models behaviour of materials used in some particular process, and then use this validated and well tested component with different types of commercial or proprietary simulator systems. This capability will provide a major advance for internal consistency in the industrial use of various types of simulation.

A second example benefit will be the ability to develop highly specialized, reusable models of unit operations which are not available as part of some standard set - multi-tube reactors are a classic example. These specialized components will then be reusable in the range of models for different applications, from simulation for design to online optimization.

3.2 Framework for Conceptual Design

Simulators differ widely in architecture and implementation, but all have common functionality imposed by the underlying modelling tasks which they address. This functionality can be summarized in terms of four key ‘conceptual’ component types:

- **Simulator executive:** this component is the simulator executive (based upon a simulation executive) which is responsible for: installing other components and registering them with the operating system; managing interactions with users; accessing and storing data; reporting and analysis of simulation calculations.

- **Unit operations:** these components represent physical processing unit operations, and possibly perform specialized roles such as performing additional calculations to support process optimization.
- **Physical properties:** a critical simulator functionality is the ability to model properties and behaviour of the materials which are used or created by the process. Properties and behaviour includes both thermodynamic and transport properties.
- **Numerical solvers:** this includes both the specialized mathematical methods used to evaluate the equations that describe a UNIT OPERATION (unit solving), and the methods used to evaluate the overall flowsheet (flowsheet solving).

The CAPE-OPEN standards will be defined in terms of this conceptual design, but will impose no requirements on the actual architecture and implementation of a compliant simulator. Instead it is anticipated that both simulator vendors and implementors of CAPE-OPEN compliant components will initially to a large extent ‘wrap’ existing code such that it presents the correct software interface. In future designs they will be able to incorporate the standards directly into their designs so the code will not require the wrapping layer.

This will ensure that the existing functionality of a simulator (including its user interface, unit operation library, physical property systems etc.) is protected while offering immediate access to additional functionality when necessary.

The core simulator into which CAPE-OPEN compliant components are installed is referred to as the *host* simulator and its built-in features as *native* components.

3.3 Design Objectives of the CAPE-OPEN Standard

The CAPE-OPEN standard will be designed so that the value and usefulness of existing commercial simulators, and that of in-house process modelling systems, is preserved. This is consistent with an approach of defining standard interfaces for a small number of large components (termed coarse granularity). For some types of components, e.g. thermodynamic or numerical methods, interfaces at the component level as well as at the individual methods level will be provided. Nonetheless, commercial and in-house developers will have the option to implement compliant components using finer grain approaches which implement a capability to extend or modify underlying unit operation behaviour. CAPE-OPEN will also research into the finer granular component systems.

The CAPE-OPEN design objectives include provision of upward compatibility with anticipated future versions of the standard. There is an expectation that such future CAPE-OPEN standards will support advanced architectures including distributed computing, concurrent computing using multiprocessor machines, and Web technology. CAPE-OPEN will undertake complementary research into many of these areas.

A stated design objective for the CAPE-OPEN standards is to achieve a minimal computational performance penalty in simulation models which essentially duplicate existing architectures. While component architectures may involve some computational overhead, it is certain that overall computing performance advances (both processor speeds and architecture) will outweigh any unavoidable computational overhead that results from CAPE-OPEN’s coarse grain granularity approach.

3.4 Design Methodology

CAPE-OPEN software interfaces will provide communication between the components of a simulator. Analysis and design of these interfaces depends heavily on the various ways in which simulators are used and on the interdependency of simulator components as they carry out various tasks. All work packages are making extensive use of the Unified Modelling Language (UML) analysis tools in order to support the analysis and conceptual design of CAPE-OPEN standards.

UML provides different views of planned and proposed software systems. Use cases, a key part of UML, document the functional interactions between actors and a software system. An actor may be an external user of the system or some defined subsystem. Each individual Use Case captures some user-visible function, which may be small or large, and which achieves some discrete goal for the actor. For example, actors identified for the unit operation analysis domain include simulation builder, simulation user and internal components such as a flowsheet manager and the unit operation itself.

3.5 The Conceptual Work Package

The conceptual design within CAPE-OPEN has been carried out as part of the Conceptual work package following the inaugural project meeting on 20 January 1997 in Barcelona. The kick-off meeting of the work package was attended by nearly 40 persons from the 15 organizations within the project. The tasks within the conceptual work package were discussed and allocated to each organization, each other work package and to three dedicated conceptual work groups that were agreed during the meeting. These were, 1) Priorities, 2) Architectural framework and 3) Methods and tools. There are two main sub-tasks within the work package, that of conceptual design and conceptual review. Whereas the latter will take place throughout the project, the former is substantially complete with the release of this document. Conceptual design is the foundation upon which the rest of the work in the project is based.

3.6 Timing of CDD Documents

The development of conceptual design has been a stepwise process. Figs. 3-1 and 3-2 show the timings and the work processes envisaged at the kick-off meeting. The first document, CDD1 was released after three months of the project start. It was meant to be a draft working document that contained a substantial amount of material and a lot of ideas which were not necessarily reconciled as yet. It was the objective of CDD2 to present a reconciled and reviewed design. CDD2 was to be a statement of the essentially complete conceptual design.

The expected time for completion of CDD2 was six months after the start of the project. However, it has taken longer than that to get it fully reviewed and agreed upon. The entire project has regarded the conceptual foundation to be of crucial importance for the success of the project and more time taken at this stage to be worthwhile in order to avoid delays later.

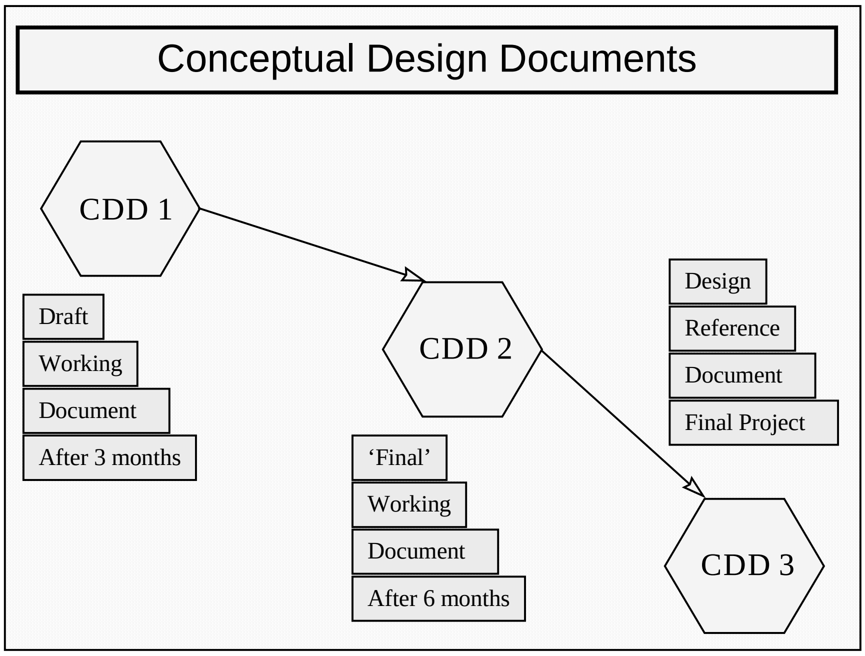


Figure 3-1: Timings for Conceptual Design Envisaged at the Kick-Off Meeting

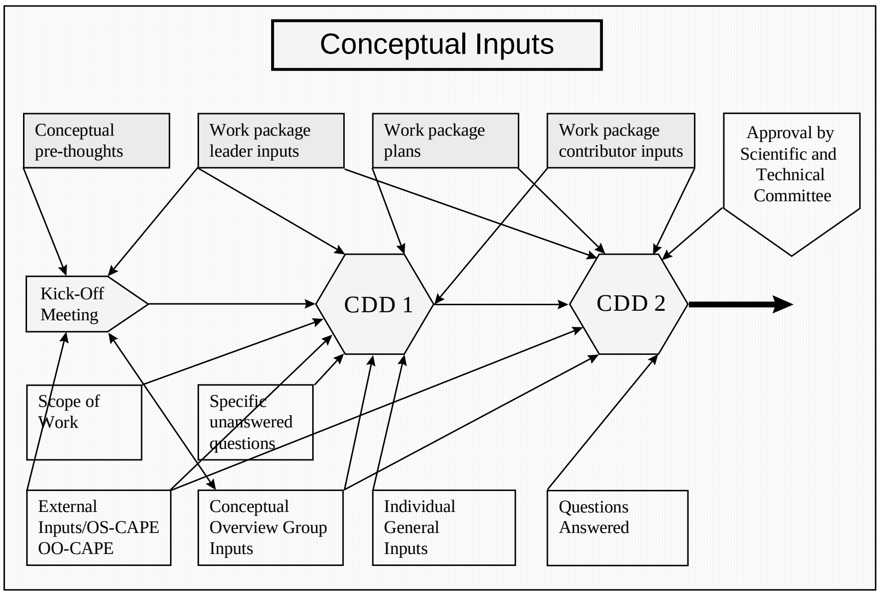

Conceptual Inputs
Conceptual pre-thoughts
Work package leader inputs
Work package plans
Work package contributor inputs
Approval by Scientific and Technical Committee
Kick-Off Meeting
CDD 1
CDD 2
Scope of Work
Specific unanswered questions
External Inputs/OS-CAPE OO-CAPE
Conceptual Overview Group Inputs
Individual General Inputs
Questions Answered

Figure 3-2: Inputs to Conceptual Design Envisaged at the Project Kick-Off Meeting

3.7 The Contributors to this Document, Review and Approval Process

Some 40 experts from the participating organizations have contributed to the conceptual design in CAPE-OPEN. There have been hundreds of technical discussion papers or significant technical correspondence. It was necessary to reach consensus and conclude the fruitful discussions. To this end a number of sections were identified that needed to be completed for inclusion in CDD2. An author was appointed to each section to be responsible for writing a concise statement of the design for that area. Often the authors were not the persons who contributed the most to the design itself but were brought in to help to reach a conclusion. Each section was reviewed by a few persons who were often the key contributors to the conceptual work in that area. Table 3-1 gives a list showing the authors and the reviewers of the different sections.

The strategy was to have a small team responsible for completing the sections following the more general earlier debate that had taken place. The guideline was for comments from the reviewers to be individual and private and for the author to satisfy fully each reviewer through modifications to the earlier drafts. It should be acknowledged that several names not mentioned here have made very significant contributions to conceptual design.

The document itself has been put together by ICI by combining the different inputs and Version 1 was distributed to each organization for further comment and approval. At this stage most of the comments that were returned related to structure, formatting and display with most of the conceptual integrity issues having been resolved in the earlier steps. Further comments have been incorporated by ICI and Bayer. Before release outside the project, the document has been approved by both the Scientific and Technical Committee and the Steering Committee of the project.

Section Number - Section Title		Author responsible (reviewers)
1	Executive Summary	Bertrand Braunschweig (Allison Howard, Tom Malik)
2	Technical Executive Summary	Tom Malik (Peter Banks, Dave Smith, Stefan Artlich, Herb Britt, Bertrand Braunschweig, Wolfgang Marquardt, Sergi Sama, Hans-Horst Mayer)
3	Introduction	Tom Malik, Mike Davies (Wolfgang Marquardt, Sergi Sama, Hans-Horst Mayer, Peter Banks, Bertrand Braunschweig)
4	Essential Requirements for CAPE-OPEN	Nii Asante, Bill Johns (Ron Chartres, Renee Esposito)
5	Role of a CAPE-OPEN Compliant Simulator Executive	Herb Britt (Rafique Gani, Sergi Sama, Stefan Artlich)
6	Streams and Ports	Bill Johns, Peter Edwards (Wolfgang Marquardt, Xavier Joulia, Philip Vacher, Jean Pierre Belaud, Laurent Jourda)
7	Unit Operations Component	Mike Davies (Costas Pantelides, Peter Banks)
8a	Thermodynamics and Physical Properties System	John Grogan (Werner Drewitz, Malcolm Woodman)
8b	Materials Concept	Xavier Joulia, Laurent Jourda, Jean-Pierre Belaud (Werner Drewitz, Bill Johns)
9	Numerical Component	Renee Esposito, Jean-Pierre Belaud (Mike Williams, Ben Keeping)
10	Validation	Stefan Artlich (Herb Britt, Rafique Gani, Sergi Sama, Philip Vacher, Wolfgang Marquardt)
11	Path Conceptual Design	Peter Edwards (Nii Asante, Sergi Sama, Wolfgang Marquardt)
Appendix A1	Methods and Tools within CAPE-OPEN Project	Bertrand Braunschweig (Mike Williams, Stefan Artlich, Richard Martin) Based on entry in CDD1 authored in addition by Ralf Bogusch, Stefan Zlatintsis, Andy Lui, Michael White, Curt Arnold, Joe de Almaida, Michael Williams. We also acknowledge Michael White's previous work in this area that has been incorporated here.
Appendix A2	What is UML?	Richard Martin (Tom Malik, Costas Pantelides, Tony Kakhu)
Appendix B	Process Examples	Philip Vacher (Renee Esposito, Tom Malik, Stefan Artlich, Jean Pierre Belaud)
Appendix C	Relation with pdXi and Other initiatives	Rafique Gani (Herb Britt, Peter Edwards)

Table 3-1: The Authors and Reviewers Responsible for CDD2 Sections

3.8 Explanation of Document Layout

This document is laid out in a sequence that is top-down. Following the summary and introduction chapters, the first chapter on a simulator component is the executive. This is followed by the other components, the validation and research activities and other useful information in the appendix.

Chapter 1 gives an executive summary. This is for the executive that only needs an overview of the project but is not concerned with the details of the conceptual design.

Chapter 2 is a summary of this document and contains the essential technical decisions reached in the project. It is intended to be a reminder to the project personnel as CAPE-OPEN progresses and is also for informing the technical personnel in the wider community who are interested in the CAPE-OPEN project.

Chapter 3 (this chapter) introduces the conceptual design and this document.

Chapter 4 gives the CAPE-OPEN priorities and essential requirements as determined by a consultation exercise following the kick-off meeting. These are combined priorities for the project as a whole and the criteria used are explained within the chapter.

Chapter 5 describes the role of a CAPE-OPEN compliant simulator executive. It defines the functionality and the services expected from such an executive and its relationship with the other components.

Chapter 6 explains the concepts of streams and ports which are closely related to the executive and to the other simulator components (particularly the unit operations). These concepts have been developed through a significant amount of original debate in the project. Essentially they enable interfaces to be defined without imposing restrictions on the internal stream structures of a given simulator.

Chapter 7 treats the unit operation model interface. There are interfaces to the simulator executive, the unit operation interfaces with physical properties and numerical components. Other issues such as configuration of unit operations, reporting by unit operations, persistence and archiving of unit operations and the prototype implementation are also covered.

Chapter 8 covers the thermodynamic and physical properties interface and other items covered by the THRM work package. It defines the three areas to be addressed by it - the Property and Equilibrium Servers, the Physical Properties Database Interface and the Simulation Databank Interface.

Chapter 8 also describes the material concept for CAPE-OPEN. All simulators include the facility to model material streams. It follows that simulators need to provide means to model the behaviour of matter within these streams. The chapter aims at giving a possible way that CAPE-OPEN could define and model matter by employing an object-oriented approach. The proposed model provides a framework that handles materials commonly arising in CAPE-OPEN physical property libraries. At the same time, it is readily extendible to handle other materials identified as important by CAPE-OPEN partners.

Chapter 9 describes the interfaces for the numerical components. It considers the numerical solvers, e.g. linear and nonlinear algebraic equation solvers and DAE solvers. Also, the optimization solvers and sequential modular tools for flowsheet solving are considered.

Chapter 10 describes the work and conceptual design of the validation work package. The philosophy of validating the CAPE-OPEN compliant components is expounded and the types of tests specified.

Chapter 11 describes the work and conceptual thinking of the PATH work package. This work package concerns the future technologies expected and part of the research will in itself address conceptual issues in simulation modelling. Other items of interest are making use of distributed computer technology, multi-processing and concurrent computing. The plans for a CAPE-OPEN repository are also described.

Appendix A considers the methods and tools that will be used in the CAPE-OPEN project, both for the progress of work itself within the project and methodology to be used to describe the standards. The formal notation and interface definition languages selected for the CAPE-OPEN project are described. The expected middleware to be used and the philosophy with regard to both COM/ActiveX and CORBA is expounded. The second part of Appendix A is a tutorial for managers and technical personnel in the UML methodology, an object oriented technique that will be used within the project. Whereas more detailed treatises are available elsewhere on this subject, it is considered valuable to have a concise summary for easy reference here. The most widely used individual steps and diagrams of the UML methodology are described including Use Cases, Sequence Diagram, Class Diagram and Interface Diagram.

Appendix B contains a set of real process examples selected through consultation with the CAPE-OPEN partners, in particular the operating companies. It is planned that examples of this kind will be modelled and solved in different ways for demonstrating the value of the CAPE-OPEN project to the process modelling community and managers within industry. Whereas the examples in the validation work package are for testing the integrity of the components and the standards defined, these examples are for demonstrating the value of CAPE-OPEN to industry. Thus, some may be developed on one modelling package to begin with, to be modified subsequently to incorporate simulator components from different sources.

Appendix C relates the work of CAPE-OPEN with other relevant initiatives, in particular pdXi, CAPE-NET and SVPPM. These related initiatives are described and possible synergies between CAPE-OPEN and them explored.

4.0 Essential Requirements for CAPE-OPEN

This section presents the primary or essential requirements of the CAPE-OPEN project, as identified by a survey of all the partners in the project. After consultation to identify the key issues, each partner was asked to rate a list of possible goals for CAPE-OPEN as “essential”, “desirable” or “not needed” (the latter covers also “outside the scope of the project”, thus a goal that might be desirable in itself but did not fit with the other work of the project). The criterion that was used for judging that a requirement was essential to the success of the project was that at least two non-academic partners considered it essential. Where only two such partners considered it essential, general support from the rest of the partners was taken into account (that is, the extent to which other partners considered it “desirable”). Goals not considered as “essential” by the commercial partners, but rated highly by academic partners, are of course quite appropriate to the research element of the project in work packages such as “PATH”. This chapter is, however, concerned only with the priorities of the commercial partners (it being recognized that, even meeting these priorities requires research as well as development).

Capabilities that (although not initially essential) are seen by the majority of the partners as desirable features are listed as desirable requirements. Where it is possible to meet these goals with marginal effort over and above that required to meet the “essential” goals, the project will attempt to meet them. Certainly they will be included in the early conceptual studies so that a fair assessment of the effort required can be made.

To facilitate referencing, all features/requirements are assigned a two-letter identifier and a number (based on the classification used in the survey). The two-letter identifiers are given at the head of each subsection.

Section 4.1 gives the “essential” requirements and 4.2 the “desirable” requirements. The chapter omits features that were considered to be irrelevant or undesirable by CAPE-OPEN partners.

4.1 The CAPE-OPEN Project Primary/Essential Objectives

4.1.1 Interchangeable Elements (EL)

- Thermodynamics/Physical property systems **(EL1)**
- Data for chemical species within physical properties packages (and adding new species) (**EL2**)
- Individual physical properties methods within physical properties packages (**EL3**)
- “Raw” physical property data sets (**EL4**)
  This facility enables data regression to be performed in order to fit the parameters in any chosen form of physical properties correlation.
- Physical properties methods for as yet undefined properties (**EL5**)
- Unit operation models within simulators of the same type (**EL6**) (see 4.1.2)
- Unit operation models from one simulator type into another (**EL7**) (see 4.1.2)
- Numerical methods for solving dense sets of nonlinear algebraic equations (**EL8**)
  (a standard component of steady-state sequential-modular and similar simulators)
- Numerical methods for solving sparse sets of nonlinear algebraic equations (**EL9**)
  (a standard component of steady-state equation-based systems)

- Numerical methods for solving nonlinear differential and algebraic equations (**EL10**)
  (a standard component of dynamic simulators)
- Numerical methods for constrained and unconstrained nonlinear optimization (**EL11**)
- Numerical methods for solving large sets of linear equations (**EL13**)
  (both dense and sparse solvers)

Over half the members of the CAPE-OPEN project supported the view that it was essential that each of the above elements should be interchangeable. In most cases almost all considered that the elements must be interchangeable, and in virtually all cases, those respondents that did not consider interchange "essential" considered it to be "desirable". Support for the interchangeability of the following element was at the minimum necessary for it to be considered a CAPE-OPEN essential requirement (namely two members), but most members considered it desirable:

The priority for the following has yet to be assigned:

- Partitioning/Tearing/Sequencing for modular simulators **(EL12)**
- Data reconciliation packages (**EL14**)

4.1.2 Simulator Types (TY)

- Sequential Modular Simulators, such as Aspen Plus and Pro II (**TY1**)
- Equation-Based Simulators, such as SPEEDUP (**TY2**)
- Sequential and Non-sequential Modular Simulators, such as HYSYS (**TY 3**)
  These differ from sequential simulators in that, for most commonly occurring unit operations, the direction of computation through a unit operation is not predefined.

The ability to handle each of the above simulator types were considered to be essential by virtually all CAPE-OPEN members. The following was supported by 30% of members which, although less than the more frequently occurring types, still makes the requirement sufficiently important to be essential for the project:

- Modular Hierarchical Simulators (**TY 4**)
  In these simulators, the unit operation models can be built up from sub-models. These sub-models can correspond to physical parts of the modelled unit operation (for example, individual stages of a distillation column) or simply distinct blocks of the computation (often with the objective of removing iterative calculations from within the unit operation computation). Simulators capable of breaking unit operation models down into a hierarchy of sub models achieve their goal in a wide variety of ways. This requirement is not interpreted as a requirement that the method of building models from sub-models will be standardized, but rather a requirement that the sub-models themselves can be interchanged.

4.1.3 Specific Simulators (SI)

Simulators for which interfaces shall be available for interchanging software components are:

- ASPEN simulation products: Aspen Plus, DynaPLUS, SPEEDUP (**SI1**)
- HYPROTECH simulation products: HYSYS (**SI2**)

- SIMSCI simulation products: Pro II, Protiss (**SI3**)
- Existing in-house systems from operating companies in the project (**SI4**).

It is one objective of the CAPE-OPEN project that operating companies will be able to interface their in-house components easily to commercial simulators to eliminate the necessity for maintaining their in-house systems. It follows for this case that it is only the in-house components that need to meet CAPE-OPEN standards, not the in-house executive programs.

Virtually all members considered that the above degree of interchangeability was an essential minimum for the project to claim success.

4.1.4 Neutral Files (NF)

The following requirement was supported as "essential" by only two non-academic members, thus reaching the minimal level of support to be considered "essential". It was, however, considered "desirable" by the majority of members and was put forward as one of the original motivations for the CAPE-OPEN project. It is, therefore, considered to be an essential requirement which is very widely supported.

- Interfaces to pdXi or some other neutral file (**NF1**)
  They will be provided for transmitting information between systems (for example, initializing a dynamic simulation from the steady-state simulation of another system). They can also store physical propertyand unit operation data

4.1.5 Material Forms (PH)

The following requirements refer to the methods used to specify the material state. Thus, for PH1, it is sufficient to define the molar proportions of each chemical species present in the fluid. For PH4, it is not possible to define all the proportions independently because one or more chemical equilibria reduce the degrees of freedom available for specification. For (PH5), more information than merely the molar proportions is required because the size and shape of the particles needs to be known before all bulk properties can be computed.

- uniform fluids defined as molar compositions of specific chemical species **(PH1)**
- uniform fluids defined as mixtures of pseudo-components **(PH2)**
- uniform fluids with electrolytes **(PH3)**
- uniform fluids with species in chemical equilibria such as dimers **(PH4)**

The above were all considered to be essential to CAPE-OPEN by the great majority of respondents (over 75%) and essential or desirable by all correspondents. The following were considered to be essential to CAPE-OPEN by half the respondents and desirable by nearly all the others.

- particulate systems (**PH5**)
- polymers and polymeric mixtures (**PH6**).

4.1.6 Physical Properties (PP)

The following list is largely consistent with the list of material types identified in 4.2.4 above and has similar support. Thus nearly all respondents want properties appropriate for uniform fluids. Similarly, the proportion that want other materials (e.g. particulates) handled also want to be able to compute the appropriate properties for these materials. The CAPE-OPEN "essential" list is then:

- properties required for the calculation of vapour-liquid equilibria (**PP1**)
  (for example, fugacities and activity coefficients)
- properties required for the calculation of equilibria of more non-ideal mixtures (**PP2**)
  (for example liquid-liquid and vapour-liquid-liquid equilibria)
- properties for the calculation of vapour-liquid-solid equilibria (**PP3**)
  (for example, vapour pressures and activity coefficients)
- other thermodynamic properties for vapours, liquids and solids (**PP4**)
- transport properties (**PP5**)
  (for example, viscosity, thermal conductivity, and diffusivity)
- as yet undefined properties of fluid systems (**PP6**)
- properties of particulate systems, for example specific surface and permeability (**PP7**)
- properties of polymer systems, such as flow properties for extrusion, setting temperatures and strength (**PP8**)

4.1.7 Process Characteristics (PR) and Application Areas (AP)

It was considered essential by virtually all respondents that CAPE-OPEN compliant systems must be able to deal with simulation of both steady-state **(PR1)** and unsteady (dynamic) **(PR2)** continuous processes. Over half of the respondents state that it is essential that batch processes **(PR3)** can be handled. There was, however, little support for including stochastic simulation of batch processes within the scope of the project. Semi-batch processes are processes which include both batch and continuous stages, for example a semi-batch process may include equipment capable of running continuously such as pumps and heat exchangers along with batch reactors and batch filters. Support from two non-academic partners makes this class of process essential to CAPE-OPEN **(PR 4)**. On the other hand, it is probable that by providing interfaces suitable for both batch and continuous plant, negligible effort will be required to extend the interfaces to include semi-batch plant.

Although the primary focus of CAPE-OPEN is on simulation, simulation tools are now being extended to include facilities to assist in improving the design and operation of these processes. Respondents considered which of these emerging application areas should be considered essential to the success of CAPE-OPEN. As expected, straight-forward off-line simulation **(AP1)** was considered essential by all partners. Off-line optimization of design parameters and operating conditions **(AP2)** was considered essential by over 75% of partners and off-line data reconciliation **(AP4)** by nearly half.

4.1.8 Performance (PE)

When components from the same supplier are interfaced using the CAPE-OPEN standards there shall only be a small loss in run-time performance as compared to the "native" interfaces **(PE1)**. When interfacing components and systems from the same supplier, use of the CAPE-OPEN interface shall lose no functionality compared to the "native" interface **(PE2)**. Both of the above requirements were strongly supported. It should be noted that they should not be taken to imply that that all the functionality of the original systems should be available through CAPE-OPEN; some of this functionality will be outside the priority areas identified by CAPE-OPEN. The original functionality will, of course, still be available using the "native" methods of the respective simulators. Similarly, there may be occasions when run-time performance is degraded because the native systems take advantage of proprietary facilities. Such use of non-standard facilities is deliberately outside the scope of CAPE-OPEN; the restriction is not likely to lead to any long-term disadvantages.

4.1.9 Hardware and Software (HS)

There was virtually a unanimous requirement that the CAPE-OPEN standards shall cater for stand-alone PC's running MS-Windows **(HS1)**. Over half of the respondents also considered it essential that CAPE-OPEN should apply to stand-alone UNIX boxes **(HS2)**. Two non-academic partners stated that it was essential that CAPE-OPEN also apply to stand-alone VMS systems **(HS3)**, and just over half felt that this was desirable. On the other hand, nearly as many respondents felt that VMS should definitely be considered to be outside the scope of CAPE-OPEN. The implication of this response seems to be that the project should attempt to ascertain early on what the implications in man-months of effort will be to include VMS in the standards. If the effort required is high, discussions should be held to review or confirm our commitment to VMS. With regard to networks, there was again universal support for all-PC networks with NT servers **(HS4)**, all-UNIX networks **(HS5)** were supported by well over half of the respondents, and heterogeneous networks with UNIX servers and PC clients **(HS6)** by a majority of respondents.

In relation to software with which CAPE-OPEN standards should communicate to/from, there was virtually universal support for FORTRAN (77 and 90) **(HS12)**, and for C **(HS13)**. Over 80% also supported links to/from C++ **(HS14)**, and around 50% to/from Java **(HS15)**.

4.1.10 End Users (US)

CAPE-OPEN standards have to be designed with a variety of end-users in mind. These may range from domain experts (e.g. in chemical reaction kinetics) with no or limited programming experience to people with substantial expertise in the technologies used to set up the CAPE-OPEN standard interfaces. There was virtually universal support that the CAPE-OPEN standard interfaces should be immediately usable by FORTRAN Programmers **(US1)** with no other special computer expertise, by expert system programmers familiar with middleware interfacing **(US2)**, and by programmers familiar with other languages (e.g. C++), who do not necessarily have special expertise in the interface technologies **(US3)**. There was also a requirement that CAPE-OPEN systems should be usable by simulator end users with no specific programming skills **(US4)**; thus components should be able to be exchanged on a "plug and play" basis without requiring any programming.

4.1.11 Maintenance (MA)

All of the following requirements were considered essential by the great majority of respondents and desirable by all. CAPE-OPEN standards must provide a mechanism for adding new functionality beyond the duration of the project **(MA1)**. Adding such new functionality shall not require any re-coding of the components, interfaces or systems previously interfaced **(MA2)**. In addition, given clear descriptions of the native interfaces, it shall be possible for a user to write a "wrapper" to the CAPE-OPEN interface without requiring access to the source code of the component being wrapped **(MA3)**. This requirement implies that, if a commercial simulator includes clear instructions as to how to add, for example, a new unit operation model, it should be possible to add a CAPE-OPEN conformant model without access to the simulator source code or the model source code. Additional code will, of course, have to be written to take, for example, calls from the commercial simulator and convert them into the form needed by the CAPE-OPEN software component. It may even be necessary to re-link one or both pieces of code (simulator or component), but it should not be necessary to modify the source code of either. This requirement describes both a realistic scenario and, by implication, defines a maximal level of complexity that can be tolerated in the standard interfaces.

4.1.12 Error Handling (ER)

It is recognized that each commercial simulator is unique at least in some respects. Thus they differ not just in the form of their argument lists, but in the information that they use and require. For example, some simulators propagate derivatives in order to solve sets of nonlinear equations by Newton methods, others do not. Similarly, some simulators calculate only outputs from inputs for a unit operation model. Others allow the calculation of the unit operation model whenever a suitable combination of inputs and outputs have been defined to satisfy the mathematical degrees of freedom. To take a component from one system and place it in another, thus gives rise to a potential for many more errors than when all the components are developed for just one simulator. Unless CAPE-OPEN takes steps to localize errors as far as possible within the relevant components and to the interfaces at which the incompatibilities occur, there will be scope for difficult-to-trace run-time errors. If a simulation based on CAPE-OPEN compliant components repeatedly fails with difficult-to-trace errors, CAPE-OPEN will fall into disuse and the project will have failed. As a consequence, CAPE-OPEN members generally stated that they considered a high level of error handling an essential component of CAPE-OPEN. Specifically, the following two requirements were considered essential by virtually all respondents:

- recognition of system differences (**ER1**)
  (for example, the recognition that an attempt has been made to interface incompatible systems, which if not trapped might give obscure run-time error)
- transmission of error messages (**ER 2**)
  (Ability to transmit an error message [such as “value out of range” or “failure to converge”] to the simulation executive, rather than simply fail or return wrong values).

The following two requirements were considered essential by half the respondents and desirable by all:

- correction of incompatibilities (**ER 3**)
  (Ability of the interface to make good minor incompatibilities between simulation systems [for example, providing enthalpy values or derivatives required by one component, but not delivered by another. Similarly, the ability to discard superfluous values/derivatives]).
- transmission of information relating to the reliability of calculations (**ER 4**)
  (for example, signaling a warning if a correlation is extrapolated beyond its valid range).

4.2 Desirable Features of the CAPE-OPEN Standards

The desirable features are presented below generally much more briefly than the essential features described above.

4.2.1 Interchangeable Elements (EL)

The following were all considered to be essential by at least one non-academic partner, and were considered essential or desirable by over half the respondents. Some like MINLP (Mixed Integer Nonlinear Programming) and Control Design Tools were supported by two non-academic partners but positively objected to by more partners than supported their "essential" inclusion in CAPE-OPEN. In general, it may be concluded that these features are high priority, although not initially "essential". The project, at least at the conceptual stage, should thus explore the implications of designing standard open interfaces for the relevant components, although not necessarily carrying the work through to the extent of building prototypes to demonstrate the interfaces.

- MINLP Packages (**EL15**),
- Control Design Tools (**EL16**),
- Continuation methods for solving nonlinear algebraic equations (**EL18**)

4.2.2 Simulator Types (TY)

No additional desirable requirements.

4.2.3 Specific Simulators (SI)

The following links were considered to be desirable by a majority of respondents. On the other hand, virtually no respondents considered the links to be essential, and the response rate for these features was only about 60%. Their priority within the current CAPE-OPEN project must, therefore, be considered to be relatively low.

- The ability to link with gPROMS, Prosim, Belsim, and Tisflo (**SI6**), languages/scripting systems that enable users to define performance requirements or constraints outside the main simulation package (**SI7**), CFD packages (**SI8**), pipe flow network simulators (**SI9**), and other systems: MATLAB and SIMULINK (**SI10**).

4.2.4 Material Forms (PH)

- The ability to handle porous **materials** with high specific surface (**PH8**). Examples include catalysts.

4.2.5 Physical Properties (PP)

No additional properties were identified. (The "essential" list is already comprehensive in that it requires CAPE-OPEN to be able to handle additional properties not already noted).

4.2.6 Process Characteristics (PR) and Application Areas (AP)

A number of further application areas were identified as "desirable":

- The ability to **apply** the CAPE-OPEN interfaces in systems that generate simplified models (**AP3**).

Examples of such systems would be the generation of simplified models for online control or for operator training. Another instance would be a simulator that used two physical properties Material Templates for the same material. In this approach, a simple fast-to-compute template is used for routine computation which includes adjustable parameters that are refitted from time to time against a second, very accurate, but relatively slow-to-compute template.

- Online optimization and data reconciliation (**AP5**),
- MINLP Optimization (**AP7**).

4.2.7 Performance (PE)

No further requirements.

4.2.8 Hardware and Software (HS)

- The ability to work with stand-alone VMS systems (**HS3**) is clearly on the borderline between “essential” and “desirable” and is discussed further under 4.1.9.
- Other heterogeneous networks (**HS7**). Specifically, respondents referred to networks including VMS. This desirable requirement was given as “essential” by one non-academic respondent and was stated to be desirable by most. It must, therefore be considered to be of fairly high priority.

5.0 Role of a CAPE-OPEN Compliant Simulator Executive

This section describes the role of a CAPE-OPEN compliant simulator executive and lists the CAPE-OPEN interfaces and services required. Relevant material would also apply to any other host application that uses CAPE-OPEN components, such as an equipment design program that supports the CAPE-OPEN Thermodynamics interface.

A CAPE-OPEN compliant simulator executive (SE) must provide the primary services listed below. It is assumed that the host simulator executive is the user's "everyday" simulator and that a typical simulation would consist primarily of native unit operation models with a mix of CAPE-OPEN "foreign" components, which are highly functional and typically of small size compared to the simulator executive or an entire process simulator. Thus emphasis is given to maintaining the familiar host simulator executive user interface and behaviour.

Another general principle is that any service or information that cannot be provided by a stand-alone Unit Operation, Thermo, or Solver components should be provided by the simulator executive. For example, a unit operation should never make itself a request for information about the flowsheet topology or the global specifications of a flowsheeting problem. If this information is needed in the unit operation the user will define the appropriate information streams (see Section 6 for details). During run time the information is then provided by the simulator executive.

Anything that is overly specific in this chapter is so only to illustrate the concepts. It is assumed that the formal design process will determine the actual interface definitions and other details.

5.1 User Interface

The simulator executive provides the primary user interface for building and running a simulation, optimization, or other flowsheeting problem. However, CAPE-OPEN component user interfaces may be embedded in or invoked from the simulator executive user interface.

5.2 Flowsheet Topology

The top-level flowsheet topology (streams and their connection to unit operations) is defined, viewed, and stored within the simulator executive, and is known only to the simulator executive. However, a unit operation may represent a subflowsheet (e.g. for a refrigeration section of a plant), in which case the subflowsheet topology would be viewed within the Unit Operation's user interface. The subflowsheet appears to the simulator executive as a single unit operation, both when configuring and when solving the top-level flowsheet. The internal structure and user interface for subflowsheets should not be a subject for CAPE-OPEN standards. Just as the internals of a Property Package are understood only by the Property System that generated it (see Section 5.10), the internals of a unit operation that represents a subflowsheet need only be understood by the model server that generated it. In fact, the fact that the unit operation represents a subflowsheet would not be known to the simulator executive or any other CAPE-OPEN component.

Interface requirements: Unit operations must be able to provide the simulator executive with the number and the properties of allowable stream connections. This is done through the mechanism of "ports". A port has a name (which should indicate the location within the unit operation to allow the user to make a proper connection, e.g. "column feed" or "top product"), a direction (inlet or outlet), and a type (material, heat, work). The port definition does not include "*Material Template*", although a *Material Template* will be assigned to any instance of a port in a flowsheet. It is then up to the unit operation to determine if it can handle a material object presented to it through a port (see 5.12.2 Material Objects and Section 8.8).

Prototype suggestion: It is proposed that we deal only with multi-phase conventional fluids. We might even deal with a fixed structure, thus deferring the definition and handling of extensible *Material Templates.*

5.3 Unit Operation Specification

Unit Operation specifications may be provided in either of two ways:

5.3.1 Via a Native User Interface Provided by the Unit Operation Component

In this approach the unit operation is configured using a native user interface. However, there are two ways of doing this:

External configuration

The unit operation is configured and validated stand-alone, preferably using the same property package that is used for the unit operation in the larger flowsheet. The unit operation is then saved or exported as a CAPE-OPEN Unit Operation file. As a CAPE-OPEN Unit Operation, the original feeds and products become the CAPE-OPEN ports, with the stream IDs as the port names. The model server can either be a simulator (e.g. Aspen Plus or HYSYS) or a server for a specific model type (e.g. MyReactor) but must be able to be run stand-alone. For models created from flowsheet simulators, this amounts to entering the native simulator user interface and configuring one or more unit operations (i.e. a subflowsheet).

The CAPE-OPEN compliant simulator executive has a generic CAPE-OPEN interface model in its model library. When placed in the flowsheet, this model displays the file browser, and the user selects the appropriate file. Loading the file creates a handle to the model server. The CAPE-OPEN interface model can then query the component for its ports and connect flowsheet streams to the unit operation.

This scheme is similar to creating a spreadsheet in Excel, and later inserting it in Word using Insert Object Create from File. A variation would be to add a New button to the CAPE-OPEN interface model, which would allow you to select a model server (Aspen Plus, SPEEDUP, ProII, HYSYS, gPROMS, etc.) and create a new model using the server's GUI, invoked from the host simulator executive using in-place editing. This scheme is similar to creating an Excel spreadsheet directly within Word using Insert Object Create New. Once created, the unit operation could be edited using in-place editing. Alternatively, public unit operation parameters can be modified via the Public Variable interface.

Integrated configuration

In this approach, registered CAPE-OPEN models appear directly as additional Unit Operations in the list of available ones and can be placed directly in the flowsheet, where it can be configured as any other Unit Operation. Whenever the unit is edited, be it as the initial configuration or after having been run to modify its parameters, the Unit's native user interface is displayed. This scheme is somewhat analogous to creating an Excel spreadsheet directly in Word using the Insert Excel Worksheet toolbar button.

Although using the Unit Operation's native user interface it is intended that it will not be necessary to have the complete native process simulator installed just for configuring the unit operation model for the use together with the non-native simulator executive.

5.3.2 Via Generic User Interface Services Provided by the Simulator Executive

In this approach, the model relies on the simulator executive to provide standard CAPE-OPEN user interface services, operating on public variables exposed by the model (through the Variable interface). This standard interface could be as simple as displaying a list of input parameters in a spreadsheet format.

Note that a supplier of a typical FORTRAN model, without a GUI, would have two choices for GUI services:

1. Rely on the standard CAPE-OPEN user interface services, use scheme 5.3.2
2. Use the GUI tool-kit of one of the simulation vendors to create a CAPE-OPEN compliant model specific user interface, use one of schemes in Section 5.3.1.

In either case, a wrapper provided by one of the simulation vendors would typically (though not necessarily) be used to turn the FORTRAN model into a CAPE-OPEN component.

Prototype suggestion: The individual work packages will decide which configuration scheme will be used for producing their component prototype. Nevertheless it will be assured that in the end there will be simulator executive prototypes available which will allow the exchange of all three component types (Unit Operation, Physical Property, Numerical).

5.4 Flowsheeting Problem Definition

Stream specifications, global constraints (e.g. design specifications spanning more than one unit operation), optimization problems, set/free variables, and other flowsheet level specifications are defined within the simulator executive, using native language and/or GUI facilities (not CAPE-OPEN standard ones). For some of CAPE-OPEN's objects to work, though, the simulator executive must provide them with the ability to access this information. For instance, a Unit Operation must be able to modify the values of its outlet streams.

Interface requirement: A standard interface to public unit operation input and results variables is required. For example, a design specification in a sequential modular simulator executive might manipulate an input variable of one unit operation to achieve a desired value of a quantity computed from the results of another unit operation. In an equation-based simulator executive, this specification could be expressed as a global constraint. A mechanism is needed to display the unit operation variables for the user to choose from, and to Get and Put variable values. The display and selection mechanism could be a standard service provided by the simulator executive (e.g. display a simple list of exposed variable names), or the unit operation component could display a native variable browser.

5.5 Flowsheet Solution

The simulator executive assembles the flowsheeting problem and solves it. This is true regardless of which flowsheet solution method is used: sequential modular, equation-based, or simultaneous modular, and whether the unit operations provide modular interfaces, equation-based interfaces, or both. A simulator executive may use CAPE-OPEN Solver components during solution, but it is assumed that the simulator executive communicates directly with the unit operations to set up the problem structure, to pass inputs (streams, parameters, or state vectors), or to retrieve results (streams, parameters, or equation residuals and Jacobians). For example, it is up to the simulator executive to perform the mappings, perturbations, etc. to use a sequential modular unit operation in an equation-based flowsheet solution.

Interface requirements: These interfaces are defined in the Unit Operation and Numerical sections. Basically, `Solve`, `Evaluate`, and `Initialize` methods are needed.

5.6 Reporting

The simulator executive provides the primary mechanism for reporting the results of a flowsheeting problem, either graphically or as documents. For example, stream results are reported by the simulator executive. Results from the CAPE-OPEN components can be provided in either of two ways:

1. Each component can provide its own report section, to be assembled by the SE. If the component provides its own user interface (5.3.1), the component's native plotting and other GUI facilities can be used.
2. The Simulator Executive CAPE-OPEN standard user interface services (5.3.2) may be used, accessing public UNIT results.

Interface requirements: CAPE-OPEN components need an optional REPORT method.

5.7 Error Handling

The simulator executive provides mechanisms for handling errors generated by CAPE-OPEN components and shutting down the run if necessary. It is felt that this is a particularly difficult and important area for CAPE-OPEN to address. It is not clear how successful the user will be in diagnosing flowsheeting problems that may be caused by interaction with "foreign" components, and it is not clear how to handle components dying in the middle of a simulation, temporarily slow communications, etc.

Interface requirements: Three types of information need to be handled:

1. An error code standard for type and severity needs to be defined to enable the simulator executive to know what to do after a unit operation is executed.
2. The simulator executive must be able to display text messages generated by unit operations for the user. The user should be able to control the detail of information displayed.
3. Middleware related handling of component status and errors.

5.8 Persistent Storage

The simulator executive provides the master file storage for a simulation using a compound storage mechanism provided by the CAPE-OPEN middleware framework. This same structure, provided the OLE compound document architecture is used, provides an approach for OLE Automation access to component data and methods.

Interface requirements: A Persist method is needed.

5.9 Neutral File

The format for the neutral file is yet to be agreed (see Section 8.6 Simulation Databank Interface). One possibility is the use of the pdXi neutral file format. In this case, the simulator executive provides the top-level user interface for accessing or generating pdXi neutral files. Generating or querying a pdXi file is similar to generating a report. The simulator executive calls the pdXi API to get or put the stream data. Then it requests the CAPE-OPEN components to call the pdXi API directly to get or put their internal data. A simulator executive cannot provide pdXi services for unit operations, since the simulator executive does not know the semantics of public variables.

Interface requirements: `Get_pdXi` and `Put_pdXi` methods are needed. (in case pdXi format is selected, alternative names if alternative format is selected)

5.10 Properties

5.10.1 Property Packages, Property Systems, and Option Sets

A Property Package is a complete, validated, reusable collection of chemical components, methods, property constants and model parameters for a process application, which might be a superset of those needed for a given simulation. Method and data references are packaged into one or more *Option Sets* to allow for different conditions within the process. Even though a Property Package may contain more than one Option Set, it is also possible for a simulation to use more than one Property Package (possibly from different Property Systems).

A Property Package is created within the environment of one Property System (e.g. PROPERTIES PLUS or HYSYS) and is understood only by that Property System. A CAPE-OPEN compliant Property System implements CAPE-OPEN interfaces both for communication with clients (unit operations and simulator executive) and for incorporating CAPE-OPEN components, so that "foreign" properties, data, and calculation methods can be included in a Property Package. For example, a PROPERTIES PLUS Package may use a liquid density model from PRO/II properties. A Property Package can be stored in file (format is proprietary to the generating Property System) and can be used in many simulations, possibly opened "read-only" to prevent changes by simulation users.

An Option Set is a complete set of method and data references for calculating properties, including options like whether to apply a Poynting correction, and special methods for some components, such as water. A Property Package may contain more than one Option Set, to handle different conditions in the flowsheet. Each Option Set carries descriptive text information to help the user make a selection and to understand the methods and data it uses, range of applicability, etc. There is no standard definition for an Option Set, since each property system structures them differently. So an option set is only a pointer to information understood by the property system. The term "option set" is used here in a general sense, not as in the specific implementation of Aspen Plus. We probably need a different, neutral term, such as Calculation Set to avoid confusion.

5.10.2 Properties and Simulation Executive

CAPE-OPEN property packages are defined externally to the simulator executive and loaded via the simulator executive user interface. Optionally, run-time CAPE-OPEN property evaluation interfaces may be provided for properties defined natively within the host simulator executive property system. When more than one Property Package is defined for a simulation, selection of an Option Set implies selection of the Property Package from which it came.

Creation and loading of property packages can be handled similarly to the External configuration approach for unit operations. A Property Package is a file (or part of a file) understood only by its Property Server. When a Property Package is loaded, the simulator executive gets a handle to its Property Server, and can then invoke the interfaces.

Interface requirements: When a simulation is being built, the simulator executive needs to get the list of components and the list of option sets defined in the Property Package. The simulator executive builds an aggregated list of all of the option sets from all of the loaded property packages and then asks the user to select one of them as the default option set. Depending on the features of the simulator executive, option sets may also be selected for flowsheet sections, individual unit operations, streams, etc. The simulator executive also needs to make the option set list available through an interface to the unit operations for those unit operations that can use multiple option sets (such as a heat exchanger).

The above approach could vary significantly depending on how *Material Template*s develop. It is quite likely that option sets will be associated with *Material Template*s, rather than with flowsheet objects directly. In this approach, the flowsheet object (which could be the simulator executive), asks the material object to evaluate some properties, using the option set associated with it. See Material Objects below.

Run time property evaluation interfaces are defined in the Thermo section. The simulator executive may invoke these interfaces to calculate stream properties, flash streams, etc.

5.11 Chemical Species

The same chemical component may have different ID's in different property packages. Simulator executive's will use the CAS number to identify which chemical species a component represents. How components are handled also depends greatly on the way materials are handled. It is quite likely that component lists, like option sets, will belong to materials. Even so, the simulator executive probably needs to build and maintain a consolidated list of components from all the loaded property packages.

5.12 Assumptions and Issues

The following assumptions and issues greatly affect the detailed requirements and design of the simulator executive interfaces and services.

5.12.1 Ports

It is assumed that there are no CAPE-OPEN standard definitions for streams. Rather there are standard definitions (actually extensible *Material Template* definitions) for the material passed through Ports (see 5.12.2). It is up to unit operations to map between their internal data and ports, and for the simulator executive to map between streams and ports. It is also up to unit operations to report back their ability to handle a particular *Material Templates*. Thus a port is simply a "socket" on a unit operation to connect a stream to. Once connected, material objects can "flow" through ports. But the port itself has no material definition.

5.12.2 Material Objects

It is assumed that some object representation for materials, along the lines proposed in Chapter 8.8, will be adopted. It is further assumed that this representation will be used to exchange information both between unit operations and the simulator executive (i.e. via ports), and between unit operations and property servers. Such an approach neatly encapsulates material characterization and properties.

The following is anticipated:

Option sets and component lists will be associated with *Material Templates* instead of simulations, streams and unit operations directly. Thus a material object will form the complete characterization of a material *and* the methods and data for calculating properties of the material. Property evaluation methods can be invoked directly on the material object. Because Option Set is a property of the material object, the appropriate Property System and methods and data are used transparently to the user (e.g. a unit operation model).

Material Templates, or more precisely the interfaces by which properties are obtained through the Templates, must be defined as CAPE-OPEN standards. These interfaces must be accessible from 3 of the 4 major components from which any simulation is built, namely the Simulator Executive, the Unit Operation Models and the Physical Properties System. For example, CAPE-OPEN may define a Fluid Material as a component list, an option set, a phase assumption (e.g. V, L, VL, VLL, LL), and phase subobjects (flow rate, temperature, pressure, mole fraction), plus a list of calculated properties for the total material and each phase.

The simulator executive user interface will provide a mechanism for assigning *Material Templates* for the flowsheet, for a section of the flowsheet, and for unit operations and streams in the flowsheet, depending on the native capabilities of the simulator executive to do so. Unit operations can also assign different *Material Templates* internally. Approaches must be agreed for handling mismatched component lists.

It is still to be determined where the **Material Template** objects will be created. Component lists and Option Sets and phase assumptions used for regressing VLE data are all fixed in a Property Package.

An advanced CAPE-OPEN simulator executive would be able to carry material objects it did not understand with flowsheet streams, so long as the unit operations connected to the stream could deal with material. Such an advanced approach would require the following standards for material objects:

1. Identification of state or accelerated variables for flowsheet convergence.
2. An input mechanism for specification of feed streams
3. A mechanism for defining and creating *Material Templates*, probably as part of a Property Package

5.12.3 Client/Server

These issues have not dominated the conceptual design activity but it is recognized that the design of simulator executive GUI and runtime interfaces requires them to be considered.

5.12.4 Equation-Based Interface

It is assumed that SOLVER components deal only with solving sets of equations that have already been structured globally for the flowsheeting problem by the simulator executive. Thus structure, residuals, and derivatives are provided directly and independently to the simulator executive by equation-based Unit Operation components.

6.0 Streams and Ports

“Streams” and the related concept of “ports” are defined and discussed in this chapter. The term “stream” is used to describe different internal representations that existing simulators use to record the different types of flows that exist in physical processes that are being modelled. From a conceptual viewpoint, these simulator streams may be classified into three main types, namely material streams, energy streams and information streams. A material stream represents the information needed to specify the physical state and flow rate of a material at some point in a process, typically as it enters or leaves a unit operation (see 6.2.1). Energy and information streams similarly represent energy and information flows within a physical process that is being modelled (see 6.2.2).

Internal streams used by simulators will not be standardized by CAPE-OPEN. Instead, standard ways will be defined for accessing and setting stream information. The term “port” is used to represent a software interface that enables contents of the proprietary simulator streams to be accessed. These ports provide a standard way to fetch data from the simulator executive and to return data to the simulator executive for subsequent use in the simulation model, e.g. as input to a downstream process block. One consistent approach will be adopted by CAPE-OPEN for material flow, energy flow, and information flow. Providing standard ports will enable strong encapsulation of unit operation models. The benefit of this will be that unit operation models can be developed independently, with the assurance that they will interface with any CAPE-OPEN compliant simulator.

The approach presented in this chapter is consistent with that put forward in Chapter 8 which proposes a standard framework for modelling matter.

6.1 Introduction

This chapter is written from a sequential modular simulation perspective. However, it is believed that the approach adopted applies equally to equation-based simulation.

In sequential modular simulation, a flowsheet model is described as a set of connected flowsheet blocks or unit operations. Flowsheet blocks typically represent processing unit operations but they may be used to represent other flowsheet elements. For example, flowsheet blocks may be used to represent feeds and products entering or leaving the actual process. Streams, which connect flowsheet blocks, may be conveniently classified into three main types. These are material, energy and information. A material stream will typically represent the flow rate, temperature and composition of a physical flow within the process that is being modelled.

In CAPE-OPEN, the term stream is primarily restricted to describing the internal representation used by a simulation executive to record flows of material, energy and information. These streams are an integral part of existing simulators and the CAPE-OPEN standards are defined so that there is no need for simulation companies to change their proprietary internal representations of streams. Instead, the CAPE-OPEN interoperability is achieved by introducing standard ports. Within a CAPE-OPEN process simulation, the port concept is a conversion standard that permits stream information stored in a simulator executive to be communicated to CAPE-OPEN compliant unit operations. The port also provides a standard way for communicating information back to the internal simulator stream. Thus a port is simply a conceptual, standard socket on a unit operation which connects to a standard interface provided by the simulator for its internal stream data.

Defining standard ports provides the flexibility needed in order to simplify interfacing existing and future unit operation models with different simulator executives. Initially CAPE-OPEN ports will provide standard access to material streams that are definable by mole fractions and/or molar flow rates, including streams that are defined by pseudo-components. However, the port standard is being defined in a way that will enable the future handling of streams containing reactive mixtures, electrolytes, particulates and polymers.

CAPE-OPEN has no work packages directly addressing streams and ports, but these concepts do provide a unifying link between the major work packages. Section 2 of this chapter discusses streams, and Section 3 discusses ports. The main standardization activity will concentrate on “ports” which provide standard access to stream information within the simulation executive. In this way, established simulators may retain their own internal forms and methods for streams, but will still allow CAPE-OPEN components to access the information in a standard way.

6.2 Streams

All existing process simulators implement the concept of a stream, which is used to represent the connection between flowsheet blocks. Streams may be classified into the three main types of material, energy and information. The most frequently occurring stream type in most process simulations is material. Material streams represent the physical flows of material from one process unit operation to another, for example material flowing through a pipe. Simulators provide stream information to simulator users in a variety of standard reports. However, each simulator stores its stream information internally in a proprietary form, which differs from simulator to simulator. CAPE-OPEN does not require the internal working of simulators to be modified. Instead, it enables the interfacing of existing or new simulator components from a variety of sources without recoding, recompiling or performing a separate linking operation.

The graphical user interface (GUI) of a simulator will display connections as going directly from one flowsheet block to another. In CAPE-OPEN, the flowsheet blocks seen in the GUI may represent unit operation models that may be implemented as separate CAPE-OPEN software components. The CAPE-OPEN design is based on the principle that connections between such blocks or unit operations are all handled by the simulator executive. This requirement that the simulator executive has control of the format of every stream arises because the simulator executive is responsible for organizing the solution of the overall flowsheet equations for material and energy balances.

To provide reliable standard access to these streams, it is necessary to understand the information that is stored in them. This information will be discussed in more detail in the following subsections.

6.2.1 Material Streams and Material Templates

The most complex streams to model are those which represent physical flows. Such streams must record sufficient information for all relevant properties to be extracted. For example, if enthalpy is not stored, there must be a method for computing enthalpy. It follows that calculation procedures must be associated with each physical stream. Most frequently this calculation capability is in the form of a thermodynamic package that enables properties of state to be computed. These thermodynamic packages will be accessed using a “Materials Template”, as described in Chapter 8.

CAPE-OPEN will not rely on how a “materials template” association is made by any particular CAPE-OPEN simulator. It will rely on the fact that, for any process stream, a template is unambiguously identified. Methods which simulators use to ensure access to the desired calculation methods will include:

- have one properties package that applies everywhere,
- have appropriate properties packages connected to each unit operation model
- have appropriate properties packages connected to each stream.

6.2.2 Energy and Information Streams

Energy streams are, on a parallel with material streams, used to represent energy flows in the absence of corresponding material flows (for example, transfer of heat energy across a conducting interface or transfer of mechanical energy through a shaft).

Information streams are employed to represent many other information flows important in process flowsheet simulation. An example of an information stream is a connection from a downstream unit operation that adjusts the division of flow between 2 output streams in an upstream unit operation. This “information stream” contains only one data item, the desired ratio of outlet flows. Typically flowsheet iteration will continue until a downstream physical flow achieves some desired value. As in this flow ratio example, information flows are often used to set values that would otherwise be set as fixed data items.

As with material streams, ports will be used to communicate between the internal representation of energy and information streams and the flowsheet unit operations.

6.2.3 Stream Flow Direction

Streams convey a direction of flow, so that all streams have a source and a destination. These sources and destinations are normally other unit operations in the process, but may also be process inputs and outputs. We have also indicated that a flowsheet block (such as a unit operation) will generally use values for its real world inputs in order to calculate values for its real world outputs. However, users of simulation models sometimes want to use a calculation direction that is different from the direction of physical flow. In response to this user need, models with this capability have been designed for certain types of unit operation. A successful simulator executive must, therefore, provide flexible support for flowsheet components within which the calculation direction may be different from the nominal or defined direction of material flow. Further, we note that for some processes, and for the corresponding simulation models, the direction of physical flow may change during the course of the simulation, e.g. in a dynamic simulation of a surge tank system.

6.2.4 Equation-Based Systems

This chapter focuses primarily on providing standard interfaces that allow modular components to be interchanged within and between modular systems. It is apparent, however, that standard CAPE-OPEN modular interfaces will apply equally to interfacing modular components with equation-based systems. The interfaces will also apply to components generated from equation-based systems that are to be interfaced with modular systems. Interfacing equation-based systems directly with other equation-based systems without generating intermediate, modular models is not discussed in this chapter. However, it is believed that essentially the same approach will apply.

6.3 Ports

A “port” is an abstraction of a location on a physical unit at which material, energy, or information (for example, measured pressure drop, valve position) enter or leave. The abstraction includes consideration of both software models of physical units and flows, and conceptual units and information which have no concrete physical analogue.

The specification of CAPE-OPEN standard ports removes the necessity for having standard stream structures in simulators or unit operation models. Such ports thus enable us to connect together simulation software components that employ different internal data structures. Clearly, however, where unit operation models (or simulators) do not internally employ data structures that correspond to CAPE-OPEN standards, they will have to include pieces of software that translate to and from the standard forms.

To be effective, CAPE-OPEN ports must provide high level standards for transmitting information specific to process engineering computation. For example, in a heat exchanger simulation, heat transfer coefficients may need to be computed which require, in turn, access to thermal conductivity, density, viscosity and heat capacity values. Whilst only a subset of values are transmitted directly through the port, the port provides methods for accessing the remaining values. As well as the high level standards, ports must provide mundane low-level standardization so that there is a guarantee that information exported by one port can be understood when imported by another. For example, we may agree on the use of IEEE standard 64 bit floating point numbers.

6.3.1 Benefits

The major benefit of considering ports, rather than streams, is that they enable unit operation models (and any other software component requiring similar access to information) to be considered in isolation. Thus in modelling a unit operation we only have to consider the flow of information into and out of that one unit operation, not how the unit operation is connected to make up a model of a complete flowsheet.

A side benefit of the “port” approach is that unit operation models might be run, for example for test purposes, independently of any host simulator. It just requires that a method is available for transmitting the required information to the ports.

6.3.2 Attributes

It is expected that the three major attributes common to all ports will be name, direction and type. These attributes are discussed in turn here:

6.3.2.1 Name

Each port has an identifying name, which, for example, for a distillation column may be “column feed”, “top product”, “bottom product”, “reflux ratio”, “pressure drop across tray”, and “energy flux to reboiler”.

The name attribute (“column feed” etc.) may be used by a simulator Graphical User Interface in defining the topology of the complete process. For example, we may have connected “reactor: outlet” to “column: feed”, or connected “orifice plate: pressure drop reading” to “PID controller: controlled variable input”.

6.3.2.2 Direction

The direction attribute defines whether a port delivers information to a unit operation model (input port) or receives information from a unit operation model (output port). It still has to be decided whether input and output are simply port attributes or whether they effectively define port sub-types. It has further to be decided whether some ports may both deliver and receive information. It should be noted that the port direction may not correspond to the physical direction of flow, which might be a separate attribute. For example, a unit operation model might compute a required make-up flow rate. The physical flow would then be into the unit operation, whilst the information flow would be out of it.

6.3.2.3 Type

In discussing port names above, it is clear that ports may transmit a number of distinct types of information. In the distillation column example given, we have material information (“column feed”), energy information (“energy flux to reboiler”) and information not readily classified further (“reflux ratio”, “pressure drop across tray”).

It is convenient to classify ports according to the type of information transmitted and, for this purpose, we introduce a “type” attribute. The three port types corresponding to the three classes of information are Material Port, Energy Port and Information Port.

Material Port. In defining Material Ports, one of the key decisions made in CAPE-OPEN is that Physical Properties Templates (see Chapter 8) will be associated with Material Ports. It follows that, at each port to which, or from which, a material flow is connected, not only will a Material Object for the connected flow be available, but also all the information necessary to create Material Objects within the unit operation model. For example, material objects modelling the liquid on each stage of a distillation column can be created having the same set of components as are present in the column feed and employing the same methods for computing physical properties. This consistency of approach between materials arising within unit operations and materials delivered through ports gives benefits in consistent access to properties. For example, a unit operation writer may issue requests such as “get viscosity of material entering through port 1” in virtually the same way that he can ask for the property of any other material in the unit operation model. This approach to Material Ports is consistent with the general proposals for modelling of matter described in Chapter 8. We gain great flexibility by accessing physical properties methods through ports because, no matter how a simulator executive actually associates physical property methods with materials, the port hides it and makes it appear as if it were associated with a stream. Thus, if a simulator actually only allows one physical properties package (or template), that same package will be accessed through every material port in the simulation. If the simulator associates physical properties packages with unit operation models, the same package will be accessed through every port to a given unit operation model. If there is a separate association with individual streams, separate packages may be accessed through separate ports. This port approach thus allows a wide variety of simulator types to be included within the same standard framework.

Energy Port. Energy Ports transmit energy fluxes. They define the type of energy transferred (heat, electrical, shaft work etc.). For specific energy types, further detailed information may be required. For example, in modelling heat flux, it may be required to know the temperature at which the heat is delivered, or if delivered over a temperature range, the whole cooling curve. Similarly, for electrical energy, we may need to know frequency, voltage etc. It follows that, in a similar way to that discussed for materials, it would be desirable to establish standard Energy Templates, with the ability to create corresponding Energy Objects. Energy Templates may then be associated with Energy Ports. In other cases, a single real number may be sufficient to characterize the energy flux. It will be noted that all material ports also deliver an energy flux corresponding to the energy content of the material. The Energy Port definition here is consistent with the Material Port in that the energy flux has a defined direction (which may, or may not, be the same as the direction of the information transmitted through the port).

Information Port. Information Ports transmit a wide variety of information not readily classified as Material or Energy. The information may be no more than a scalar number, where the standard may consist simply of specifying a floating point format. In other cases, the information may be much more complex. For example, it may be required to transmit an array of stage efficiencies, or a relationship for computing stage efficiencies.

7.0 Unit Operations Component

7.1 Introduction

This chapter describes the conceptual design of CAPE-OPEN standard interfaces for unit operations. These unit operation software components are the individual building blocks used to create models of the sequences and networks of individual chemical and physical operations that make up a manufacturing process.

The two main types of simulators are termed sequential modular (SM) or equation based (EB). Both types are widely used and the CAPE-OPEN standards will cover interfaces of both types. Commercial suppliers of specialized unit operation components can either provide single compliant unit operation components supporting both SM and EB simulators, or they may choose to provide specialized components which are dedicated to a particular environment.

A conceptual overview of a CAPE-OPEN compliant simulator is given in Chapter 5, and this abstraction underlies both the CAPE-OPEN overall design and the unit operation conceptual design. In particular, unit operations will communicate through interfaces with all three remaining principal components: the simulator executive, a physical properties component, and a numerical component.

7.2 Unit Operation Interface to Simulator Executive

The actions of the user during development of a simulation are an important determining influence on the interface between the unit operation and the simulator executive.

In general, these will include choosing flowsheet blocks, making connections, initializing selected unit operations, and specifying feeds and products. In particular, a flowsheet builder will use the ability of the simulation system to carry out a set of specific tasks including: *create unit operation; add unit operation to flowsheet; specify unit operation material connections; specify unit operation information connections; delete unit operation from flowsheet; delete unit operation; define unit operation report, edit unit operation; and save unit operation.*

In order to connect unit operations in simulation flowsheets, the CAPE-OPEN concept of a port is used (see Chapter 6.0). A port is defined as an interface that connects material, energy, and information streams from block to block. Within process simulation, a port converts stream information stored by the simulator executive to the form required by the unit operation or other simulation blocks. Ports are provided by unit operations as sockets to which the simulator executive can connect its streams.

Port interfaces provide the flexibility needed in order to simplify interfacing existing unit operation models. For example, CAPE-OPEN may provide ports that access both mole fractions and molar flowrates. The ports will cover polymers and other material types already identified by CAPE-OPEN as important, and they will also be extendible to accommodate additional types.

Ports are the mechanism for information exchange between the unit operation and the simulator executive, but do not in themselves define the data items to be transferred. This is the role of *global* and *public* variables.

Global variables are standard names associated particularly with material ports, for example those for temperature, pressure, flowrate and composition. Global variables will be defined in the CAPE-OPEN standard and will have the same meaning across all compliant simulators.

Public variables are defined by a unit operation or other flowsheet block and (during solution) can be exchanged via information ports. These variables may also be accessed directly by the simulator executive during configuration and reporting, as is discussed further below. These names will not be defined by the standards, but means of communicating them will be defined. Note that the meaning of ‘public variable’ in this context is distinct from that in software engineering.

Global and public variables may either be read/write, or read only; variables which may be set can be used to initialize a unit operation or may be reset during solution, for example by an optimizer. Read-only variables may be used in calculation blocks or during reporting.

7.3 Unit Operation Interfaces with Physical Properties Components

During the overall process of creating a simulation model, the user must define how materials behave during the simulation process. This is done by utilizing a physical properties software system that 1) provides a library of chemical species data and physical property calculation routines, and 2) provides a mechanism for selecting a required set of chemical species and calculation routines.

Unit Operations will have interfaces to a Physical Property Server (see Section 8.4.1) which allows the use of one or more physical properties software systems that support the need of unit operations for material behaviour models.

Design of these interfaces is the primary focus of the thermodynamics and physical properties activity (see Chapter 8) but will involve close cooperation and collaboration of the Unit Operation activity.

7.4 Unit Operation Interfaces with Numerical Components

As described previously, flowsheet building includes the steps of choosing flowsheet blocks: making connections, initializing the selected unit operations, and specifying feeds and products. A flowsheet manager handles this overall process which applies to both sequential modular (SM) and equation based (EB) simulators.

Prior to actual solution, both SM and EB simulators typically require a pre-solving phase to analyse aspects of the problem. In SM simulation this will include identification of recycles and the determination of appropriate tear streams and corresponding calculation sequence. In EB simulation this might include structural analysis of the Jacobian matrix and possibly block triangularisation.

After pre-solving, estimated values are set for the unknown variables and the unit operation solving sequence is started. In successive iterations new values are set based on the preceding calculated values. This updating of values is called promotion and different methods are used. A flowsheet is solved when the residual difference is less than the user defined convergence specification.

More complex activities involving unit operations and numerical components include infeasible path optimization. With this technique, recycle balances are included as equality constraints within an optimization. This approach is often faster than solving recycles for each step of an optimization.

A third activity is dynamic simulation. Here both variable values and rates of change are exchanged between the unit operations and numerical component.

Design of these interfaces is primarily a responsibility of the numerical activity (see Chapter 9) but will involve close cooperation and collaboration of the Unit Operations activity.

7.5 Unit Operation Configuration

This activity includes addition of a unit operation to a flowsheet by the user and subsequent data entry by the user, for example numbers of inlets and outlets, internal configuration options (with/without reboiler etc., numbers of trays, tray types) and numerical parameters.

This is a complex area with scope for varying degrees of integration between the simulator executive and the unit operation. Supported levels of integration may eventually include (in approximate order of complexity):

Basic batch: here each instance of a unit operation retrieves data from a data file, independently of the simulator executive.

Basic: as basic batch, but the simulator executive interrogates each instance of a unit operation for its connectivity requirements to enable it to be drawn correctly in the simulator's GUI.

Intermediate: the unit operation provides its own user interface functionality (e.g. forms for data entry and editing) which can be invoked from the host simulator's user interface. Data storage for the unit operation is managed by the host simulator.

Full batch: as basic batch, but data storage and retrieval for the unit operation is managed by the host simulator.

Full: the host simulator user interface displays native style forms for unit operation data entry based on additional configuration information provided by the unit operation.

Levels intermediate and upward assume at least a basic persistence standard as discussed below. It is anticipated that the Unit Operation work package will define and test standards to support at least basic/batch and intermediate levels.

7.6 Reporting by Unit Operations

Global variables such as stream compositions, pressures and temperatures will typically be automatically reported by the host simulator's reporting facilities.

Tailored reports may be created by reporting routines supplied with the unit operation. Other than the ability to invoke the generation of these reports, they are not otherwise covered by CAPE-OPEN standards.

The host simulator may optionally be able to include unit operation public variables in custom reports developed by the user.

7.7 Persistence and Archiving by Unit Operations

In a CAPE-OPEN context, persistence refers to preserving the state of a software component beyond the current computation. It usually means taking information from main memory (RAM) and storing it in long-term memory (disk, tape etc.) so that it can be retrieved later.

There are a number of persistence mechanisms available. One simple implementation of persistence may use a neutral format (implemented using ASCII files) as the stored form. By contrast, it may be highly desirable to have more complex persistence implementations (such as that involved in storing a Word document that contains both an embedded Excel spreadsheet and an embedded Visio drawing).

In process simulation, different levels or methods of persistence are used for different reasons. Such varying motivations or reasons might include providing fast restart facility or providing archival storage in order to reproduce a simulation calculation many years later.

The conceptual design for Unit Operations specifies that unit operations will provide simple, standard persistence methods, e.g. Save and Restore. If more complex arrangements are required in a particular simulator, the responsibility will lie with the simulator vendor and the unit operation provider to achieve them using standard CAPE-OPEN methods (see 8.6 Simulation Databank Interface).

7.8 Unit Operation Prototype Implementation

Prototypes of unit operations and other CAPE-OPEN components will be used in order to test proposed interfaces. These prototypes do not need to represent any advanced simulation capability beyond that of working within a component based simulation architecture. This means that they can be simple implementations created with a rapid prototyping tool, or they can be a wrapped version of public domain or legacy code provided by any of the CAPE-OPEN partners.

The first unit operation prototypes developed and tested will be simple blocks, including mixer, splitter, combination mixer-splitter, and flash. The latter will be used to test initial versions of interfaces to physical property and numerical components.

Prototype unit operations will be tested in a prototype test-bed capability. The test-bed capability will utilize both CORBA and COM/DCOM middleware technology. Unit operation prototypes will implement component interfaces using both middleware technologies (see 10.0 Validation and 11.0 PATH Conceptual Design).

8.0 Thermodynamics and Physical Properties System Component and Materials Concept

8.1 Introduction

Thermodynamics and physical properties are among the most frequent calculations carried out in process simulators and it is crucial to consider them in the context of CAPE-OPEN.

These properties are required for most real process modelling problems and also needed in their own right. It is possible to code physical properties directly into individual process models or write them locally whenever required. It is clearly advantageous to avoid this through a common system that has become the standard in most process simulators today.

The accuracy of a simulation is often extremely dependent on the thermodynamics models. Once developed, models are necessary for simulation in the whole life cycle of a process. Standardized interfaces will enable integration of those models into all appropriate software.

8.2 Units of Measurement

All interfaces should be restricted to SI units only. If an application wants to use other units, it is the responsibility of that application to take care of the conversion. This decision has implications on the other work packages and has to be confirmed by all partners.

8.3 Naming Conventions

In order that component parts of CAPE-OPEN compliant systems may communicate with each other, it will be necessary to define a list with **standard names** for the **properties** and **state variables / global variables**. This list will be extensible in that new standard names may be added from time to time as the need arises. In addition, where there is a need, other names may be used locally for the purposes of communication with proprietary models.

The basic list is derived from the needs of "conventional" materials but needs to provide a base of common properties for other CAPE-OPEN material types such as electrolytes, polymers and petroleum assays.

Ideally, the list should be based on a already existing, commonly accepted list, such as IK-CAPE, DIPPR or pdXi/STEP. However, it is possible that none of these lists will be sufficiently expressive for the purposes of CAPE-OPEN, in which case the list will have to be created within the project. The list will **have to be updated** from time to time to guarantee consistency. It would be best if after CAPE-OPEN has ended this could be done by a widely accepted institution (IUPAC, for example). If the list is not maintained, there will be a proliferation of names for new properties/methods and inevitably name clashes in other cases.

8.3.1 Pure Component Properties

Included in the Pure Component Properties List are the pure component attributes such as the component name and CAS number. Different physical properties packages may have different names for components and users may well want to use informal names in their simulations. It will be necessary, however, to supply a recognizable component identifier for property calls etc. so it is perceived that there will be a need in every simulation for a global component list and translation table - either to a unique, internationally recognized identifier or to the name used in the selected physical properties system. The following sections describe the categories of properties which will be required.

8.3.1.1 Constant Properties

Constant properties of pure components are those properties which are not functions of state variables. In this sense they are truly constants, but it must be recognized that the accepted values are measured values and may change as measurement techniques improve. Where values such as Critical Temperature are used in fitting model parameters (see 8.1.2) to experimental data, the actual value used must be a model parameter and not rely on the pure component value being the same.

8.3.1.2 Model Dependent Properties

Model dependent properties of pure components are properties which are functions of state variables. The function is defined by the model and it may be solved directly or iteratively. For each component property for which a model is to be used, a set of constant parameters will be required. These may come from a variety of sources, e.g. from a parameter database or fitted to measured data from a database or supplied by a user.

Derivatives

Derivatives are built of the property name, a point with a D meaning "Derivative" and the name for the variable, e.g. `property.Dtemperature` is the derivative of property with respect to temperature. The possible variables for differentiation are the stack/global variables.

Names for state variables/global variables

Names for state variables, e.g. temperature, pressure etc., need to be agreed across all work packages.

8.3.2 Mixture Properties

Nearly all property names of pure components can be used for mixture properties as well, however it is convenient in some cases to have special names. Some new properties will be required, e.g. for bubble- and dew-point conditions. Note that calculation of dew- and bubble-points is not part of the Property Server but part of the Physical and Chemical Equilibrium Server.

8.3.3 Universal Constants

A function of the form

```
call get_universal_constant ("name",value)
```

which can get universal constants such as velocity of light, gas constant etc. is considered to be desirable although not essential to CAPE-OPEN. Again it should be noted that such values are accepted measured values and may be updated from time to time. It could be a function of CAPE-OPEN to declare and establish a maintenance process for "currently accepted values" (e.g. via IUPAC).

8.4 Thermodynamics and Physical Properties

The objectives are to define standard interfaces for thermodynamics and physical properties and to demonstrate their effectiveness through selected examples and prototypes. Using these interfaces, it should be possible to carry out simulations in different simulators using a thermodynamics package from any vendor. It should also be easy to integrate the user's own models.

Beginning with very simple examples such as vapour pressure and bubble-point calculation on the basis of activity coefficients, the vendors will write a prototype for their thermodynamics packages.

8.4.1 The Property Server

The task of the Physical Properties Server is to calculate physical properties. A special interface for every property is not desirable, as the list of interfaces would be impossible to maintain and very difficult to extend in a controlled manner. Instead, a single interface is preferred with information about the required property given by a parameter in the parameter list (**PropertyList**). A new property may thus be introduced without changing the structure of the interface.

The second piece of information needed to do the calculation is a description of the physical object (**PhysOb**) for which the calculation has to be made, e.g. a stream, a set of pure components, or a mixture.

The third piece of information will be the state (**State**) of the physical object (e.g. temperature, pressure, …). For property calculations (as opposed to phase equilibrium calculations), this will generally be temperature and pressure.

To perform the calculation, information for the selected Physical Package System (PPS) is required. PPSs generally have several methods to calculate each property, so a method must be chosen and the required parameter set must be specified. A set of properties with the chosen method and parameter set for each one form an object called an Option Set. At (or before) the start of any simulation, one or more Option Sets must be created for use within the simulation (see also 8.4.1.3). When a physical property calculation is requested, the PPS must be instructed as to which predefined "Option Set" (**PPS_OS**) is to be used.

The Property Server will only calculate properties of the physical object as given. If it is necessary to calculate the distribution of species between co-existing phases, the Equilibrium Server (see 8.4.2) must be used for this first. Property calculation requests will, therefore, generally need to specify the phase for which the request is made (e.g. liquid viscosity, gas viscosity). Rather than extend the list of properties with specific names (e.g. "viscosity_liquid"), the phase should be specified separately, via the State Object. The method of specifying the phase needs to be agreed upon across all work packages.

There should be one interface for pure components and one for mixtures. This is because pure component calculations will need to return one property value for each component, while for mixtures there will just be one property value for the whole mixture.

A calling sequence may look like

- for **pure component properties**:

```
call pure_prop  (PropertyList, PhysOb, State, PPS_OS,
                  values, ErFl)
```

- for **mixture properties:**

```
call mix_prop  (PropertyList, PhysOb, State, PPS_OS,
                values, ErFl)
```

(These are shown as FORTRAN calls for illustration only. It is only to show the information that has to be transferred to the Physical Properties Server. For example, the information about the PPS_OS could be transferred separately by a preceding call(s). If only one option set is used, this call could be made once early on and then nowhere again.)

where

`values`	returns the calculated property/-ies
`ErFl`	returns any run time Error Flags, e.g. "Bounds Errors". Different components and/or properties may produce different messages

It is important to make sure that the severity of any error is easily recognized by the calling program so that, if necessary, calculations can be aborted or an iterative calculation can reassess its previous step. The text or actual failure code may be of secondary importance for the simulator - only relevant if required for reporting to the user (see also 8.4.1.2).

8.4.1.1 Pre-Calculation Check

Before attempting calculations for any property it should be possible to make a single call to test if the calculation is in principle possible for this property and for these components by the specified PPS_OS. This would check that the model is defined and all needed parameters are set, e.g.

```
      call prop_check (PropertyList, PhysOb, PPS_OS)
```

For each property from `PropertyList` it is checked whether the calculation for `PhysOb` by `PPS_OS` is possible or not. If it is not possible, an error message has to be made. Details are left to the implementation. In this case there is no need to have two different interfaces for pure and mixture properties.

8.4.1.2 Post-Calculation Check

After convergence has been reached and all calculations are done, it should be possible to make a call to check if the results are valid and not out of bounds, e.g.

```
      call prop_valid (Property, PhysOb, State, PPS_OS, ReFl)
```

Such a function would have to be called for the extreme values of each of the state variables, namely upper/lower temperature/pressure/concentration. The meaning of `ReFl` (i.e. "Reliability Flag") may depend on the Property Package used. It should at least give some indication as to whether a calculation is valid or invalid, but it could also give more informative messages like "slightly extrapolated" or "unconfidently extrapolated" and so on. These textual forms should make sense in an overview of the calculated flowsheet.

8.4.1.3 Configuration Interface

The required PPS_OSs have to be set up before the simulator starts to calculate. This is done via an interface called the Configuration Interface. The actual set-up procedure cannot be part of CAPE-OPEN standardization because it depends too much on the different Property Packages. Knowledge of the selected Property Package will be essential for using it.

The configuration interface may be accessible from within the simulator if the PPS is known to the host, otherwise it will be a stand-alone application which is part of the PPS package. In either case, it must be possible for the PPS to communicate to the simulator which PPS_OSs have been set up (e.g. via a Physical Properties Data file) and for the user to tell the simulator to attach a specific PPS_OS to each object in the simulation. Clearly, the calculation engine of the PPS also needs access to the details of each PPS_OS, and in the case of an external PPS this would be achieved by a single initialisation call before the calculation starts.

8.4.2 Physical and Chemical Equilibrium Server

The Physical and Chemical Equilibrium (PCE) Server is part of the THERMO Server and not just a unit operation. It is, of course, possible to have a unit operation to do essentially the same task either by direct use of the PCE server or by an iterative calculation making calls to the Properties Server.

The task of the PCE server is to determine the physical and chemical equilibrium given the concentrations of the total mixture and, for example, any two state variables which are sufficient for the task.

The server will determine the number of phases, phase fractions and concentrations with a call of the general form:

```
call PhysChemEqu (PhysObIn, State, PPS_OS, PhysObOutList,
                  PropList, PropValues, ErFl)
```

`PhysObIn`	Number of components, ComponentIds, concentrations
`State on input`	Contains two state variables out of a possible set (Temperature, Pressure, Enthalpy, Volume, vapour fraction...)
`State on output`	At least Temperature and Pressure
`PPS_OS`	Information about physical and chemical equilibrium for each phase (see below)
`PropList`	List of properties, which are to be calculated additionally
`PhysObOutList`	Number of phases, concentrations and amounts (fractions) of each phase
`PropValues`	Values of additionally calculated properties
`ErFl`	Run time Error Flag

PPS_OS should contain information about chemical equilibrium for all phases, rather than each phase, even each liquid phase, having its own PPS_OS. This allows for the easier specification of mutually compatible calculation methods for each phase and is more compatible with current simulator practice.

Because it is an Equilibrium server there is no need for it to handle kinetic reactions. That would be the function of a Unit Operation.

If only physical and no chemical equilibrium is to be calculated, then a call of the following form should be available:

```
call PhysEqu  (PhysObIn, State, PPS_OS, PhysObOutList,
               PropList, PropValues, ErFl)
```

Similarly for bubble- and dew-point calculations there could be special calls with the same parameter list, e.g.:

```
call BubbleTemperature (...)
```

This could simplify selection of special algorithms for these special case calculations.

8.5 Physical Properties Database Interface

CAPE-OPEN will use the most appropriate definition of a Neutral File. The current neutral file format used within IK-CAPE to link the DECHEMA database and AspenTech software is one example and the pdXi format is another that will form the starting points.

The contents of a Neutral File would be a mixture of measured data and parameters of models. The definition of standard models and functions is part of the scope, but it may be possible to do this in conjunction with the project SVPPM (Standardization and Validation of Physical Properties Models, see appendix C.3).

If all software conforms to CAPE-OPEN standards, it should be possible to use any physical property model within any Physical Property Package. If, however, it happens that a special model cannot be used with a particular Physical Property Package, it could be useful to be able to generate pseudo measurements from the special model and then use these to regress parameters for a suitable model in the package.

8.6 Simulation Databank Interface

The subtask has the aim to define a standard interface for the input data and results of a simulation. It should be possible to restart a simulation - possibly with a different simulator and years later!

A standard for simulation database is to be written. Both the Bayer Simulation Database interface and the pdXi interface will be considered to see their suitability in time for making the textual descriptions of the same.

8.7 Remaining Issues for Thermo

The issues remaining for Thermo include e.g. design of interfaces for solid, electrolyte, polymer and oil properties.

There is also an issue of whether the same models (at least standard models) in different packages using identical data should give the same results. It has been agreed that this issue is not central to CAPE-OPEN. However, for standard models (at least those for which no iteration is needed) in "normal cases" some operating companies consider it desirable that results should be "acceptably similar". By inference, the results should therefore also be "acceptably close to the correctly calculated results".

8.8 Modelling of Matter

8.8.1 Objectives and Introduction

Some model of matter is needed for at least two reasons. First, properties (thermodynamic and other) must to be able to be accessed from within the unit operation models. For this purpose, a sufficient model of matter is needed to ensure that we have unambiguously defined what is needed to recover a physical property. Secondly, “streams” will never be accessed directly, because they do not conform to any standard beyond that of the simulator we are using. The material concepts could be associated to "ports" of the unit operations. We need to be sure that our “port” transfers enough information to be able to access the stream properly. A matter model is able to ensure that we store consistent blocks of defining information.

Generally, the existing libraries, in order to calculate physical and thermodynamic properties and phase equilibria, focus on the calculation of the properties rather than on what is modelled (the matter). A consistent model of matter is missing and only appears through the thermodynamic and physical properties that are computed (such as enthalpy, fugacity or viscosity).

CAPE-OPEN has adopted an object-oriented approach, that is to design around the « What » aspect before defining the « How » aspect. Then the process of modelling involves:

- the identification of abstractions of the real world from a physical point of view (what software structures represent real-world material characteristics?).
- structuring these abstractions with the aim of being as close as possible to reality (what material description can be built up using descriptions of constituent parts of the material).

In this section, we only deal with the « What » aspect, not with the implementation aspects.

The matter is modelled in a structured and hierarchical way; the lowest element is the component and the highest element is the material universe. The UML Class Diagram is given with some explanations.

8.8.2 Structure

Our view of matter and its hierarchical structure appears on the following figures:

1. Hierarchical Structure of Material (Figure 8-1)
2. UML Material Class Diagram (Figure 8-2)

In the figures, the highlighted words are **classes**, with ***Abstract classes*** in italic. In the following explanatory text, Objects are underlined.

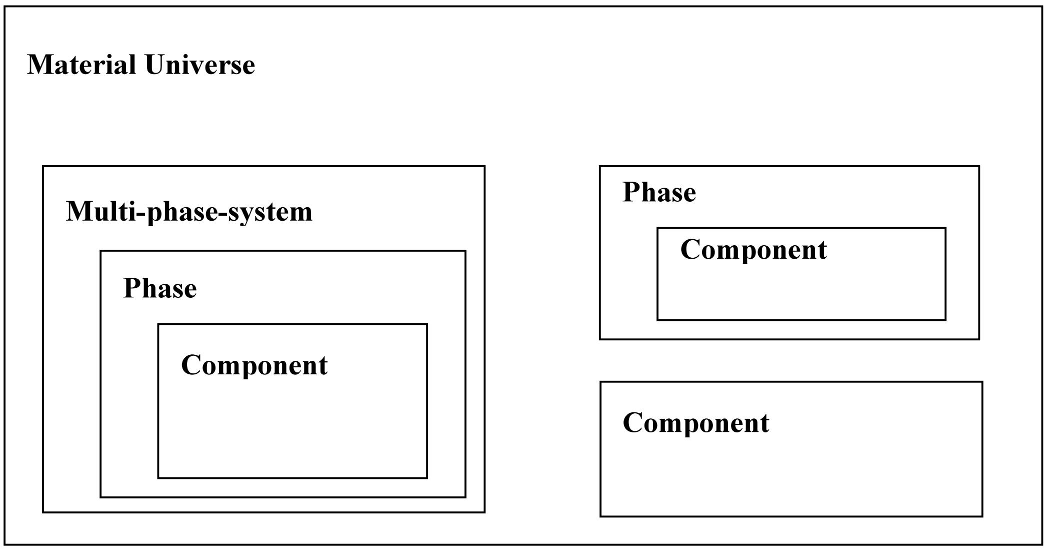


Figure 8-1: Hierarchical Structure of Material

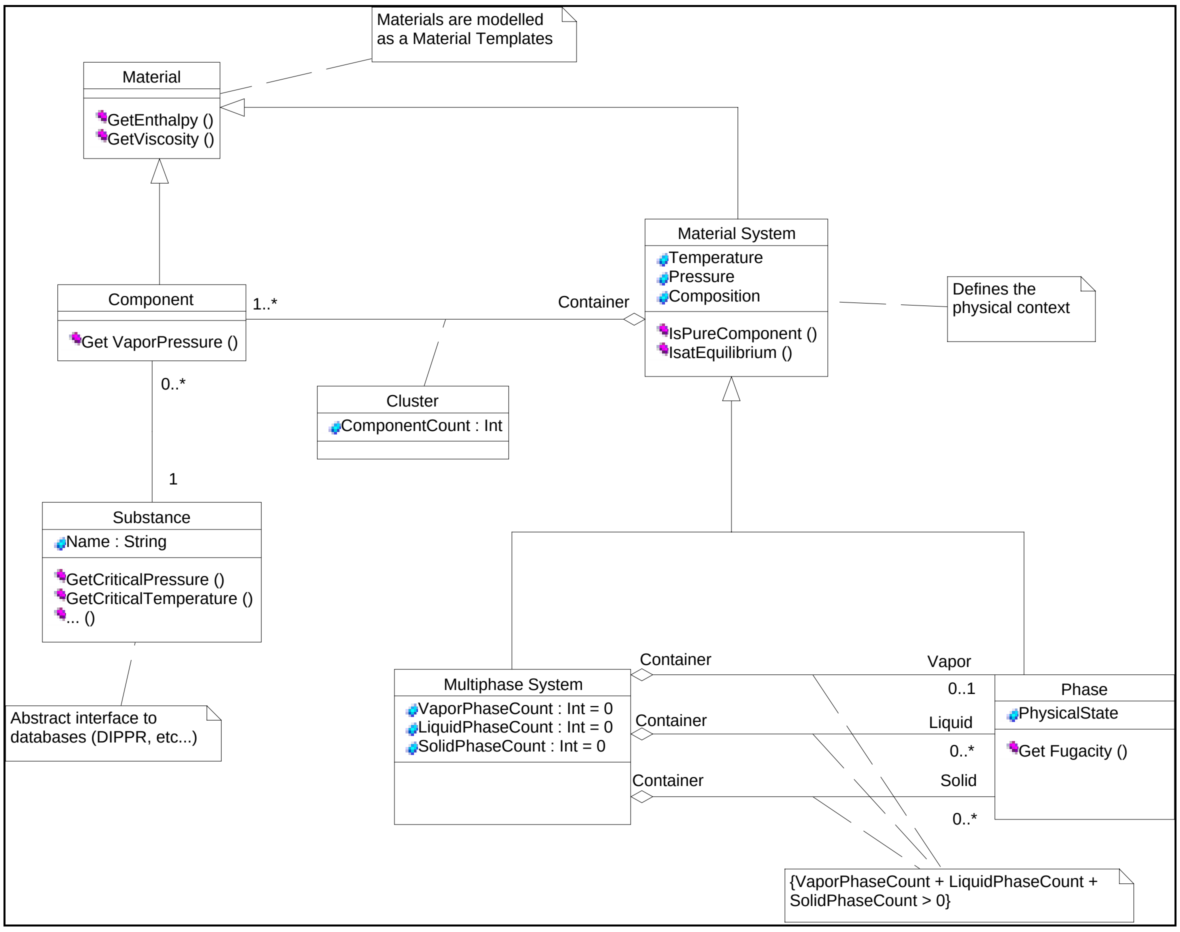


Figure 8-2: UML Material Class Diagram

Material is an abstract class, modelled as a **material template**. A material provides methods for calculating physical properties together with the data (physical/chemical constants and coefficients in correlation equations) required by the methods. **Component** and ***material system*** are specialized ***materials***.

Substance is an abstract class. It provides an abstract interface to the universal characteristics of a chemical species. These characteristics are physical/chemical constants and correlation parameters for a particular chemical species and are stored in databases such as DIPPR. There may be more than one substance representing the same chemical species, because it can have several databases. ***Substance*** provides data rather than calculating physical properties.

Component represents a chemical species. A component is related with one substance but there may be more than one component being related with the same substance.[1]

[1] You may want to model the same substance differently and not call the two corresponding components by the same name. Thus you might create two components "aqueous contaminant" and "cooling water" which, although both derived from the substance "water", may have different properties and may never mix.

Material system is divided into **multiphase system** and **phase**. A material system contains a cluster[2] of components and provides the physical context characterized by the intensive state (temperature, pressure, composition). It may or may not be enforced that a material system is at equilibrium (thermodynamic and mechanical).

Multiphase system is composed of liquid, vapour or solid phase(s). A multiphase system contains a cluster comprising all components occurring in the individual phases. A constraint prevents a multiphase system from being empty, i.e. containing no phase. It is not intended to represent systems with variable number of phases during simulation, e.g. a VL multiphase system that becomes a VLL multiphase system. The initial multiphase system must be defined with the maximal number of phases occurring during simulation. A VLL multiphase system will always have three phases structurally but can have a single phase physically.

Phase represents a homogeneous part, non-necessarily continuous (dispersed phase such as drops, bubbles, particles), of **multiphase system**. A phase contains a cluster of components occurring in that phase. The binary interaction parameters (e.g. NRTL, Uniquac or equations of state) are defined in the associated component cluster. A phase has a constant physical state during the simulation.

Material universe comprises all materials. An interpretation of this concept could be that it is used as a container that comprises all materials occurring in a simulation model.

The above gives a consistent general structure for modelling matter which may be refined or revised as a result of more detailed studies currently being undertaken using UML.

You might also want to create models to model some ill-determined component by using data for a well known component. For example, in a final distillation step, you might want to remove "heavy boilers" on whose composition you are not quite sure of. You get the data that you need to model "heavy boilers" by creating a substance "heavies" that has the same properties as "Xylene". In this case the substance "Xylene" is used to create two components "heavies" and (somewhere else in the process) "xylene". You can use the same idea to create several such components.

[2] A cluster is a simple set (or agglomeration) of components which is related to the mixture notion. The elements of a same cluster belong directly or indirectly to a same material context which locates them.

Each component of a cluster has an index which is local to this cluster. The properties calculations of material system have only access to components by means of a cluster, each cluster having its continuous indexing. Thus through several different clusters a same calculation (for instance a phase equilibrium calculation) can be carried out for several different contexts, especially for different numbers of components by phase. The cluster is then very useful for systems with uncondensables, non-volatiles, electrolytes, ...

9.0 Numerical Component

9.1 Introduction

The numerical work package is now divided into the three following subtasks:

- Solvers
- Streams
- Sequential Modular Specific Tools.

According to the CAPE-OPEN essential requirements, see Chapter 4, the following items, identified as essential requirements by a survey of all the partners in the project, have an impact on the numerical component.

The CAPE-OPEN project shall define standard model interfaces to enable the **interchange of the following elements of simulators**:

- Numerical Methods for solving dense sets of nonlinear algebraic equations (EL 8)
  (a standard component of steady-state sequential-modular and similar simulators)
- Numerical Methods for solving sparse sets of nonlinear algebraic equations (EL 9)
  (a standard component of steady-state equation-based systems)
- Numerical Methods for solving nonlinear differential and algebraic equations (EL 10)
  (a standard component of dynamic simulators)
- Numerical Methods for constrained and unconstrained nonlinear optimization (EL 11)
- Tools for sequential-modular simulators (EL 12)
  (Partitioning, Tearing and Calculation sequence to achieve torn set)
- Numerical Methods for solving large sets of linear equations (EL 13)
  (both dense and sparse solvers)

The general **types of simulators** that shall be catered for are (see Section 4.1.2)

- Sequential Modular Simulators, such as Aspen Plus and Pro II (TY 1)
- Equation-Based Simulators, such as SPEEDUP (TY 2)
- Sequential and Non-sequential Modular Simulators, such as HYSYS (TY 3), and
- Modular Hierarchical Simulators (TY 4).

9.2 Solvers

Several «Numerical Objects» were identified:

LAO: Linear Algebra Object

NLAO: Nonlinear Algebra Object

DAEO: Differential Algebraic Equation Object

OPTO: Optimization Object.

We will use also the following abbreviations:

EO: Equation Oriented

SM: Sequential Modular.

9.2.1 The CAPE-OPEN Linear Algebra Object

The function of the LAO is to provide a mechanism for the solution of full and sparse sets of linear systems of the form:

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}.$$

We will provide capabilities for square linear systems only: **A** is the square matrix of coefficients, **b** is the right-hand side, and **x** the unknown.

In the full case, the LA Object has to be provided with the dimension of the system and the values for (A, b).

In the sparse case, additional information on the pattern of nonzeros is needed. Patterns that occur frequently and have special methods of solution could be chosen among the following non exhaustive list:

- tridiagonal
- band diagonal with given bandwidth
- block diagonal
- block tridiagonal
- block triangular.

However, the most common case will be an unstructured sparse matrix, for which the positions of the nonzeros will need to be listed individually.

Once the LAO is initialized with the appropriate data, then it can be requested to solve the corresponding set of equations.

9.2.2 The CAPE-OPEN Nonlinear Algebra Object

The function of this object is to solve full or sparse nonlinear systems of equations.

Typically, numerical methods for solving full (respectively sparse) nonlinear systems of equations are needed by steady-state SM (respectively EO) simulators.

It should be possible to use residuals and derivative values provided by the client. The system should always be square.

In the case of a dynamic EO simulator, if any differential variables are present in the system, it will be assumed that their time derivative values are fixed at zero. The purpose of this is to make it simple to apply the NLAO to steady state initialisations. Specification of more general initial conditions for DAEs will require the variable value and its derivative to be presented to the NLAO as independent (algebraic) variables and also the appropriate initial conditions to be appended to the system as additional equations.

9.2.3 The CAPE-OPEN DAE Solver Object

This object will handle integration of mixed sets of differential and algebraic systems between discontinuities, reporting the results as it proceeds.

At present, no provision is made for handling of discontinuities. Obviously, this is an important issue that needs to be sorted out at a later stage.

As with the NLAO, the DAEO will handle only square systems of variables and equations. The DAEO should allow specification of initial conditions. In practice, this means that the DAEO will be able to set up a square system of algebraic equations grouping the initial conditions and the model equations:

$$C(x(0), \dot{x}(0), y(0)) = 0 \qquad \text{(initial conditions)}$$

$$f(x(0), \dot{x}(0), y(0)) = 0 \qquad \text{(model equations)}$$

and ask any NLAO to solve the corresponding nonlinear system.

The DAEO will then advance the solution of the model equations through time, until some caller-specified condition is reached or a model discontinuity occurs. It may ask a LAO to solve linear systems arising during this process, or it might operate at a higher level and make use of an NLAO at each step of the integration process.

Facilities for reporting the variable values at intermediate times will need to be incorporated.

9.2.4 The CAPE-OPEN Optimization Object

This object will handle the solution of optimization problems. In the Users Requirements, it has not been clearly stated if the OPTO should handle both steady state and dynamic problems. This object will likely handle the solution of steady state optimization problems only.

Therefore, it will make use of an NLAO to solve the underlying problem at each iteration. It will have to be provided with appropriate additional information, including identification of a single component of the solution as the objective function, and specification of the optimization parameters which it is free to set, together with their valid ranges.

User interaction will be needed to report the progress of the optimization.

9.3 Sequential Modular Specific Tools

In Sequential Modular (SM) simulators, there are at least two levels of calculations:

- the unit operation level: calculation of all the output stream values given values for all input streams and for all operating and equipment parameters
- the flowsheet level: calculation of all the torn stream values given values for all residuals on recycle streams and specification equations.

We are only concerned with the flowsheet level. The different tools include the following, classified in two levels within the executive:

- Pre-processor level (partitioning of the flowsheet, search of the calculation sequence)
- Partitioning/Ordering (Tool 1)
- Tearing (Tool 2)
- Sequencing (Tool 3)
- Processor level (unit operation calculations)
- Convergence module (Tool 4).

9.3.1 SM Tools (Pre-Processor Level)

We have a consensus on defining a toolbox grouping together Tool 1, 2 and 3 into a unique Sequential Modular Tool for the pre-processor level.

The main advantages of this alternative are:

- one common interface (Executive-SM Tool) is defined, so that the whole package (SM Tool for the pre-processor level) is interchangeable
- as partitioning/ordering and sequencing algorithms are trivial, they will always be integrated directly into the SM toolbox
- this choice gives developers more facilities in order to wrap the existing code or write the SM Tool (the three tools are combined in the code, just one interface to standardize).

It is important that, through the interface, both process quantitative information and process qualitative information can be transferred, allowing the host-program to provide raw information and the SM Toolbox to be responsible for all calculation sequence (algorithms, criteria, weight estimate, …). Thus, all process engineering know-how can be in the «intelligent» SM Tool package. Of course, for more flexibility, qualitative information should be optional.

9.3.2 Convergence Module (Processor Level)

The Convergence Module (Tool 4) is in charge of solving, for each Maximum Cyclic Network, the corresponding set of nonlinear algebraic equations. It plays the role of a specific solver, but the interface is closer to a unit operation interface than to a solver interface.

Like a unit operation, Tool 4 has to exchange information with the host-program; it may have its own solver or use an external solver according to the interface defined.

10.0 Validation

10.1 Introduction

The objective of the validation work package is stated in the introduction of Work Package 5 (Section 2.4.5) of the Project Programme for CAPE-OPEN:

"The ultimate technical goal of CAPE-OPEN is to be able to integrate components which have been developed independently and without previous knowledge of each other. The project thus needs a work package to prove and verify that the integration goal has been achieved. Only by this proof can it be assured that the required openness is achieved and adopted by the European CAPE community. Hence this work package covers the aspects of proof of concepts, assurance of feasibility, of performance, correctness and unambiguity of specifications."

Indeed, it is clear that no proposed standard interfaces can be put forward with any confidence if they have not been tested. The validation work package thus will ensure that the standards, as defined in the individual work packages, can be converted to working software that delivers the requirements specified by the users.

Two decisions have been made during the conceptual phase of the CAPE-OPEN project which strongly influence the work in the Validation work package:

- Use a modern Component Software Approach throughout the whole project.
- Commitment of the three vendors in the project to provide CAPE-OPEN conforming prototypes for both simulator executives and server components as specified in the individual work packages.

10.2 Validation Activity in CAPE-OPEN

10.2.1 Key Points in the Concept for the Validation Activity

During the conceptual phase of the CAPE-OPEN project, the following key points have been identified which have to be considered when setting up the validation activity:

1. The component software approach is already being adopted commercially and is an almost mandatory approach if we are to achieve our goal of freely swapping software written in different computer languages, using different compilers.
2. If we look at the component interfaces as "plugs" and the simulator executive interfaces as "sockets", we have to test that a variety of plugs fits into several sockets. This is the way we can assure that any vendor has a sufficiently clear specification of the standards to write both working plugs *and* sockets. To ensure that the socket side of the standard works, sockets from at least 3 independent sources must be validated to cover the breadth of philosophies used in simulator executive programs. Similarly, independent sources for the plugs have to be validated.
3. Effort should not be devoted to validating sockets of a simulator which is not supplied commercially because this simulator will not be used by more than one participant.

4. The validation exercise should start before the key decisions on the standards have been finalized. Furthermore, the operating companies should play a role in validation so that, at project completion, there is an assurance of the applicability of the standards to their priority technical areas. It is important that all criteria specified for the standards are testable and that relevant tests are undertaken. Any untestable or untested features of the standard are unlikely to be adopted, nor are they likely to achieve their goals if attempts are subsequently made to use them. The validation exercise must, therefore, form an integral part of the standards development activity from the outset and must fully involve the operating companies and the simulator vendors.
5. The validation concept for CAPE-OPEN must follow the requirements expressed in the priorities list (see Chapter 4). The validation activities will especially be performed on the hardware platforms mentioned in this list.

10.2.2 Component Software Validation Tasks

For validation in CAPE-OPEN, a three–level approach will be used to ensure close integration into the whole standard-development task from the user-requirements to the final deliverables:

- Elementary testing: Test of each interface independently by using test harnesses.
- Integration testing: Test of replaceability of existing vendor components by CAPE-OPEN conforming components of the same vendor, performed on a whole process unit.
- Plug&Play testing: Test of interchangeability of software components developed by different authors (vendors, in-house code from operating companies).

Prototype components will be provided by some of the CAPE-OPEN partners as part of their deliverables.

Figure 10-1 shows the flow of tasks for the validation activity within CAPE-OPEN.

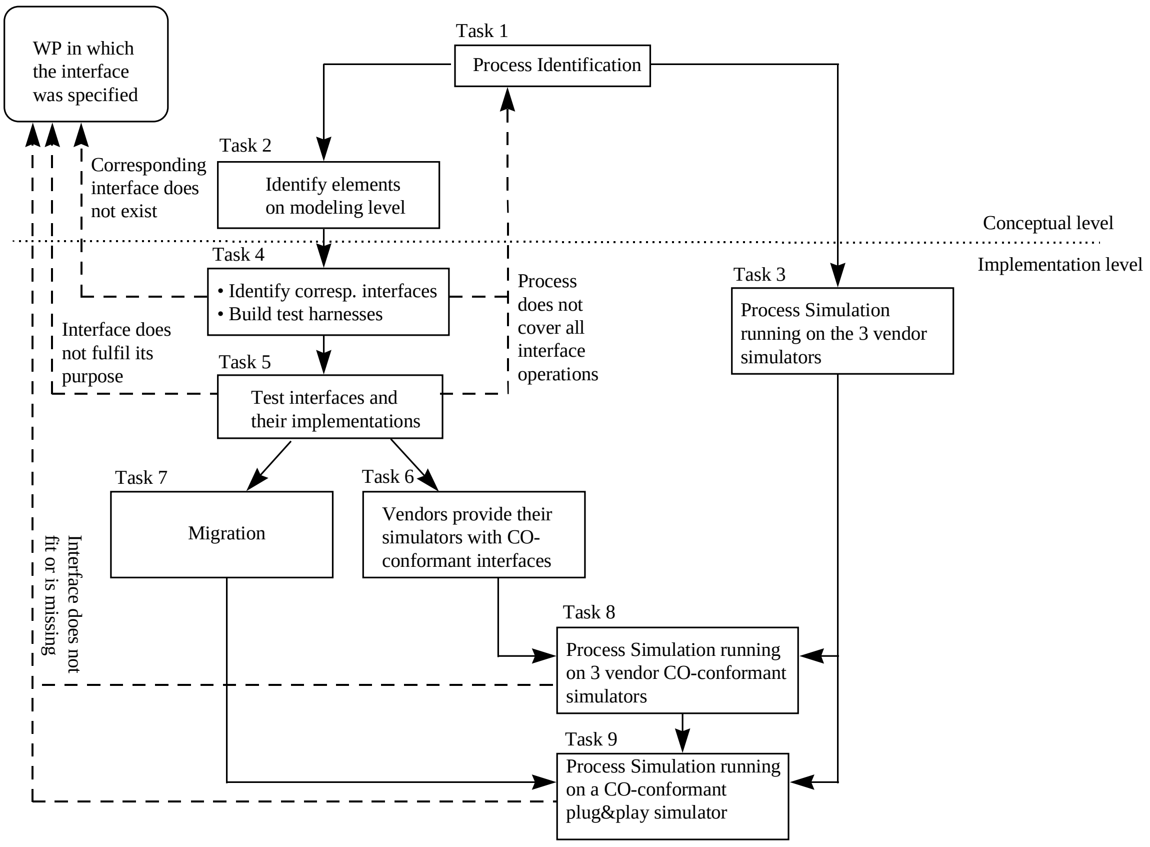

WP in which the interface was specified
Task 1
Process Identification
Task 2
Identify elements on modeling level
Corresponding interface does not exist
Conceptual level
Implementation level
Task 4
• Identify corresp. interfaces
• Build test harnesses
Process does not cover all interface operations
Task 3
Process Simulation running on the 3 vendor simulators
Interface does not fulfil its purpose
Task 5
Test interfaces and their implementations
Task 7
Migration
Task 6
Vendors provide their simulators with CO-conformant interfaces
Interface does not fit or is missing
Task 8
Process Simulation running on 3 vendor CO-conformant simulators
Task 9
Process Simulation running on a CO-conformant plug&play simulator

Figure 10-1: The validation tasks and their dependencies

The following tasks are identified:

Task 1 Following the users requirements, identify a number of complete processes that incorporate the essential characteristics identified. These characteristics are listed in Chapter 4. The process identification will be a selection of already existing process simulations (running on at least one vendor simulator).

The test problems must include "control" and "optimization" studies which both need standard access to "non-stream" information. Whilst there has been much discussion of stream standards, methods of accessing non-stream information are more diverse and relevant standards will need rigorous testing.

The next two tasks will be handled in parallel:

Task 2 Identify elements on modelling level within each process that can be tested individually before integration tests (including numerical and physical properties). This task may reveal essential work not currently allocated to a work package (for example work relating to interfaces with the simulator executive). If such work is identified, it will be allocated to the most relevant work package.

Task 3 handles the processes identified in Task 1. It takes the processes to all of the vendor products, thereby providing a means to compare the results of simulations carried out on separate simulators and establishes a basis for comparing the systems before and after CAPE-OPEN interfaces have been applied.

Effort may be needed to establish appropriate data sets. Nevertheless, it should be pointed out that the project is not a project in simulation itself. Thus, in principle, it does not matter if the data is wrong, or the simulation results are in error; it only matters that the results are plausible. By "plausible", we mean that, if a polymer expert looked at our polymer simulation, he should not be able to say that the simulation bears so little relation to what needs to be done in industry that any open interface probably has little industrial relevance.

Each of the processes must be set up and simulated on at least two simulators. The results should be compared; they are not expected to be identical but should be similar. The results for each will be verified using native vendor interfaces.

Task 4a concerns the identification of interfaces that correspond to the modelling elements that have been identified during Task 2. The identification will be done through the definition of the UML "Use Cases".

The identified interfaces must be part of the set of interfaces that will be delivered by the work packages. Thus missing interfaces indicate that something is missing in the corresponding work package. If the work packages are not able to deliver the required interface specifications at this time, the testing has to be delayed. On the other hand, interfaces produced in a work package but not identified in the current task may indicate that the UML "Use Cases" are not complete (e.g. Physical Properties Database or Simulation Database could have been forgotten).

Task 4b will establish simple test environments, or test harnesses, to allow the testing of interfaces and component implementations. These will be a series of small stand-alone programs that can accommodate the various components that need to be tested. In the design phase for these test harnesses the PATH work package will provide its experience from the set-up of the simulator workbenches (see Chapter 11). The harnesses will set out to mimic the range of simulator types found in practice, including for both modular and equation-oriented simulators:

- those with preset direction of computation as well as those allowing alternative directions of computation; the test harness will also run through the alternatives permitted
- dynamic as well as steady-state (including different approaches used)
- those employing no derivatives as well as those employing derivatives (allowing different combinations of derivatives)
- solution of linear and nonlinear algebraic equations
- solution of differential-algebraic equations (note that, since these are not integration tests, the equation solver interface cannot be tested together with the dynamic unit operation interface, except when the coding for so doing turns out to be trivial)
- if it is decided to undertake the task, interfaces to partitioning and tearing algorithms (Remark: In the most recent User Requirements review, this task was identified as "very desirable", but not as "essential".)
- interfaces to VLE calculations, to LLE and VLLE calculations, to particulate system calculations, to polymer system calculations.

These tasks will not be undertaken simultaneously, thus some interfaces will be established in the first month or so, and some may not be started until near the end of the work package. The tested components and test harnesses will be delivered to CAPE-OPEN partners to assist them in developing their own conforming components.

The test harnesses may reveal deficient functionality, and any such deficiencies will be reported to CAPE-OPEN partners to allow revisions to be developed.

Task 5 will be to test the interfaces and the various components with the harnesses developed in Task 4, thereby recovering design flaws or showing the workability of single interfaces. The tests will be directly derived from the UML „Scenarios“. The components to be tested are provided by the individual work packages. Obviously the Validation tests need to be matched to the status of each prototype, i.e. when a prototype has only a limited functionality or is only made for a special hardware platform.

After having passed the elementary testing of interfaces, again two parallel tasks will be carried out. Both of them concern existing simulators and their adaptation to CAPE-OPEN conformance.

Task 6 will handle the three vendor’s products. For this task the vendors adapt their simulator executives by providing prototype open interfaces. They will as well extend the prototypes delivered in the individual work packages by, for example, including CAPE-OPEN standard Thermo calls within a CAPE-OPEN standard unit operation model. They will be guided by the draft standards published by CAPE-OPEN, the test harnesses provided in Task 4 and the set of example problems provided under Task 1.

The vendors will disclose documentation on their work in migrating from current proprietary standards to CAPE-OPEN standards to the extent that proprietary information is not compromised.

Task 7 will handle migration from current traditional architectures, especially in-house legacy FORTRAN code, to component software. This migration is needed to preserve operating company process expertise. The goal is to provide a description of the steps necessary to achieve successful migration. The IK–CAPE Thermodynamics package will serve as one starting–point for the migration investigations.

Task 8 integrates Task 6 and 3. It will run the processes identified in Task 1 on the three vendor simulators, after their adaptation to CAPE-OPEN conformance. The results of Task 8 will be compared to the results of Task 3 for each of the vendor products. It will be verified that for each vendor product, the result of using the CAPE-OPEN standard interfaces of the simulation of each of the test problem processes is the same as that obtained from the same vendor product using their native interfaces. The validation tests will be passed when results differ only minimally with respect to numerical values and runtime performance. Where a test fails, corrective action will be taken in revising the standards and/or the migration guidelines.

Task 9 will broaden the scope of Task 8 to testing of plug&play simulator behaviour by interchanging components between the commercial systems and with components delivered by operating companies. Task 9 clearly also further validates the migration guidelines developed in Task 7.

The tests will verify that the behaviour of components transferred to a new environment is essentially identical to that in their native environment. At the end of this exercise, the companies should have available simple guidelines which enable in-house components to be interfaced routinely with any commercial simulator.

During the execution of these tasks each flaw, mismatch and/or unexpected behaviour needs to be documented and passed to the work package group that it concerns, i.e. at least the group that specified the respective interface, so that corrections can be initiated as soon as possible. The dashed lines in Figure 10-1 show some of the reporting actions but not all of them..

Additional remarks:

None of the above activities relate directly to the specific type of simulator (sequential–modular, equation–based). Certainly the validation activities that need to be undertaken are the same for both types. Furthermore there are validation activities which result from the interaction of sequential–modular and equation–based components, for example:

1. Can equation–based simulators accept components written to the modular standards, specifically unit operation and physical properties components?
2. Can equation-based simulators export components to the modular standards?

10.3 Simulation Databank Validation Tasks

The tasks in Figure 10-1 establish that components can be interchanged between systems. There is a substantial, and in some ways more extensive, test programme to verify the simulation databank interface (see Section 8.6). It is not possible to define this test programme fully until the technical approach is established by the responsible work package. Nevertheless, there are already first ideas on how a testing schedule could look like. We illustrate it here for the case that the use of simulation files would have been decided: In this case simulation files would be written by one program and read by another. The proposed testing schedule is:

1. Test that topology (without controllers etc.) can be written and read by the same system and by a different system.
2. Test that topology (without controllers etc.) plus inlet flow rates, initial stream estimates etc. can be written and read successfully between the same and different systems.
3. Test that topology plus relevant unit operation and physical property data can be written and read successfully between the same and different systems, when the components (unit operation models etc.) are the same in the two systems. (That is the same set of components have been interfaced to both systems).
4. As 2. but including controllers and optimizers. This combination tests that instructions to access information such as stage temperatures are successfully coded as are instructions to set data values.
5. As 3. but where the components are not the same between systems; for example a component is missing in the receiving system (for instance no liquid/liquid extractor) or there is no exactly equivalent component (for instance only a multi-stage distillation column in the receiving system where the donor had a packed column).

Such tests should not cause failures but should make clear where human intervention is needed to supply missing data. With regard to superfluous data, decisions have to be made as to whether the receiving system automatically rejects such data; it should certainly be stored in case that later in a project life cycle another receiving system can make use of it.

6. A more exact test than 4., in which the stored file may specify connections to or from points that do not exist in the receiving system.

11.0 PATH Conceptual Design

11.1 Introduction

The main motivation of this work package is to provide a window to advancing simulation and software technology in order to ensure that work done in CAPE-OPEN is future-proof. The work package addresses both conceptual and technical issues. In the conceptual area, there is an opportunity to assess the effectiveness of alternative conceptual designs that may be impossible to pursue in the other CAPE-OPEN activities.

PATH will provide frequent demonstrations and presentations of results for the rest of the project. These will serve as a basis for discussing further work in PATH and for the overall project. A key role is to address questions which arise during the project and which cannot be handled by other activities in CAPE-OPEN which have more focussed deliverables.

Where appropriate and relevant to the CAPE-OPEN standards work, the participating simulator vendors are keeping PATH abreast of their work in progress so that we have the maximal benefit of their collective experience in the standards area and can avoid duplicating work that is already being done.

Areas of work discussed below are 1) conceptual issues in simulation modelling, 2) software technology, 3) prototyping activity, and 4) a CAPE-OPEN repository.

11.2 Conceptual Issues in Simulation Modelling

Work in this area is aimed at defining the future state of chemical process simulation environments. The bulk of the work done by other CAPE-OPEN work packages will provide ways to take existing technology and enable it to work more effectively, including the ability to exchange software components. However, the way in which this is done should anticipate how simulator technology will advance in the longer term. Providing such a long-term perspective will lay a solid groundwork for future technology advances. This perspective will be achieved by 1) defining the future simulator executive in a *top-down manner* using object-oriented design methods, and 2) by not being constrained by existing systems.

As described, there are many issues and conceptual ideas that cannot be resolved in the mainstream of CAPE-OPEN. Some ideas might be omitted due to resource limitations even though they are regarded as important (e.g. evaluation of different stream concepts). To the maximal extent possible, these ideas will be addressed and evaluated using a prototype workbench capability provided by this work package. This support work complements conceptual design work done by PATH.

Since unit operations are the crucial blocks that make up a simulation, there will be an initial focus on unit operation design and related architectural issues. A starting point is the assumption of an active unit operation object that provides three main types of functionality. These are 1) runtime functionality during simulation, 2) functionality for set-up or initialisation (including extension and modification of the base unit operation object), and 3) reporting a unit operations computational state when its solving status is any one of ready/waiting, completed or interrupted.

From an architectural viewpoint this raises a number of important considerations. PATH will study and evaluate the following questions and issues in depth.

1. A systematic OO based approach to the unit operation object design requires some degree of model structuring. In particular, the unit operation can be divided into phases and phase

connections. Any of these structural entities are then described by a set of balance equations which can be refined hierarchically. Another particular example is reaction kinetics. Allowing unit operation components to be hierarchically decomposed facilitates customisation by the end user and makes it possible to reuse fine grained sub-components.

2. Is it better to have active unit operation objects combining runtime as well as definition functions or is it better to have a separate model server component to build/modify the model that subsequently may be integrated into some other CAPE components (like the unit operation component)?
3. Is it desirable to build a single unit operation object for a particular processing task, which can then be used in multiple contexts, such as various simulator executives (sequential-modular, simultaneous-modular or equation-oriented) and various modelling application areas (steady state or dynamic simulation, design, optimization, parameter estimation, data reconciliation)? An alternative is to build unit operation objects that are based on deriving different unit operation classes from a base or super unit operation class in order to improve runtime efficiency and maintainability for the various application contexts. Experimentation will create an understanding of how this potentially impacts the CAPE-OPEN interface standards.

Eventually, the relative merits of these approaches to unit operation design will be judged by their value to industry, but there is an expectation that important advantages can be obtained in the longer term by utilizing a more systematic approach. For that reason, it will be important to evaluate the area and define the potential impact on the CAPE-OPEN standards. The conjecture is that more versatile components are possible without incurring a large increase in computational overhead.

Summarizing for this area, an important task is to do *conceptual research work* on future simulator architectures. The goal is to contribute to a better understanding of the design of component-based simulator architectures. This will allow CAPE-OPEN to anticipate long range developments in simulator technology in order to provide for future compatibility with CAPE-OPEN standards. Work done will place heavy emphasis on experimental work using a prototype workbench in order to resolve these questions. It will also help address questions and issues that arise within the main project.

11.3 Distributed Technology and Multiprocessing/Concurrent Computing

Work in this area comprises three parts. The first part focuses on existing technologies (i.e. middleware, and Internet or intranets) that build the interaction basis for the component simulator approach. The second part concentrates on new software concepts and developments that can improve simulation technology. The third part addresses the area of multiprocessing.

Middleware and Internet

The goal for this first part is to ensure effective inter-working of different middleware and also to gain experience with Internet distributed component applications. In particular work will be done on:

- Platform Interoperability: Distributed platforms like Orbix and OLE/DCOM will be used within the project. Tools such as OrbixDesktop will be tested in order to ensure interoperability of the two distributed platforms. This will allow exploitation of software components from each of the two platforms, thereby guaranteeing openness of the simulator from a software technology perspective.

- Internet Computing: Extending distributed computing to the Internet, using ActiveX and Java based technology, allows for the utilization of the CO-Repository (discussed later) as a component library. Other aspects of distributed computing will be covered, e.g. delegation of computationally intensive jobs to remote sites that offer appropriate computational power.

New Simulation Software Concepts

The second part of this work area is aimed at developing new concepts and software in order to help simulator users to configure a “best of breed“ simulator for a given problem. Major questions to explore include:

- *Automated Model Building:* To what extent can a simulator executive help a simulation model developer or model user to solve a specific problem, for example:
- *Software Component Selection:* The goal is to select optimal components for a given problem domain (e.g. the simulator may select component interface implementations that are optimal with respect to numerical stability, performance criteria,...). Can component selection be automated with respect to the component configuration for a given problem?
- *Component Set-Up:* The goal is to optimize component set-up based on the specific hardware and the current workload of the simulation environment (e.g. utilize idle machines, set up components on multi-processor machines when components can utilize multiple processors, and assign highly interactive components in a way that favours efficient computation). Can this be effectively automated?
- *Component Granularity:* What are the costs of having fine/coarse grained components with regard to providing assistance for component selection and component set-up? What is the impact of different levels of granularity on code-overhead, stability, managing of component based simulation applications and performance? What is the effort needed to build representative simulation models from simulation components that have different degrees of granularity? On one hand, overhead in terms of code and computational resources increases as component granularity becomes finer. On the other hand, coarse grained components are less flexible with regard to composing “best of breed“ applications.

To reach the above goals, we will do conceptual work on software environments that concentrates on the specific chemical engineering aspects. This will ensure sound development and exploration that address issues that have been identified.

The programme of work will also cover the practical topics for software technology used to support component architectures (sometimes termed middleware). Work will focus on the leading relevant technologies of OMG’s middleware standard, denoted as CORBA, and the Microsoft standards of COM/DCOM and its various ActiveX implementations. Rapid prototyping tools will be used as much as possible. There will be a heavy focus on distributed and Internet computing, management of components (including versioning issues), and operational issues of error handling in a component based simulation environment.

Multiprocessing and Concurrent Computing

The third part of this work area will briefly assess coarse grain parallel algorithms that can be used in modelling and simulation (for 2-16 processors). Experimental work will be done on using various forms of ActiveX technology in a true concurrent computing environment. This work will be done because multi-threaded and concurrent computing (supported by multiprocessors) are already moving into the mainstream. Research is needed in order to provide understanding of the related future conceptual issues for the body of CAPE-OPEN work.

Initial work will investigate how to parallelise sequential simulator code by:

- identifying sets of tasks in a program's structure that allow the tasks of one set to be executed in parallel with other sets (macro-parallelism), and by
- identifying tasks that may be parallelised themselves (micro-parallelism).

Issues to examine include synchronisation, error handling, robustness and reliability. Experimental work with the prototype testbed will address these issues.

11.4 Prototyping for Exploration and Technology Assessments

Some minimal prototype implementation is needed in order to test the effectiveness of component-based simulation architectures. Much of the testing will focus on design of component interfaces, but testing will also be done to support conceptual and technical work carried out by the work packages.

Therefore, PATH will build a simple component-based test environment that provides a basic level of simulator functionality. The test environment will take advantage of existing code provided by CAPE-OPEN partners or code that is publicly available. The aim is not to create a new simulator, but to establish a workbench that can be used to explore new conceptual ideas and new technology. This work will be coordinated with work done for validation issues in order to look for potential overlaps. For example, some common infrastructure might be used for implementation purposes.

The prototype only needs to contain enough unit operations to make realistic tests of component-based simulation. We estimate that the number of unit operations needed for useful testing of assemblies is in the range of 10-20 (compared with several hundred that are typically encountered in industrial practice). This means that a numeric component can use simple, well-known methods to solve flowsheets. The only need is for a reproducible environment that tests and validates the interactions of various component interfaces.

Starting with unit operations, the prototype testbed will be extended to include components for numerics and physical property calculations. A starting environment will include a simple framework, a small set of unit operations (e.g. mixer, splitters and flash), a solver (e.g. Newton-Raphson), and a simple physical property component (e.g. ideal mixture VLE). The framework will provide some minimal user interface functionality.

From a software standpoint, the workbench capability will encompass the two mainstream software standards that support the use of software components. These standards are represented by the OMG based CORBA and the Microsoft COM/DCOM based ActiveX technology. Rapid prototyping tools such as Microsoft's Visual Basic 5 and Visual J++ will be used. These tools and environments are increasingly used in the development of advanced computing applications.

11.5 CAPE-OPEN Repository

This work will create an Internet-based electronic repository that provides a unique resource of valuable CAPE-OPEN work results. The organization of the repository will utilize a powerful suite of analysis tools and methods which are designated UML (Unified Modelling Language). As described below, key UML information will be included in the repository.

UML provides different views of planned and proposed software systems. Use cases, that are a key part of UML, document the functional interactions between actors and a software system. An actor may be an external user of the system or some defined subsystem. After defining Use Cases, the analysis and design of class interfaces is done, and then test cases are generated.

The CAPE-OPEN repository will provide efficient information retrieval and browsing. It will be designed to store at least the following items:

- Prototype components which will be used for test cases. Additionally, a public library of components may be provided which can be used by everyone.
- UML analysis and design models (i.e. Use Cases, collaboration, sequence, component, and Class Diagrams) used to understand the intended behaviour of CAPE-OPEN component interfaces.
- Interface specifications in IDL (i.e. object, interface, and method description including a literal description of their functionality) as needed to build standard CAPE-OPEN compliant components.

Initially, most UML models and interface specifications will be prepared as Microsoft Word documents and Visio models. As we gain experience we will use UML automated tools so that consistency checking can be done. Anticipated future expansions of the CAPE-OPEN repository may include consistency checking among stored models and software components.

Appendix A1 Methods and Tools within CAPE-OPEN Project

A1.1 Main Technical Choices

The following technical decisions are supported by work done during the OO-CAPE and OS-CAPE projects and by individual achievements of many partners involved in the development of modern software for process simulation.

1. The standard interfaces should be defined and expressed using an **object-oriented** approach. The OO approach is currently the best technical solution for developing interface standards. It also encompasses the « conventional » procedural approach.
2. The standard interfaces assume that a process simulator is made of several **components**.
3. The standard interfaces should use existing **middleware**, namely ActiveX/COM and CORBA. More specifically, the standard interfaces should be expressed **in both forms** in order to be future-proof.
4. The standard interfaces should be applicable **to several hardware platforms and operating systems**.
5. There should be a distinction between **project work** and **the standard.** The life-span of the standard is expected to be much greater than the duration of the initial CAPE-OPEN project itself. Choices made for the project - because it has limited resources and duration - should not compromise the quality and scope of the standard.
6. The standard interfaces should allow the encapsulation of **legacy code**.

A1.2 Notation and Interface Definition Languages for the CAPE-OPEN Standard

A1.2.1 Notation

We adopt the Unified Modelling Language (UML) for the CAPE-OPEN set of object models. UML unifies the popular OMT, Booch and Jacobson methods for object-oriented projects, and it is becoming a de facto standard with high acceptance from the OMG. UML is seen as the way forward for CAPE-OPEN. Appendix A2 provides an introduction to the UML notation.

A1.2.2 Interface Specifications

We will express the interface specifications **both for ActiveX/COM and CORBA**. This means that the CAPE-OPEN interface specification documents are expected to contain two parts[3], one describing the ActiveX specification, one describing the CORBA specification. We will further reference the interface specifications as **CO/Ax** and **CO/CORBA.**

[3] In addition to the formal models in UML

The CO/Ax specifications will be expressed in MIDL, the Microsoft Interface Definitions Language. The CO/CORBA specifications will be expressed in IDL, OMG's Interface Definition Language. We will evaluate the **bridges between CO/Ax and CO/CORBA** as a part of the PATH work package.

A1.3 Methods and Tools for the CAPE-OPEN Project

We do not recommend a single modelling tool for developing the object models. The available tools are not mature enough. As a consequence, the UML models will be prepared with different tools (Rational Rose, Select, P+, Visio etc.) until we can recommend the use of a single one. As another consequence, the *CO/Ax* and *CO/CORBA* specifications will not systematically be automatically generated from the UML models. We will **reassess the tools** after OMG's acceptance of UML and when good UML tools will be available. We will, in the mean time, establish an **electronic repository** for all UML models developed in the project

A1.4 Work Process

The work process that we follow for each sub-task of the three CAPE-OPEN technical work packages expected to deliver standard interface specifications and prototypes is presented in Table A1.1.

Phase	**Step**	**Goal**	**Means**
ANALYSIS	Users requirements, text	Requirements in textual format	MS-Word
ANALYSIS	Users Requirements, Use Cases	Use Case models	Any drawing package or UML tool, Visio
DESIGN	Design Models	Sequence, Interface, Component Models using UML	Any drawing package or UML tool
DESIGN	UML Repository	Implement UML models in repository	Tool such as Microsoft Repository or Rose4.0
SPECS	CO/Ax Interface Specifications	Interface specifications in Microsoft's IDL	
SPECS	CO/CORBA Interface Specifications	Interface specifications in CORBA's IDL	
IMPLEMENT	CO/Ax Interface Implementation	PrototypeMIDL implementation	Visual C++, Visual Basic Encapsulated FORTRAN
IMPLEMENT	CO/CORBA Interface Implementation	Prototype IDL implementation	C++compiler, Encapsulated FORTRAN
TEST	Standalone Testing	Tested component	

Table A1.1: Work Process Phases

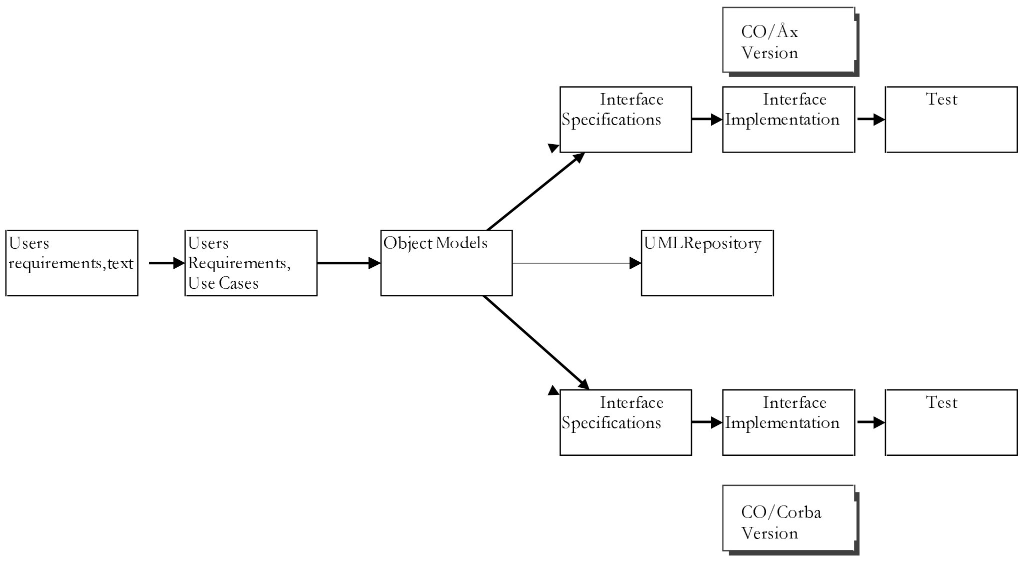


Figure A1.1: Summary of Work Process

A1.4.1 Analysis

The goal of the Analysis phase is to produce a structured set of users requirements in the form of a textual description and of a set of Use Cases.

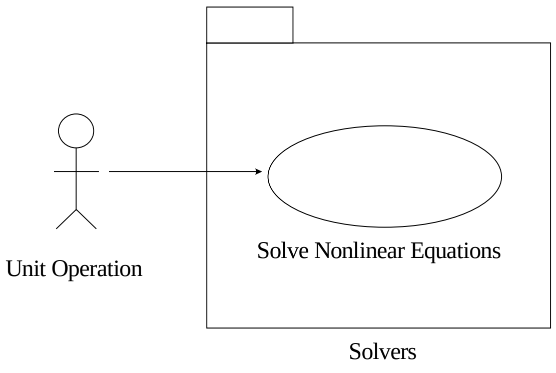
A1.4.2 Users Requirements, Text

Express the users requirements in written form. This should be obtained by consensus of the Work package team. No tools are used at this stage. The result is an MS-Word document which presents and justifies the requirements. For the CAPE-OPEN Unit Operation interface, an example could read as follows (UO means Unit Operation):

> … In this phase, the simulator evaluates each unit in turn. In the case of CAPE-OPEN UOs, it invokes the UO’s “calc” method. The UO in turn invokes methods of the simulator, thermo and numerics objects, as required, during the course of its calculation....The basic requirement is for the external UO to be able to return to the host simulator either “OK”, “failed”, or “caution”. The condition “caution” could apply if the external UO has an internal convergence loop that has not fully converged. This may not matter if, for example, the UO is in a flowsheet recycle loop and UO convergence is achieved by the time the flowsheet loop has converged. It would also be valuable to allow an option in the interface for an explanatory text string to be passed from the external UO to the host in the event of an error occurring...

A1.4.3 Users Requirements, Use Cases

Build Use Case Models from the Users Requirements, using the UML notation. The Use Case models express the core requirements and provide a basis for testing a proposed design. These models should be obtained by consensus of the Work package team. No tools are required (i.e. the drawing capacities of MS-Word suffice) although the use of UML diagramming tools is encouraged. The result is an MS-Word document including the models. The following example is a Use Case for a Unit Operation requiring to solve its nonlinear equations:



Use case description:

A Unit Operation requires that the Solvers package solve a set of nonlinear equations. When the solution is complete the Unit Operation checks if the solution has converged. If the solution has converged then the Unit Operation uses the solution. If the solution has not converged then the Unit Operation must handle the exception.

Exceptions: Solution may not have converged

Figure A1.2: Use Case Example

A1.4.4 Design

The goal of the Design phase is to produce a set of design models using the UML notation suitable for the requirements expressed in the Analysis phase.

A1.4.5 Design Models using UML

Build Sequence, Interface, Component models from the Analysis set of documents and following the UML notation. These models are essential for the design of interface objects and will be used in a later stage for preparing the interface specifications. No tools are required (i.e. the drawing capacities of MS-Word suffice) although the use of UML diagramming tools is encouraged. The result is an MS-Word document including the models. The following example extends the Use Case « Solve Nonlinear Equations ».

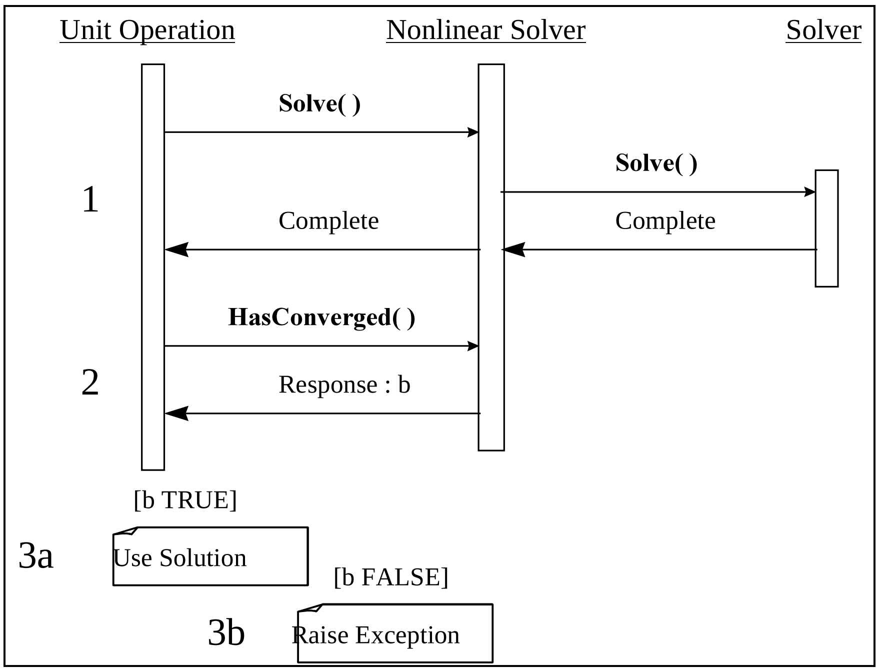


Figure A1.3: Sequence Model

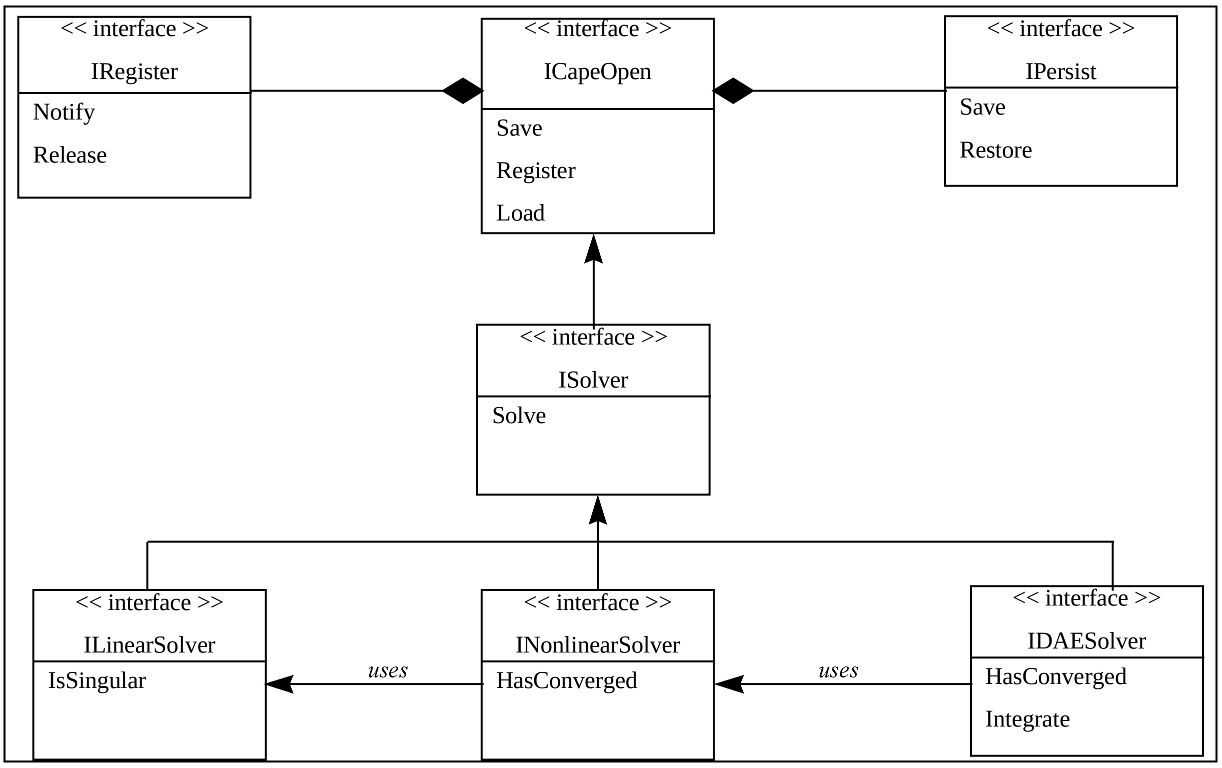


Figure A1.4:Interface Model

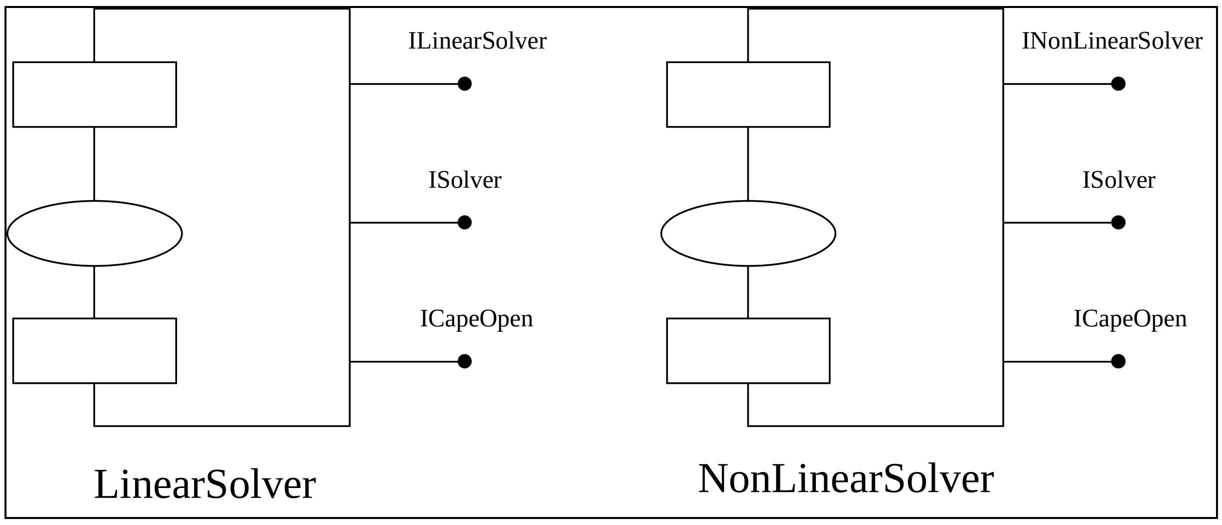


Figure A1.5 Component Model

A1.4.6 Implement UML Models in UML Repository

This phase is parallel to the mainstream activity of CAPE-OPEN. All UML models are implemented in an electronic repository which serves several purposes:

1. supply a single repository for all analysis and design models accessible to everyone;
2. prove that the UML model is consistent by introducing it in an environment with consistency checking capacities;
3. offer hands-on experience with UML tools;
4. prepare the automated generation of interface specifications.

The results of this phase are an electronic version of the UML models..

A1.4.7 Specifications

The goal of the Specifications phase is to produce the CAPE-OPEN Interface Specifications in MIDL (*CO/Ax*) and IDL (*CO/CORBA*) from the design models established in the Design phase.

A1.4.8 CO/Ax Interface Specifications in Microsoft's IDL

MIDL specifications for interface objects will be built from the design models developed in the design phase. The MIDL specifications contain:

1. object overview describing the functionalities and interfaces supported by the object;
2. interface descriptions that list all methods available within the interface;
3. detailed descriptions of the methods with input and output parameters and error codes.

No tools are recommended at this stage: a text editor suffices. However, automated generation of some of the MIDL specifications could be envisaged later on during the project, depending on the reassessment of UML compatible tools.

A1.4.9 CO/CORBA Interface Specifications in CORBA's IDL

IDL specifications for interface objects will be built from the design models developed in the design phase. The IDL specifications contain:

1. object overview describing the attributes and methods provided by the object;
2. detailed descriptions that list all methods available within the interface;

No tools are recommended at this stage; a text editor suffices. However, automated generation of some of the IDL specifications could be envisaged later on during the project, depending on the reassessment of UML compatible tools.

A1.4.10 Implementation

The goal of the implementation phase is to produce prototype ActiveX /COM- and CORBA-compliant components using the interface specifications developed in the Specifications phase. Many of these components will encapsulate legacy code.

A1.4.11 CO/Ax Prototype MIDL Implementation and Encapsulation of Existing Code

Prototype COM-compliant implementations will be produced by generating the interface code with the Microsoft IDL (MIDL) compiler and encapsulating other pieces of code within the interface. The generated interface code comprises:

1. a set of header files defining the interfaces;
2. code for proxies and stubs needed for local and remote methods invocation;
3. code for filling the COM object library.

Legacy code in FORTRAN or other language will have to be wrapped.

The use of the MIDL compiler is mandatory for this phase.

The result of this phase is a binary executable expected to work with ActiveX/COM and ready to be tested.

A1.4.12 CO/CORBA Prototype IDL Implementation and Encapsulation of Existing Code

Prototype CORBA-compliant implementations will be produced by generating the interface code with an IDL compiler and encapsulating other pieces of code within the interface. The generation of interface code comprises

1. class declarations for the interface
2. additional code for client-server communication
3. skeletons to be used for the implementation of the interface

Legacy code in FORTRAN or other language will have to be wrapped. The use of an IDL compiler is mandatory for this phase. The result of this phase is a binary executable expected to work with an ORB (Object Request Broker) and ready to be tested.

A1.4.13 Test

The goal of the Test phase is to do a quick standalone test of the prototype components before submitting them to the VALIdation team. The tests are expected to be done by the developers and should not need external review unless specific problems arise.

The result of this phase is a set of tested prototype components ready to be delivered to the VALIdation team for integration in the validation environment. Both implementations will need such internal testing.

A1.4.14 An Iterative Process

We encourage an iterative approach where the different models and implementations are subject to progressive refinements like in the following spiral development process for standard interfaces to solvers (continuous arrows indicate strong dependencies, dashed arrows indicate that the destination node can take advantage of experience gained in the origin node):

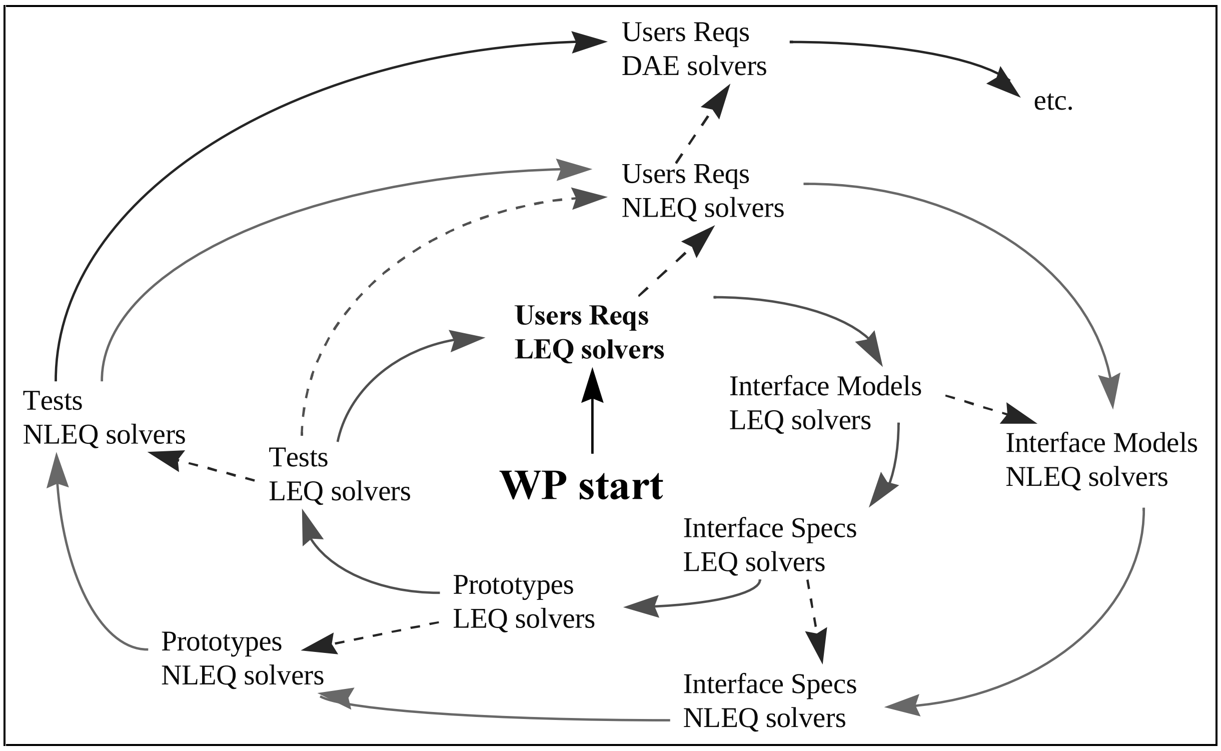


Figure A1.6: Iterative Process

Appendix A2 What is UML?

UML is a method of specifying software. It has arisen from various techniques of object oriented (OO) analysis and design. It is an attempt to unify many different methods into one, hence its name which is an acronym for Unified Modelling Language. What follows assumes that you have some familiarity with OO concepts.

A2.1 Why is CAPE-OPEN using it?

The project is using UML because it is the current best practice in software engineering. UML will be used to present the results of the requirements analysis in the three major parts: Unit Operation, Thermo and Numerical. This will allow readers to see in an unambiguous way what has been done in the work packages and allow the simulator vendors and anyone else to implement the interfaces that connect software components to simulators.

Because UML is a formal and structured method with a clear notation, it conveys user requirements, software analyses and designs in a much clearer and unambiguous way than is possible with traditional methods which employ plain English language. English is notoriously ambiguous and often creates confusion because there are many different ways of saying the same thing. English can also be imprecise which can lead to some confusion in the mind of the reader when it comes to understanding software.

A2.2 What is UML made of?

UML is composed of a set of models. A model is a representation of a particular aspect of the system being analysed and specified. It catches a particular type of information, for example, the dynamic behaviour of the system. All models are consistent with and feed into each other. Models are expressed in textual and diagrammatic forms. It is useful to use both forms because text can express information that is hard to show graphically and also vice versa.

By looking at a system from different directions as represented by the different models, what might otherwise be missed or glossed over in one model becomes obvious in another. In this way a clearer and more complete overall view of the system with less omissions and inconsistencies is obtained.

UML is composed of two types of model: static (Class Diagrams) and dynamic (Use Cases, Sequence Diagrams, Scenarios). CAPE-OPEN does not use all of the possible models in UML, just those that are useful for the project. These are now described in such a way so as to aid the reader in understanding the output of the project. It is not intended to teach how to write UML.

A2.3 Use Cases

A Use Case is a description of one particular usage of a part of the system. A complete analysis requires many different Use Cases, for example there are close to 30 Use Cases for the Unit Operation interface and a similar number for Thermo. A Use Case involves an actor who can be a person or another piece of software. An actor performs some interaction with the system being described and then goes away. It is a specific generic description of a use of the system; it allows interactions in a specific sequence but also allows alternative sequences and loops. The Use Case is attempting to capture the logical interactions rather than the physical appearance of the system. The particular interface used to execute the Use Case, such as keyboard or terminal screen, is not considered. For example, the Use Case that describes how the unit operation obtains its equipment parameters is as follows:

Title	set unit operation specific data
Principal actor	flowsheet user
Description	the builder asks the flowsheet manager to start the selected unit operation's user interface. The flowsheet manager then asks the unit operation to start its user interface if it has one. If the unit operation does not have one, the flowsheet manager either generates its own interface or tells the flowsheet user that no interface is available. If there is one, is it displayed and the user suppliers some or all the unit operation specific data. When the builder completes the interaction the user interface is terminated.
Assumptions	none
Exceptions	no user interface is available from either the unit operation itself or the flowsheet manager.

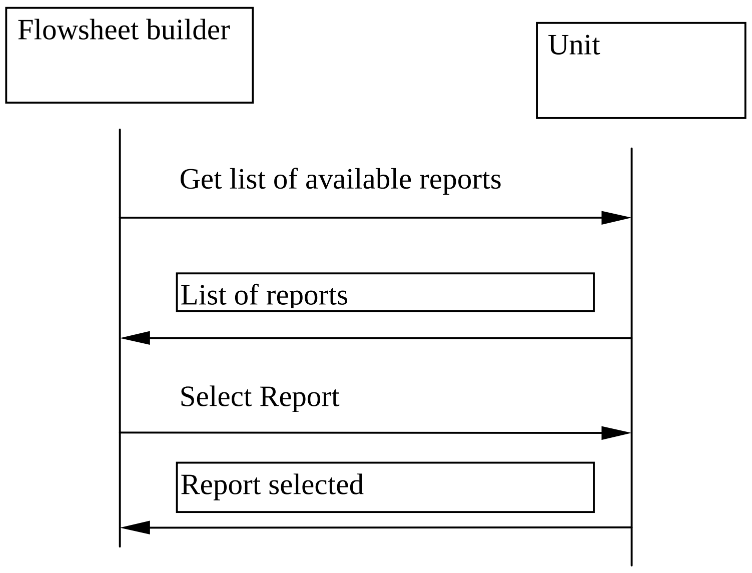
A2.4 Package

A package is a collection of Use Cases which are related, for example, unit operations have packages called "Creating a flowsheet" and "Running a flowsheet". The complete Use Case model is the collection of packages. A Use Case Diagram shows the Use Cases in a package and how they relate to each other and with the actors.

A2.5 Sequence Diagram

A Sequence Diagram shows the sequence of events in a Use Case. It shows the objects involved as vertical lines. The messages passing between the objects are shown as horizontal lines. The Y axis represents time, starting from the top and moving downwards. For example, the Sequence Diagram which shows the process of selecting the report that a unit operation is to output is as follows:

Fig. A2.1: Example of a UML Sequence Diagram



This is a very simple example showing only two objects. Typically there would be three or four different objects and more messages.

A2.6 Class Diagram

The Class Diagram shows classes which represent the objects in the system and how they are related. Each class represents a physical or conceptual entity which contains both attributes and operations in the usual object oriented fashion. Example entities are a unit operation, a user, the flowsheet manager and the simulator. The diagram shows the relationships between classes. For example, a class of objects may be broken down into a set of sub classes which inherit the attitudes and behaviour of their parents class but which are further specialized for some task. Another class may be composed of a set of other classes, a state called aggregation. This diagram is the most important one in UML because it directly maps onto the actual software as it is implemented in an object oriented language. A partial and somewhat simplified example taken from the unit operation work is as follows:

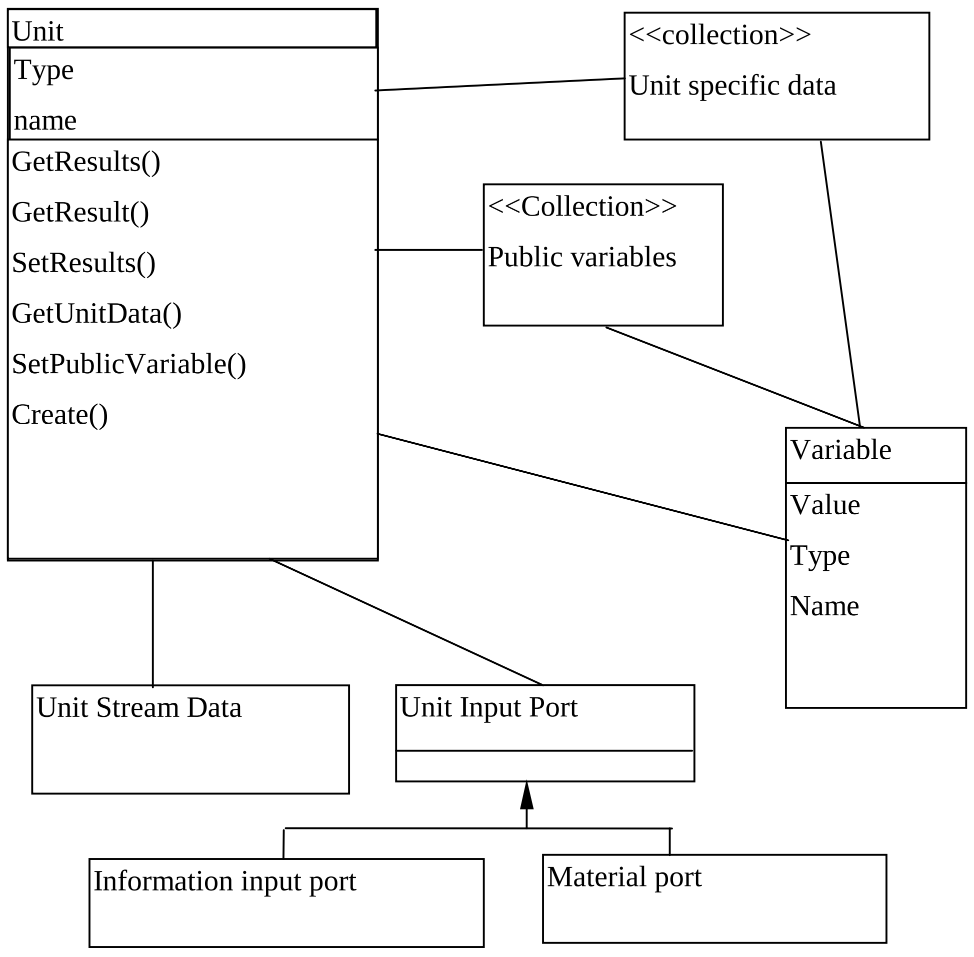
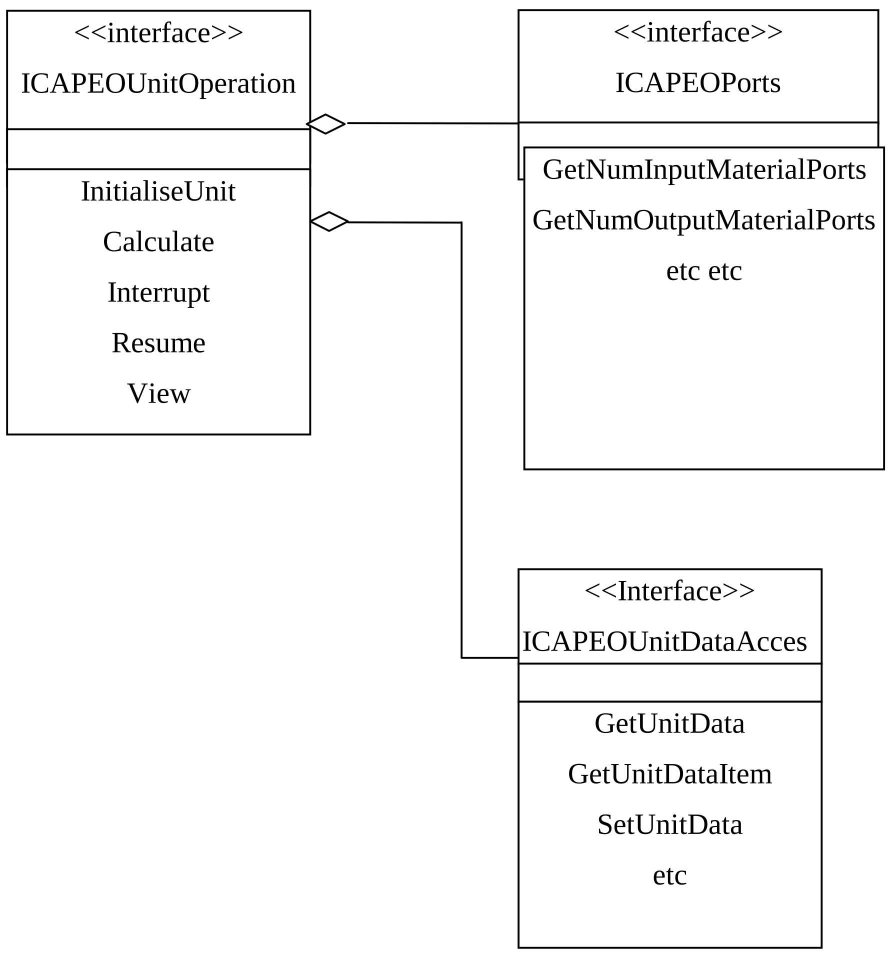


Fig A2.2: Example of a UML Class Diagram

A2.7 Interface Diagram

The Interface Diagram is similar to the Class Diagram except that it shows interfaces. An interface is somewhat like a class except that it presents areas of functionality to the user grouped in a logical “lump”. An interface has no data as a class does, only methods. In addition, an interface can be invoked by any language, OO or not. The functions in an interface can be implemented by many different objects or no objects at all, just normal subroutines. By grouping functionality into interfaces, an application can inquire if a software component supports this or that interface and make intelligent decisions on how to handle it. Another point of difference between a class and Interface Diagram is that interfaces contain each other. Interfaces do not inherit from other interfaces. This allows a useful hierarchical structure to be imposed as a way of organizing the interfaces to a component. The following example is part of the Interface Diagram for the unit operation component. It shows three interfaces (there are more, not shown) which handle the basic control of the unit, its ports and access to its data.

Fig. A2.3: Example of a UML Interface Diagram



Appendix B Process Examples

B.1 Rationale

These examples are written as complete process modelling problems from the point of view of the process engineer. Their purpose is to demonstrate the value of CAPE-OPEN to engineers and managers in the operating companies and to convince them that complete problems can be solved in a variety of combination of different simulator components.

Furthermore, the Validation work package will decide whether they will make use of these examples or write their own for individual components tests, integration tests, and plug and play tests

The idea is to develop the process examples initially in a single simulator, either vendor based or in-house and then as CAPE-OPEN prototypes become available to transfer the examples to mixed simulator components working with CAPE-OPEN standards. More detail is to be worked out for the examples but the kind of issues to be highlighted are for example the ability to take design models forward to the operations stage. At the moment, a change of simulator for different phases of a project means one has to start the model from scratch in the new system. With the CAPE-OPEN interface it will be easier to take the model through from one stage to another. This should be demonstrated to show the value of CAPE-OPEN to the operating companies and their suppliers of process systems software. The examples have been selected through consultation with the operating companies to select areas that are topical, relevant and interesting.

The seven selected examples are representative of:

- Continuous chemical processes with pure components: Example 1
- Liquid (or Solid) / Liquid equilibrium processes: Example 2
- Batch processes: Example 3
- Refining processes with petroleum cuts: Example 4
- Particulate processes: Example 5
- Polymer processes: Example 6
- Electrolyte processes: Example 7

B.2 Example Ex1: Hydrodealkylation of Toluene to Benzene

This example concerns a continuous chemical process with pure components that is well known and well studied in academia.

Description (see process scheme next page): the process involves the hydrodealkylation of Toluene to Benzene (i.e. by using Hydrogen). Hydrogen and Toluene feeds are passed to a mixer and then heated in a heater before passing on to the reactor. A second reaction also takes place in the reactor that leads to diphenyl. There is a cooler following the reactor and a flash vessel. The vapour from the flash vessel are the lights and may be used as fuel. The liquid from the flash vessel consists of Benzene product, diphenyl product and lights. Distillation columns may be used to carry out the separation and the unreacted Toluene is recycled back to the reactor.

Species in the system are Hydrogen, Methane, Benzene, Toluene and Diphenyl.

Thermodynamic: SRK (Soave-Redlich-Kwong) equation of state

Depending upon the level of detail of the model, the separation system can be modelled as a simple splitter or as actual distillation columns connected to each other. Typically, there are three columns in series, the first one is called the stabilizer that separates the lights, the second one is the Benzene column that gives the Benzene product and the third one is called the Toluene column that separates Toluene from Diphenyl.

It should also be possible to incorporate the equations for the reaction rates for the main and the side reactions. Also, it should be possible to define an economic potential in terms of value of products minus the cost of raw materials and to carry out an optimization.

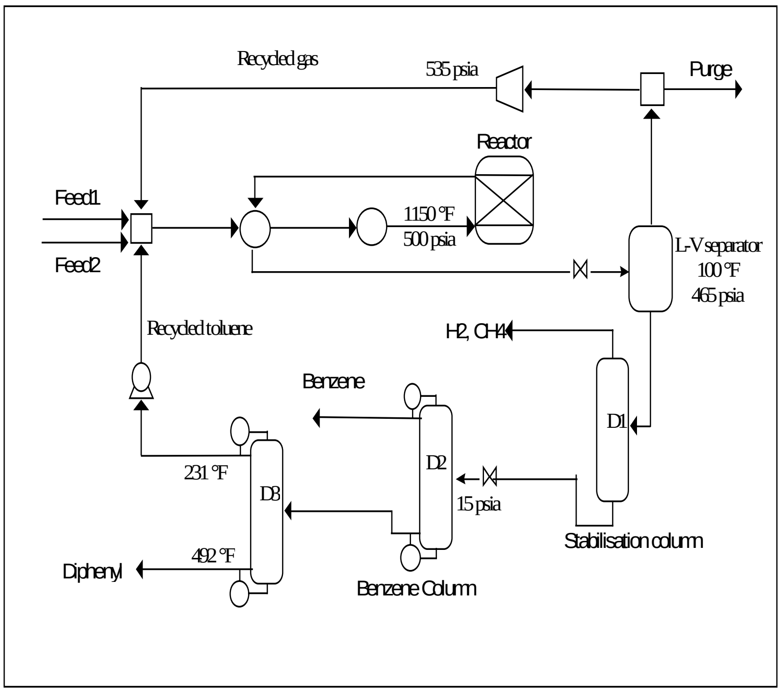


Fig. B1: HDA Example Flowsheet

B.3 Example Ex2: Paraxylene Manufacture

This example is typical of liquid/liquid or solid/liquid equilibrium separations.

The process is a simple equilibration of the xylene isomers followed by a separation of the paraxylene and recycle of the other xylenes (meta- and ortho-) for re-equilibration. The physical properties of the isomers are very close so that the separation is either by fractional crystallisation or by liquid/liquid extraction with a selected solvent. Much has been published on the process, and we could model both variants (re-using the reactor model, which is always the most trouble to set up).

B.4 Example Ex3: Recovery of Coffee by Leaching

Recovery of coffee by leaching is a liquid/solid extraction process and is representative of batch processes. This kind of problem is fairly widespread in the chemical industry. Examples are activated carbon adsorption columns, multi-stage filters (e.g. for exhaust gases) and some catalyst systems with catalyst depletion. Another example is regenerative heat exchange in which the heat reservoirs are cycled from cooling to heating mode.

A series of columns containing ground coffee are connected in series and hot water passed through to extract the soluble solids and the volatile aroma/flavour compounds. As the solids become depleted, each column in turn is replaced with a column containing fresh ground coffee. The columns are arranged such that the most depleted is contacted first by the water and the freshest column last. Column replacement is achieved by removing the most depleted (i.e. first) column and adding the fresh column at the end (i.e. last column). All the other columns are thus promoted by one position at each interchange. The process has two types of unsteady behaviour. First each column is a batch process. When the overall process has been running for a long time, a regular cycle is established with the highest concentration product produced just after a new column is added and the least concentration just before the column exchange. The second type of non-steady behaviour results from the fact that, when first started, all the columns have the same concentration. It takes many cycles (certainly more than "N", where there are "N" columns) before a steady cycle is established.

There are three modes of simulation possible:

1. Pseudo-steady state. Depletion of the columns is relatively slow, so that, if concentration profiles in the columns are assumed, a steady-state simulation can be made to give the outlet concentration.
2. The steady cyclic performance can be obtained by making a dynamic simulation of the columns but solving a set of algebraic equations to give the column concentration profiles at a particular point in the cycle. (The equations result from setting the equality that at the same point in the next cycle, the concentrations are the same as at the previous cycle). The equations are not trivial to set up and solve because strictly we are matching profiles, not point values.
3. The unsteady performance. Simulation starts with the columns all at the same "fresh ground coffee" conditions. The process is then simulated over many cycles.

B.5 Example Ex4: Light Cycle Oil Hydrodesulfurisation

This example is representative of refining processes where petroleum products are usually simulated as pseudo-components.

Description of the process

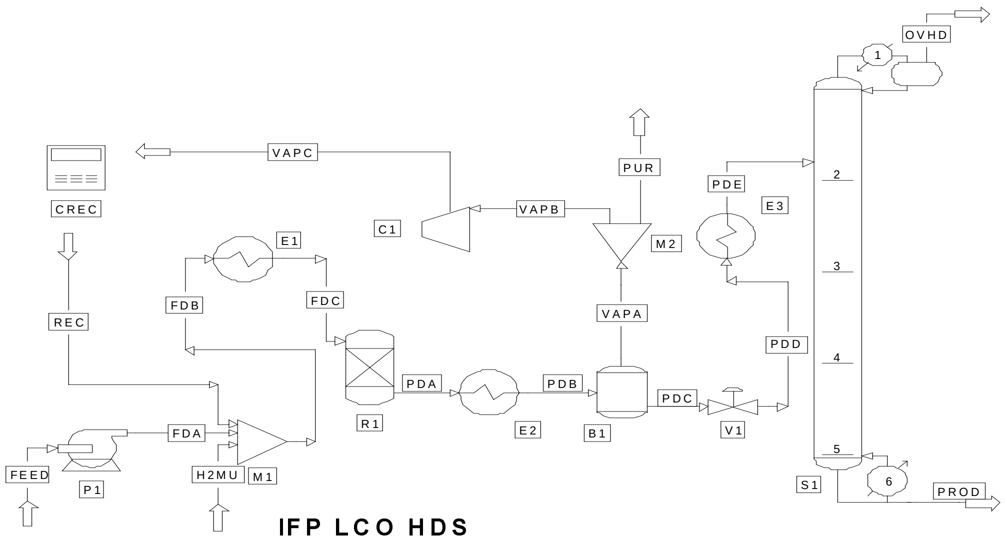


Fig. B2: Light Cycle Oil Process Example Flowsheet

The Hydrodesulfurisation process is intending for removing sulphur compounds from a refining product, here a Light Cycle Oil (LCO). The feed is treated in a fixed catalytic bed reactor with hydrogen. The optimal catalyst behaviour requires a high hydrogen partial pressure. This is provided by recycling hydrogen rich gas. Reactor effluent is cooled down. Vapour and liquid phases are separated. Liquid effluent is sent to a stripper in order to eliminate the light compounds from the end product.

Chemical species: apart from classical chemical pure components like hydrogen, methane and so on, the LCO can be defined as petroleum cuts simulated as pseudo-components. Properties of the pseudo-components must be estimated from two of the three following characteristics: boiling point, relative molecular mass and specific gravity.

Thermodynamic: Grayson-Streed (a usual method in refining) is currently used to calculate the VLE. Enthalpies can be calculated by Curl-Pitzer or Lee-Kessler methods. Density (compressibility factors) can be computed using API method for liquid and Redlich-Kwong for vapour.

B.6 Example Ex5: Particulate Processes

Particulate materials are extensively employed in ore treatment processes. A number of commercial metal ore extraction processes, together with the data to permit their simulation, are presented by A. F. Connolly (PhD thesis, University of Sydney, 1996). The ore comes as a very low concentration of valuable material associated with a valueless material (gangue). The valuable material is physically bound to the gangue both on the surface and within the bulk of the solid ore. The concentration is far too low for it to be economic to use chemical methods (or high temperature metallurgical methods) for separating the valuable product. A combination of crushing, size classification and froth flotation is used to preferentially separate solids with a higher valuable concentration.

There are many recycles (for example of oversized material from the crushers). The product is an ore of sufficiently high concentration to be treated by conventional metallurgical means. The by-product is the gangue material with valuable content reduced to a very low level.

B.7 Example Ex6: Polymer Processes

We could consider the controlled polymerisation of any of the popular polymers to produce a product of desired properties (glass transition temperature, crystalline melting point and molecular cohesion).

The Nylon polymerisation process is a good candidate for Prototypes as it is a polymer process conducted in batch reactors. Batch reactors are today of great interest in most of chemical companies.

We can use published methods of relating molecular cohesion (Kcal/m3) to other thermodynamic properties, for example:

Mol Cohesion = (H - R*T)*rho/M

where H is molar latent heat of vaporisation, rho density and M molecular weight

Molecular cohesion, or cohesive energy density, can also be empirically related to the surface tension of the polymer.

B.8 Example Ex7: Electrolyte Processes

A simple example with only one unit operation can be made available easily. The interest will result in dealing with electrolyte thermodynamics and physical properties calculation.

A complete electrolyte process is ore bio-oxidisation which is also involving particulates.

Appendix C Relation with pdXi and Other Initiatives

In this section, a brief overview of three initiatives, pdXi, CAPE-NET and SVPPM, which may have relation to CAPE-OPEN, are given together with a proposal on how CAPE-OPEN and these initiatives can work together for the benefit of all.

C.1 pdXi

The Process Data eXchange Institute (pdXi) represents an initiative of the Computing and Systems Technology (CAST) division of the AIChE to promote the electronic exchange of process engineering data. Membership in this organization is open to operating companies within the process industry, software vendors, engineering contracting firms and equipment design/manufacturing firms. The objective of pdXi is to establish an ongoing institute to develop and maintain open approaches to electronically exchange and manage process data between computer applications, databases and organizations within the process engineering discipline and within other disciplines. The term process data refers to stream and equipment data and other data used to support the process engineering activity over the entire life cycle of processing facilities.

The structure of the data exchange methodologies for translators to a neutral exchange format requires a standard format for storing the data in the repository. Methods are needed to define the protocols (a schema) for storing the data in the repository. Data (or information) models express schema definitions. It should be noted that the data models describe the schema for the data and not its instances (specific values) and are in neutral form that is independent of the repository used to store the data. If a mapping (a set of rules to translate the schema defined by a data model onto a repository) can be devised, then the data model and mapping can be used to format the repository to accept instances of data (for a more complete description, see Motard et al.: Process Engineering Databases - From pdXi Perspective. Foundations of Computer-Aided Process Design (FOCAPD), 1994, Snowmass, Colorado).

pdXi, has therefore, focused on:

1. The development of a data model (or schema) where the objective is to provide a common medium of communication that, in principle, could encompass all major technical information domains in the chemical industry (the **AP-231** project).
2. The development of an API, that is, a utility for accessing data that conforms to the common data model. The objective is to allow independent applications and organizations to use the API to export and/or import data in domains of common interest without having to be aware of each other's internal data format (the pdXi interface).

AP-231: The objective of this project is to develop Units of Functionality (UoF) for process engineering data covering substance thermo-physical properties and thermodynamic model parameters, conceptual process description, process simulation results, process specification of major process equipment, process connectivity, basic control strategies and PFD information. These are based on the use of data models which allow a high degree of attribution in model classes, inheritance used to share super-class attributes with appropriate subclasses and are suited to object-oriented environment. A large set of data models have been developed and are classified as,

1. Planning model (provides high level data as well as overall model relationships) - site information, plant information, process definition, process representations and unit of measurements.

2. Simulation-oriented data - physical properties, materials data (process streams), unit operations, topology.
3. Equipment-oriented data - shell & tube heat exchangers, pumps, compressors, vessels & tanks (including nozzles), separation towers and internals.

pdXi Interface: This is a programming utility for applications to access data, for example, between computer applications or organizations. Two forms of data transfer are considered - Working form and STEP form. The STEP form is a neutral file format defined by ISO protocols and the EXPRESS data model. The pdXi API has two options, a low level API and a high level API. The low level API provides complete access to the primitive data entities as defined by the full pdXi EXPRESS data model. The functions provided at this level are based on the ISO Standard Data Access Interface (SDAI). The low level API, however, requires navigation through the entities defined in the EXPRESS data model to retrieve or modify the desired data. That is, the user of the low level API must be familiar with the pdXi data model (which is based on the University of Missouri data model). The high level API provides a number of “convenience functions” and is intended to present blocks of data in a form that is more familiar to a process engineer than the complex “fragmented” representation inherent in EXPRESS-based models.

A data exchange application is built by writing a program that reads and/or writes application data in terms of pdXi data model entities from/to a STEP Neutral File by using pdXi API calls. The program is compiled and linked with the pdXi API library to create an executable. The data sender uses such an executable to convert the data to be communicated to the pdXi representation as a Neutral File. The data receiver reads the transmitted Neutral File’s pdXi representation of the data with another data exchange program and saves the data in the receiver’s own form.

C.1.1 Current Status

The text given below is based on the information received from pdXi.

A number of UoFs have been developed. A list of the available data models with their description is also available. The work on the development of the pdXi API is in the Beta Test phase. Further documentation on pdXi is scheduled to be released. It is possible to get further information on pdXi at their official web site at http://www-chen.tamu.edu/pdXi/.

C.1.2 Relation between pdXi and CAPE-OPEN

Based on the brief overview of pdXi, a number of areas can be identified where pdXi and CAPE-OPEN should find ways to collaborate. pdXi is about data definition (data-centric), whereas, CAPE-OPEN is about component interoperability (model-centric). Thus, if pdXi is to be considered as an organization that promotes the use of open approaches to electronically exchange and manage process data, then there are no apparent overlaps with the objectives of CAPE-OPEN and a collaboration will be to the mutual benefit of both. It should also be noted that while pdXi has been set-up as a institute with perhaps a long life time, the present European Union funded CAPE-OPEN is a project of short duration (the present project is likely to provide the backbone for an open CAPE that will lead to a broader ongoing effort such as WW-CAPE or Global-CAPE). For the purposes of the current CAPE-OPEN project, for successful collaboration, it is necessary to carefully define the topics for collaboration and coordinate the time of the deliverables. Also, pdXi is a work in progress and there is a real opportunity for harmonisation of the high-level API with the CAPE-OPEN interfaces for overlapping data such as material/stream objects. The following areas could be considered for collaboration with pdXi:

1. Meetings to inform each other of the developments on projects (work packages) of mutual interest. These meetings may also be used to demonstrate a software or discuss new results. pdXi has offered to give a demonstration to CAPE-OPEN of the pdXi API.
2. Discuss with pdXi on the feasibility of a common approach for work packages 2.2 (Physical Properties Database Interface) and 2.3 (Simulation Databank Interface).
3. Investigate the possibilities of incorporating the pdXi data models into CAPE-OPEN and the CAPE-OPEN standards into the pdXi data models.
4. Consider the feasibility of a jointly defined open standard for the data exchange application executive (which calls the pdXi API for data exchange).

It should be noted that copies of the pdXi data models for properties, streams, components and materials were made available to BASF who is the THRM work package leader. Collaboration on the definition of the data exchange executive with pdXi will be beneficial to the work packages 2.2 and 2.3.

C.2 CAPE-NET

CAPE-Net is an EU-supported initiative to enhance research coordination and technology transfer in CAPE. CAPE-NET comprises of a number of major industrial and research organizations within the member countries of the EU. A number of CAPE-OPEN partners are also represented in CAPE-NET (for example, ICI, IFP, RWTH, ICSTM and ENSIGC). The objectives of CAPE-NET are to

1. Identify, assemble and share knowledge, experience and understanding of current world-wide best practice in CAPE research, application and education.
2. Collect, collate and document this information to form a package suitable for transfer to the wider research, industrial and educational communities.
3. Take appropriate actions and initiatives to disseminate and transfer this package to its target communities.

CAPE-NET officially started on July 1, 1997 and has been granted funding for three years. Among the listed activities of CAPE-NET, the following may have particular significance to CAPE-OPEN: seminars/workshops, newsletter, case studies, benchmarks, best practice guides and demonstrations.

C.2.1 Relation between CAPE-NET and CAPE-OPEN

While CAPE-OPEN is a research project, CAPE-NET is not, and thus there are no overlaps in their corresponding objectives. There are, however, a number of ways that the two projects can benefit from each other. For example, CAPE-NET could serve as the medium for informing about the CAPE-OPEN standards and promoting the use of the CAPE-OPEN standards. CAPE-OPEN, on the other hand, should keep the leadership group of CAPE-NET informed about the progress of the project and participate in events organized by CAPE-NET to promote the CAPE-OPEN standards. Workshops and short courses could be jointly organized.

C.3 SVPPM

SVPPM (Standardized Validation of Physical Properties Models) for is an international project that has been initiated to develop standardized tests of problems and data that can be utilized to discriminate strengths, weaknesses and inconsistencies of physical properties models used in chemical process design and simulation.

In the past two decades, there has been a tremendous expansion of research into models for properties. Both fundamental and applied journals publish many new expressions each year, leading to a bewildering array of application possibilities. The number of qualified properties workers and the time available has significantly decreased as the number of newly generated models, the needs of new processes and products and the redesign of older plants has increased. The result is the undesirable situation of research results languishing for lack of validation, insufficient models to meet the demands of new problems and considerable frustration of both generators and users of property models. Generators often do not see their work being utilized and users often ignore state-of-the-art models.

Physical properties models are intended to describe a great variety of properties for all classes of substances and mixtures over a wide range of conditions. The consequence is that verification will demand somewhat different methodology than flowsheeting or process control. In particular, there are several levels of increasing complexity for which users could desire to have validation. Based on experience, SVPPM project has identified five elements and representative questions that should be applied to the computer code of a model:

1. Thermodynamic rigor and proper physical behaviour: Does the program show proper stable roots (real and positive), mechanical/thermal derivatives and stability, mixture derivatives and diffusional stability? Is the Gibbs-Duhem Equation obeyed for all properties? Are the limits correct (ideal gas law, infinite T and P)? And many more.
2. Qualitative features: Are the known extrema of derivative properties seen (for example, reduced bulk modulus or the speed of sound maxima)? Are the multiphase equilibria and critical lines (VLE, LLE, VLLE, etc.) given?
3. Quantitative Correlation: How accurately can the saturated vapour pressures, saturated vapour and liquid volumes critical compressibilities and other properties be fitted to model parameters? How accurately can mixture (binary, ternary) VLE and LLE data be correlated by the model parameters? How many data and what different properties are needed to obtain a reliable fit? What sensitivity, lack of uniqueness and initialisation issues arise with the model?
4. Quantitative prediction: If no or few data are given at certain conditions, how accurate is the description of systems at remote conditions? What happens when binary parameters are set to zero?
5. Application efficiency and robustness: Do the calculations converge rapidly and reliably? Does the model give reproducible results with different computational schemes for process unit operations?

C.3.1 Current Status

Work on Element 1 has nearly been completed. Work on Elements 2-4 have been initiated. In the first phase of the project, it was decided to concentrate only on equations of state (all forms). In order to test the thermodynamic rigor and physical behaviour of a model, a software has been developed where the user can add their model (an equation of state) and get the results of the tests. Thus, a standard way of introducing user-added models to the software based on the tests of Element 1 has been developed. There are a number of issues that needs to be considered in the introduction of an user-added equation of state (for example, pure component parameters, mixing rules, model expression (in terms of pressure?), derivatives, and so on).

With respect to work in Elements 2-4, the first task is to collect a representative set of experimental data that will be used for the tests. The next phase will be to find out how to transfer the data for a specific application and how to allow user supplied data.

Prof. John P. O'Connell (Department of Chemical Engineering, University of Virginia) is the director of the SVPPM project with NIST and DTU as two contributing partners. A leadership group comprised of representatives from NIST, DTU, UCL, Union Carbide, Exxon and University of Virginia defines the activities of the SVPPM project. More details on the SVPPM project and the test algorithms for Element 1 can be found in the project web site at `http://gibbs.che.virginia.edu/~svppm/INDEX.HTM`.

The preliminary results of the SVPPM project and a prototype software will be presented at the 8th International Conference on Properties and Phase Equilibria for Product and Process Design, April 26, 1998, The Netherlands.

C.3.2 Relation between SVPPM and CAPE-OPEN

Obviously, the SVPPM project can have the greatest significance with the THRM work package. The results of the SVPPM project will point the safest way for incorporating external thermo models into a host simulator. CAPE-OPEN should not define standards which may not agree with some of the tests proposed by the SVPPM project. On the other hand, CAPE-OPEN can help the SVPPM project in terms of defining the standards for introduction of external thermo models into a host software. The SVPPM project will also benefit from CAPE-OPEN in terms of efficient storage and exchange of data. Finally, element 5 also asks the same questions that have been asked by some of the partners of CAPE-OPEN. If these questions are important to CAPE-OPEN partners, collaboration on this element should also be considered.

C.4 Other Related Projects/Research Activities

Two research areas rather than research projects or organizations are mentioned below since the number of on-going projects at different organizations (academic and industrial) is quite large in these research areas. These two areas are modelling and system integration. The relationship to CAPE-OPEN is briefly considered below.

There is a lot of research being carried out in the area of computer aided modelling and model generation. Research in this area is complimentary to the objectives of CAPE-OPEN as this provides another way of introducing user-supplied (or generated) external modules into a host simulator.

Research and development of software for integrated systems (which combines in some way, models, simulators, simulation mode (steady state, dynamic), applications, etc.) are also being carried out in a number of academic and industrial institutions. CAPE-OPEN needs to be aware of these developments as they may propose alternative ways for introducing external software (for unit operations, thermo, solvers, etc.) into a host software.