SOFTWARE ARCHITECTURES AND TOOLS FOR COMPUTER AIDED PROCESS ENGINEERING

Edited by

B. BRAUNSCHWEIG

R. GANI

COMPUTER-AIDED CHEMICAL ENGINEERING, 11

ELSEVIER

SOFTWARE ARCHITECTURES AND TOOLS FOR COMPUTER AIDED PROCESS ENGINEERING

COMPUTER-AIDED CHEMICAL ENGINEERING, 11

SOFTWARE ARCHITECTURES AND TOOLS FOR COMPUTER AIDED PROCESS ENGINEERING

Edited by

Bertrand Braunschweig
Computer Science and Applied Mathematics Department
Institut Français du Pétrole
1 & 4 avenue de Bois Préau
92852 Rueil Malmaison Cédex
France

Rafiqul Gani
CAPEC, Technical University of Denmark
Department of Chemical Engineering,
Building 229, DK-2800 Lyngby,
Denmark

2002
ELSEVIER
Amsterdam - Boston - London - New York - Oxford - Paris
San Diego - San Francisco - Singapore - Sydney -Tokyo

ELSEVIER SCIENCE B.V.
Sara Burgerhartstraat 25
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

First edition 2002

Library of Congress Cataloging in Publication Data
A catalog record from the Library of Congress has been applied for.

British Library Cataloguing in Publication Data

```
Software architectures and tools for computer aided process
    engineering. - (Computer-aided chemical engineering ; 11)
    1.Chemical process control 2.Computer integrated
    manufacturing systems
    I.Braunschweig, Bertrand II.Gani, R. (Rafiqul)
    660.2'8155
```

ISBN: 0-444-50827-9

The paper used in this publication meets the requirements of ANSI/NISO Z39.48-1992 (Permanence of Paper).
Printed in The Netherlands.

Foreword

Industrial production is a key area of human activity. Due to its multi-dimensional importance it is strongly linked to the three pillars of Sustainable Development: economic competitiveness, social importance (employment, quality of life), and environmental impact.

The future economic power of the European Union will depend on the capability of industry to produce goods and services combining environmental awareness and competitiveness.

In their efforts to assist EU industry, European Research programs such as Brite Euram (FP4) and GROWTH (FP5) have taken these concepts into consideration. The future of European Industry, in particular within the framework of sustainable development, has to be prepared with an overall vision, hence the importance of European Scientific and Technical cooperation.

To leave problems for the next generation to inherit is not acceptable. Indeed, sustainable development encompasses the ability to produce goods that create jobs and guarantee quality of life, without generating a negative impact on the environment.

The modernisation of the industrial processes and adaptation to change are achieved through research activities that will develop new technologies and methodologies. They encompass process modelling and simulation, management systems, process integration and intensification. Agility, efficiency, safety and prevention of waste are considered as key goals.

Through European research programmes, industry and associated research organisations are guided towards cost-shared research actions through a system-oriented approach in which chemistry, physics, engineering, computer science or social sciences become essential and interdependent.

This publication is just one outcome of collaborative efforts undertaken by the projects CAPE-OPEN and Global CAPE-OPEN. The need for CAPE tools is mostly found in the production of fine chemicals and petrochemical products, but also for processes in traditional manufacturing sectors.

Establishing the latest state-of-the-art in CAPE will be of great interest to process engineers and scientists, plant designers, software developers, teachers and students.

Frédéric Gouardères
Scientific Officer
European Commission
Research Directorate-General

Preface

It has been a pleasure for us to contribute to this book as well as to edit it. The idea of editing a book on modern software architectures and tools for CAPE came while we were collaborating with several industrial and research organisations through the CAPE-OPEN initiative. We realised that we were producing a wealth of useful material and information that deserved a broader audience. Moreover, although there are numerous books written by leading authors on various aspects of Computer-Aided Process Engineering, we could not find one single reference on the software side of it, that is, how CAPE software is designed and developed? After discussion with a number of colleagues and friends, we came to the conclusion that such a book could be quite useful for the CAPE community at large.

The book has benefited from many contributions and support from individuals and institutions. The European Commission funded the CAPE-OPEN and Global CAPE-OPEN projects through the Brite-EuRam programme, and the GCO-Support project through the Competitive and Sustainable Growth programme of the 5th Framework. The Intelligent Manufacturing Systems programme supported the international activities of Global CAPE-OPEN by endorsing the project and thus allowing collaboration between Europe, Japan, USA and Canada.

We thank all authors for their contributions and their willingness to satisfy our requirements. Similarly, we are grateful to the anonymous reviewers who have provided valuable comments and suggestions. We also thank Elsevier for their interest in the book and for publishing it. Finally, we hope that the reader will find the subject matter of the book interesting, the information content useful and the vision motivating. We hope this book will contribute to the development of a new generation of software that will match the future needs of the CAPE community and beyond.

Bertrand Braunschweig
Rafiqul Gani

List of Contributors

R. Andrews	School of Chemical Engineering, University of Edinburgh, Edinburgh, Scotland EH9 3JL
N. Arora	Chemical Engineering Department, Carnegie Mellon University,Pittsburgh, PA 15213-3890, USA
P. Banks	Peter Banks Associates, Process Simulation Consultancy, 9 Emmets Park, Binfield, Berkshire RG42 4HQ, UK
P. I. Barton	Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
R. Batres	Tokyo Institute of Technology, Research Laboratory of Resources Utilisation, 4259Nagatsuta, Midori-ku, Yokohama 226-8503, Japan
B. Bayer	Lehrstuhl für Prozesstechnik der RWTH Aachen, 52056 Aachen, Germany
J.-P. Belaud	Laboratorie de genie Chimique, INPT – ENSIGCT, 18 Chemin de la Loge, F – 31078 Toulouse Cedex 4, France
L. T. Biegler	Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA
I. D. L. Bogle	Department of Chemical Engineering, University College London, Torrington Place, London WC1E 7JE, UK
B. Braunschweig	Computer Science and Applied Mathematics Department, Institut Français du Pétrole, 1 & 4 avenue de Bois Préau, 92852 Rueil Malmaison Cédex, France
D. Cameron	Fantoft Process Technologies AS, PO Box 306, N-1301 Sandvika, Norway
I. T. Cameron	Department of Chemical Engineering, The University of Queensland, Brisbane, Queensland, Australia 4072
C. Clausen	Statoil AS, Trondheim, Norway
P. Edwards	Pete.Edwards@attglobal.net
M. Eggersmann	Lehrstuhl für Prozesstechnik der RWTH Aachen, 52056 Aachen, Germany
T. I. Eikaas	Cyberlab AS, Trondheim, Norway
E. S. Fraga	Department of Chemical Engineering, University College London, Torrington Place, London WC1E 7JE, UK
R. Gani	CAPEC, Department of Chemical Engineering, Technical University of Denmark, DK-2800 Lyngby, Denmark
M. Graells	Chemical Engineering Department, Universitat Politècnica de Catalunya. E.U.E.T.I.B., Comte d'Urgell 187, 08011-Barcelona, Spain
J. Hackenberg	Lehrstuhl für Prozesstechnik der RWTH Aachen, 52056 Aachen, Germany
G. Heyen	Laboratoire d'Analyse et Synthèse des Systèmes Chimiques, Université de Liège, Sart Tilman B6A, B-4000 Liège, Belgium
D. Hocking	Hyprotech Ltd., Calgary, Alberta, Canada
K. Irons	Engineering Sciences - Market Development, 1400 Building The Dow Chemical Company, Midland, MI 48674, USA
M. Jarke	Informatik V (Information Systems), RWTH Aachen, 52056 Aachen, Germany
J. Köller	Informatik V (Information Systems), RWTH Aachen, 52056 Aachen, Germany
A. Kuckelberg	Informatik V (Information Systems), RWTH Aachen, 52056 Aachen, Germany
D. Leung	Chemical Engineering Department, The University of Sydney, Sydney, NSW,

	2006, Australia
T. I. Malik	ICI Strategic Technology Group, Wilton, UK
W. Marquardt	Lehrstuhl für Prozesstechnik der RWTH Aachen, 52056 Aachen, Germany
K. W. Mathisen	Norsk Hydro ASA, Corporate Research Centre, Porsgrunn, Norway
W. Morton	School of Chemical Engineering, University of Edinburgh, Edinburgh, Scotland EH9 3JL
R. Murris	Mi^2 B. V., The Netherlands
J. M. Nougués	Hyprotech Europe S.L., Pg. Gràcia 56, 08007 Barcelona, Spain
H. Okada	Engineering & IT Department Engineering Division, JGC Corporation, 2-3-1, Minato Mirai, Nishi-ku, Yokohama 220-6001, Japan
M. Pons	TotalFinaElf, CRDE, BP 61005, 57501 St Avold Cedex France
J. W. Ponton	School of Chemical Engineering, University of Edinburgh, Edinburgh, Scotland EH9 3JL
L. Puigjaner	Chemical Engineering Department, Universitat Politècnica de Catalunya. E.T.S.E.I.B., Avda. Diagonal 647, 08028-Barcelona, Spain
C. Quix.	Informatik V (Information Systems), RWTH Aachen, 52056 Aachen, Germany
G. V. Reklaitis	School of Chemical Engineering Department, Purdue University. West Lafayette, IN 47907-1283, USA
J. C. Rodríguez	Hyprotech Europe S.L., Pg. Gràcia 56, 08007 Barcelona, Spain
J. A. Romagnoli	Chemical Engineering Department, The University of Sydney, Sydney, NSW, 2006, Australia
S. Sama	Hyprotech Europe S.L., Pg. Gràcia 56, 08007 Barcelona, Spain
R. Schneider	Lehrstuhl für Prozesstechnik der RWTH Aachen, 52056 Aachen, Germany
M. Schoop	Informatik V (Information Systems), RWTH Aachen, 52056 Aachen, Germany
T. Shirao	Science & Technology Research Center, Mitsubishi Chemical Corporation, 1000, Kamoshida-cho, Aoba-ku, Yokohama 227-8502, Japan
V. Siepmann	Norsk Hydro ASA, Corporate Research Centre, Porsgrunn, Norway
B. J. Stenhouse	BP Amoco, Research and Engineering Centre Chertsey Road, Sunbury on Thames, Middlesex TW16 7LN, UK
T. List	Informatik V (Information Systems), RWTH Aachen, 52056 Aachen, Germany
T. Teague	Protesoft Corporation, 10400 S. Post Oak Road, Suite E-PMB#300, Houston, TX 77035, USA
J. E. Tolsma	Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
L. von Wedel	Lehrstuhl für Prozesstechnik der RWTH Aachen, 52056 Aachen, Germany
M. White	Salmon River Software, 301 Newbury Street, Danvers, MA 01923, USA
M. R. Woodman	BP Amoco, Research and Engineering Centre Chertsey Road, Sunbury on Thames, Middlesex TW16 7LN, UK

Contents

*The full version of this chapter is printed on pages 433-452.
**We regret that this chapter could not be placed in sequence.
The chapter is printed on pages 663-700.

Part I Introduction

1.1 Introduction

B. Braunschweig & R. Gani

This part consists of one chapter, which provides an introduction to the book, the contents of the book as well as a road map for the reader.

Computer aided process engineering, CAPE, is briefly discussed together with CAPE methods and tools from a software architecture perspective. A brief overview on computers & software and software technologies is also given.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 1.1 Introduction

B. Braunschweig & R. Gani

1.1.1 SOFTWARE ARCHITECTURES & CAPE

Computer Aided Process Engineering (CAPE) implies the use of computers and/or computer aided methods and tools in the solution of process and product engineering problems. According to Perris and Bolton (2001), CAPE may be defined as,

> *"The application of a systems modelling approach to the study of processes and their control, safety, environmental protection and utility systems as an integrated whole, from the viewpoints of development, design and operation"*

CAPE software, as for many other classes of software, is becoming increasingly powerful, increasingly complex, increasingly better, increasingly used, and, last but not the least, increasingly diverse. The range of applications of CAPE tools, from simple database searches for retrieving reference properties of chemical compounds, to complex plant-wide dynamic optimisation problems solved in real time, is overwhelming. It calls for numerous methodologies, techniques, tools, and algorithms in order to cope with increasing demands, users and applications.

On the other hand, software tools, architectures and technologies are evolving at a very rapid pace. They are triggered by the constant need of software providers for offering new commercial products; they are triggered by the constant need to transform new CAPE methodologies, techniques and algorithms into more versatile and powerful software; they are triggered by the opportunities offered by more powerful hardware and faster networks (including the internet, of course); and they are even more triggered by the growing demands of users for more features, more user-friendliness, and more precise and reliable results.

In the text below, we first give our views on *CAPE methods and tools* from a software architecture perspective, followed by a brief overview on *computers & software and software technologies* before introducing the contents of this book.

1.1.1.1 CAPE methods and tools

CAPE methods refer to a collection of computer-aided methodologies, techniques and algorithms that provide the framework for the CAPE tools. CAPE tools refer to the software that have been developed by transforming CAPE methods for a wide range of applications and uses. In this book we will refer to CAPE tools also as software. The software may be a single software component or a collection of software components integrated into one main software. A typical example of CAPE methods are methodologies for modelling, numerical techniques and equations solving approaches that provide the framework for CAPE tools such as process simulators. Process simulators are therefore a collection of individual software components put together through a defined architecture.

Differences in the available CAPE methods of a particular type, provides distinguishing features to the corresponding CAPE tools or software components. A process simulator based on a modelling methodology that employs predefined steady state unit operation models and a sequential modular approach for solving the model equations, is a CAPE tool for steady state process flowsheeting based on the sequential modular approach (see chapters 3.3 and 5.2 for more details).

Differences in the types of CAPE methods lead to the development of different types of CAPE tools or software components. For example, computer aided molecular design techniques has led to the development of software for solvent design (Harper and Gani, 2001) while knowledge-based techniques have lead to the development of software for selection and design of equipments (Bühner and Schembecker, 2001). Some CAPE methods, such as methods for process/product synthesis and design, data-reconciliation, real-time process optimisation, *etc.*, need more than one CAPE tool or software component. Bayer *et al.* (2001) give a good example of the need for different software tools for conceptual process design (see also chapter 7.1). Efficient transformations of these CAPE methods into useful tools need corresponding and appropriate software architectures.

New challenges and opportunities in CAPE indicate the need for integration of CAPE methods. As in the case of software components or processes, the integration of methods implies an overlap of activities (or work) and the sharing of information (or data). Rather than apply the CAPE methods and the corresponding software in a sequential manner and thereby repeat many of the common computations, the challenge is to satisfy the workflow and dataflow needs while performing the overlapping computations only once. In such architecture, the common parts of the workflow are performed only once and the common parts of the data are stored such that all other tools can access it. So, in this case, we have integration of methods as well as tools, which in turn, may involve integration of individual software components (see also chapter 6.2). Such integration allows one to consider aspects of operation, environment, cost and other process issues simultaneously, for example, in at the conceptual design stages of the process/product development or for generation/evaluation of retrofit alternatives for existing processes or products. The need for integrated CAPE tools becomes clearer if lifecycle issues are also considered. Here, the challenge is to also handle different scales of time and length. For example, the lifecycle stages in synthesis and development of manufacturing processes need to consider a number of concurrent CAPE issues or problems (Ng, 2001; Okada and Shirao, 2001). Software architecture therefore becomes an important issue in the transformation of integrated CAPE methods into integrated CAPE tools.

In the text below, some of the issues with respect to making CAPE software more powerful, increasingly complex, increasingly better, increasingly used, and increasingly diverse is briefly discussed from a software architecture perspective.

Models versus scope

All CAPE tools are model based, and therefore, models play a very important role in defining the scope and significance of the software. The models, of course, can be of different types and/or class (see chapters 3.1 and 6.1). Transformation of CAPE methods into tools usually occur via models (or modelling systems), which when solved, provides the information (knowledge) that is used elsewhere. For example, a CAPE tool such as a process simulator may consist of a number of steady state unit operation models with different degrees of complexity. Application of the process simulator in this case, provides information about the process streams through the solution of mass and energy balance equations representing the process. So, in this case, the CAPE tool generates knowledge about the steady state behaviour of the process. This knowledge may be used by other tools to make decisions on the conditions of operation, to improve the accuracy of the model and many more. Thus, at the core of most software components, usually lies a model. Consequently, at the core of integrated software components, the model component is usually in the inner core. Therefore, the accuracy of the information generated by the model affects all other software components. For similar reasons, the application range of the model defines the scope of the integrated or individual software components.

The scope of CAPE tools may also be increased through multiscale and multilevel modelling (see chapter 3.1). As pointed out by Pantelides (2001), significant advances in process modelling can be achieved through multiscale modelling, which also provides a framework for integrating aspects of process and product design (where also questions of scales of size and time need to be considered).

The need for models at different lifecycle stages of the same process, on the other hand, may be referred to the need for multilevel modelling, although, indirectly, scales of time are also involved here.

Multilevel modelling may also mean different uses of the CAPE method through different levels of model complexity or model type. For example, the transient operation of a process may be studied by combining a dynamic process simulation model with a process operation model (as in batch operations). Here, having the dynamic simulation model is not enough, the operational (batch) procedure is also needed. Also, if the generated operational information is to be used to make business decisions, it can be argued that a business model also needs to be considered. Therefore, adding different levels of models may help to increase the scope and significance of the original software (or CAPE tool).

Flexible software architectures will allow the developer of CAPE methods and tools to meet the demands discussed above with respect to models and their use in software development. The architecture needs to allow the generation of models according to the needs of the specific problem being solved (see also chapters 5.1 and 6.1). Note that the CAPE method is transformed into software through the model that also helps to define the framework for the architecture of the software.

Complexity versus reliability

Complexity and reliability are closely related to the model(s) that has been used to transform the CAPE method into software. Complexity here also refers to the ease of use, *i.e.*, how easy is it to use the software? Thus, complexity and reliability of the model determines, indirectly, the scope and significance of the CAPE tool while complexity of parts of the software architecture dealing with the user interface, determines the usability of the software. Naturally, the demands for the software to become increasingly used, increasingly better, increasingly diverse, and increasingly reliable can be addressed by considering the model complexity-reliability, user-interface complexity, *etc.* in the early stages of the software development. More precisely, during the definition of the framework for the software architecture. Here, the objective of the software architecture could be such that only those parts of the model and interface (from the total resident system) that are needed are retrieved and used for the specific problem being solved. This is similar to integrating individual software components of different level and scale into problem specific software through *intelligent* software architecture (the basis for which could be other CAPE methods). Increasing reliability may mean the use of more complex models and/or larger computation times while more features in the model may mean more flexibility. The architecture needs to find an appropriate balance.

Examples of the use of models with increasing complexity or integration of different levels of models can be found in applications where the capabilities of CFD tools are combined with those of more conventional modelling technology (see also chapter 7.2).

Integrated methods versus integrated software

This has been briefly addressed above (section 1.1.1.1). The text below further adds to the discussion.

It is important to understand that just as individual software components may be integrated together to form an integrated tool (software), different individual CAPE methods may also be integrated to form an integrated CAPE method (Bansal *et al.* 2001). For the integrated CAPE method to work, an integration of the corresponding integrated tools (at the integrated software level) or an integration of all the individual software components corresponding to all the integrated tools (at the individual software component level). For integration of CAPE methods, models (or model components or multilevel modelling) play a very important role in transforming integrated methods into integrated tools and define certain aspects of the software architecture. Integration between integrated tools and/or software components need clear definition of the related workflow and dataflow and their representation through activity models (see chapters 2.3, 6.2 and 7.1). More integration will lead to software that is more powerful, increasingly complex, increasingly better, increasingly diverse, and, if all these demands are met, also increasingly used. For example, integration with tools for data reconciliation (see chapter 3.4) and/or fault diagnosis systems (see chapter 6.4) enhances the application and use of the software components.

For integrated software, the software architecture will need to consider how much to involve the user in the various solution steps. For example, should all the conceptual design steps be made automatic or should there be user interaction at each step? Good resolution of these issues should lead to increased application and/or use of the software.

Computer power versus problem size

As computers become bigger (in memory), faster and better, the CAPE tools taking advantage of these features will also become bigger, faster, better and more powerful. To take advantage of the advances in computer science and computers, software architecture will also need to address the hardware and related issues (see also section 1.1.1.2). More computer power should not be the only criteria to increase the problem size and solution speed. Appropriate software architecture should try to optimise the size, speed, flexibility, and efficiency of the software.

User productivity versus application

This topic brings us back to the question of the objectives of CAPE and CAPE methods and tools. By providing a systematic computer aided approach to problem solving, the CAPE tools indirectly contribute positively to the productivity of the user. For example, if the user needs to spend less time to solve a CAPE related problem, he/she will then have more time to solve other important CAPE problems, assuming of course, that the CAPE tool is able to solve these problems. The software architecture has an important role here. It needs to consider the interaction between the user and the software (through the computer) so that the user will spend very little time in defining the problem that he/she needs to solve. The software (through the computer) should be able to correctly interpret the user-given data, retrieve the necessary software components, solve the problem, and generate the results report for the user. The human (user) – computer relationship is important because both have roles in the problem solution. The software architecture needs to make sure that both parties play these roles efficiently. The ratio of computing time used (to get a solution) divided by the total time used in the problem solution (including the time spent by the user to generate and/or provide the necessary data and information, the number of trials, *etc.*) could be used as an indicator to monitor user productivity. Unless special or complex and/or large problems are being solved, usually, the computer is able to solve the problem (for a successful convergence) at a very small fraction of the time needed by the user to correctly set-up the problem. If the user needs less time to set-up the problem and is able to obtain reliable results within a few attempts, the productivity of the user will increase.

The other important aspect (also partially covered above) is the question of application. It would be impossible to provide in single monolithic software all the features needed for a wide range of applications. For example, from conceptual design to on-line control and monitoring and diagnosis, from continuous to batch and hybrid, from traditional sectors *i.e.* chemicals and petroleum to food, pulp and paper, *etc.* However, with an integrated set of tools (software) having the appropriate software architecture, better and more powerful software can certainly be developed. In this way, the application range of CAPE methods and tools can be significantly increased.

Finally, the use of a systems approach means that the experience gained from one application area and/or industrial sector for one tool may be transferred to another application area and/or industry sector for the same tool or another class of tools. Software architecture will help to move the tools from one application area to another without too much extra effort, as in many cases, the knowledge structure is usually similar but the knowledge content is different. Having a flexible architecture that can handle different knowledge content within similar knowledge structures would reduce the time for software developers while at the same time, increase the number of applications and software users.

1.1.1.2 Computers and Software Technologies[1]

The computer and software industry has followed a pace of changes in the last decades, with the most recent being the universal deployment of the Internet for business and home activities.

Networks

It is not possible to consider a computer as a standalone device, except in very peculiar circumstances. Networks are everywhere, for both fixed and mobile systems: one can count on a link to the Internet or to mobile communication systems in almost any place of the world. For this reason, technologies that allow software to take advantage of networks are gaining dominant position: this is the case for *distribution architectures* such as Microsoft's DCOM and .NET, OMG's CORBA, Sun's Enterprise Java Beans; this is also the case for the XML and XML-based various standards for document and data sharing. The current trend is to propose representations, which allow business systems to discover and integrate *web services* components on the Internet on the fly, such as with the UDDI initiative, or the various WSxL languages for interoperating web services,

Machines

Current desktop and laptop computers have more computing power, more memory, and more disk space, than supercomputers used in the 1980s for the most demanding scientific applications. A PC with a Pentium IV processor clocked at 2GHz can be ten times more powerful than a Cray XMP, the supercomputing reference system back fifteen years ago. Clusters of PCs running under Linux or Windows NT give huge processing power at a reasonable price; some companies have replaced their corporate supercomputers by clusters of hundreds or thousands of low-cost PCs. Network bandwidth is growing at the same rate, with the *Ten-155 kernel* at 622 Mbits/sec. deployed over all European and developed countries, allowing to share complex data and information from distant locations. GRID computing, that is, the distribution of computation over the networks, is becoming a reality.

Programming languages

Programming languages evolve as well, with new languages appearing periodically, able to take advantages of the new hardware and operating systems. *Object-oriented languages* (*e.g.* C++, Smalltalk, Sun's Java, Microsoft's C#) almost completely replaced the traditional procedural code that was the standard ten years ago (*i.e.* Fortran and C). Developers of CAPE applications increasingly use sophisticated development tools such as Microsoft's Visual Studio or Sun's J2EE

[1] Editor's note: For an explanation of the acronyms, please see the "glossary of terms" section after Part VII.

toolkit. Source code is often generated from higher-level or abstract models and descriptions, and *component libraries* supplied with Application Programming Interfaces (APIs) are used for several simple and not-so-simple tasks (handling of lists and arrays, database handling, numerical processing, graphical user interface widgets, *etc.*). We cannot leave this list without adding the many languages used for the Internet, from the simple HTML static descriptions to the CGI scripts for database access, and to the Javascript and Active Server Pages dynamic systems, some of the most popular technologies in this field.

New Architectures

New Architectures have been proposed for the development and deployment of complex software. *Middleware* is playing an important role in these new architectures as it brings the "software glue" that puts the pieces together. Three-tiered architectures (with three layers: data, processing, interface) are standard, and n-tiered architectures with several interacting layers are proposed for more complex applications. Following this trend, the software industry designed *component-based architectures* such as the ones presented in chapter 4.1, in which middleware is acting as the "software glue" for putting pieces together. Among these component architectures, of course, we must mention:

- OMG's *CORBA* and its many technologies, from the IIOP software bus for interoperability of heterogeneous objects, to the high-level Meta Object Facility for analysis and design, through the IDL neutral Interface Description Language;
- Microsoft's *COM and COM+* component architecture, which allow Windows-based applications to interoperate, and the many development tools able to use this technology; this technology is further developed in the .NET architecture which uses the eXtensible Markup Language (XML) and the Standard Object Access Protocol (SOAP) as core communication tools;
- Sun's *Enterprise Java Beans* architecture, on top of the Java language, which proposes complex and flexible arrangements of components written in Java.
- Of course, XML itself, and its accompanying technologies (DTD, XSL, *etc.*) now has a crucial role in e-business and e-work infrastructures, gathered in what is now called EAI - Enterprise Application Integration;
- Agent and multi-agent architectures can provide opportunistic cognitive distributed processing and take advantage of the network; the last chapter of this book gives a few ideas on the development of such facilities;
- Other architectures for distribution of processing, using component technology and transaction mechanisms, which will become important when dealing with supply chain and e-commerce. We do not describe them

further in this book since there are no uses of such technologies in the applications considered.

User Interfaces

User interfaces made huge progress in recent years. In terms of visualisation, people do not speak of UIs anymore, but of GUIs, or *Graphical User Interfaces*. Invented by Xerox at their Palo Alto Research Center in the 1970s, GUIs involving desktops, windows, menus, mice etc. were made popular by Apple in the 1980s with the advent of the Macintosh, the first graphical personal computer. Since then, Unix workstations, Windows-based PCs, Personal Digital Assistants and other equipment (including some mobile phones, Linux boxes, *etc.*) were all equipped with graphical user interfaces. More advanced displays make use of 3D visualisation such as in "reality centers", "immersive rooms" and other *virtual reality* devices. On the other hand, interfaces used by humans to drive the computers followed a similar evolution, moving from keyboard input, to mice, touch-pads and touch-sensitive displays, voice-activated devices, virtual reality sensors, eye movement tracking systems etc. Development tools are available for all these technologies, from simple libraries with APIs to complete GUI development systems.

Methods, methodologies, algorithms

Methods, methodologies and algorithms for software development and scientific computing have been subject to step changes as well. Together with the development of new tools, new languages, new architectures, methodologies and methodological tools have been proposed, some of which gained enormous success in the recent years. In terms of *object-oriented software development*, we moved from a world of little know-how and of great confusion, to a world now dominated by the standard and widely accepted UML (Unified Modelling Language) notation and its accompanying design and analysis processes; *design patterns* which capture re-usable elements of object-oriented software design were developed and almost immediately exploited by the software community; a very abstract system such as OMG's Meta Object Facility, which represents software artefacts at the meta-meta level[2], was only understood by a handful of researchers ten years ago, but is now used in daily practice by software designers who exchange design models through the XMI (XML Metadata Interchange) language. In the algorithmic domain, algorithms for software and for scientific computing also followed a significant trend of evolution, allowing to tackle large problems that could not be solved in the 1980s. We talk about numerical algorithms, especially for solving and optimising large process models, in Section 3. Here, we mention other types of algorithms, such as the ones used for statistical analysis, data mining, machine learning or stochastic global

[2] Models of models of objects!

optimisation. In these fields as well, numerous improvements were proposed, such as neural networks for learning nonlinear behaviours, fuzzy logic for approximate reasoning, evolutionary (genetic) algorithms for global optimisation, among others[3].

In summary, computer and software technologies made such progresses in the last decades that we cannot approach CAPE software development the same way as we did even only ten years ago. This has significant consequences on the organisation of work for providers of CAPE tools. People need to be trained, new work processes must be put in place, and this takes even longer than developing the technologies themselves. But it is a necessary step, and we hope that this book can contribute to the process.

1.1.2 OBJECTIVES OF THIS BOOK

Scientific software, commercial or otherwise, is needed to solve CAPE related problems by industry/academia, for research & development, for education/training and many more. With increasing availability and use of computers in industry/academia, at work and at home, the use of the appropriate software together with their development and availability has become an important issue. There are increasing demands for CAPE software to be versatile, flexible, efficient, robust, reliable and many more. This means that the role of software architecture is also gaining increasing importance. The software architecture needs to reconcile the objectives of the software, the framework defined by the CAPE methods, the computational algorithms, the user needs and tools (other software) that help to develop the CAPE software. The objective of this book is to bring to the reader, the software side of the story with respect to computer aided process engineering.

The text below explains the contents of the book, which is divided into seven parts plus a glossary of terms, a subject index and an author index. The contents of each part are briefly described, followed by a road map for the reader in terms of "who should read what?", since all parts of the book may not be of interest to every reader. The reader is, however, encouraged to read everything in order to get a complete picture.

1.1.2.1 The content (themes, topics, summary of sections)

The book tries to give a rather exhaustive view of the themes and topics of relevance to CAPE applications over the lifecycle of processes. It addresses tools for design, development and operation of continuous, batch and hybrid processes,

[3] All of these deserve much attention and much more ink, but since they are not in the focus of this book, and however frustrating it is for the writer and for the reader, we only mention their existence. Another book is needed on this subject.

in the scope of CAPE as defined above. However it does not contain detailed material on molecular modelling and simulation, on computational fluid dynamics (CFD). These technologies are very relevant for the development and operation of processes but deserve more attention than only a chapter in a book on CAPE.

We wrote the book using our experience in the modelling of chemical, pharmaceutical, petrochemical and oil processes (both refining and offshore facilities). Some of the content can be applied to other process industries *e.g.* polymers, agrochemicals, waste and water treatment, since some of the tools share the same concepts. It would me more difficult to transpose the concepts in quite different industries such as food, steel manufacturing, pulp and paper, thermal and nuclear although some general ideas will probably apply.

The book starts with an assessment of the needs of the process industries in terms of CAPE software. The three chapters in Part II, VISIONS AND NEEDS, address these needs and include a general view, a view on the specifics of batch processes, and the needs for lifecycle modelling, since it is now well accepted that models should follow the lifecycle of processes from conceptual design to operation and de-commissioning. Following this, we show what our medium term vision is, taking into account the needs and what facilities the computer industry will offer. Part II is mainly written by industrial users who are experienced with CAPE tools and their utilisation in large and smaller corporations.

Part III, FRAMEWORK FOR CAPE TOOLS, introduces current architectures for CAPE, as proposed by leading scientists and vendors of CAPE software. It gives concrete examples on how these frameworks can help to solve problems expressed in Part II. Its scope includes modelling, optimisation, monitoring, for continuous, discrete and hybrid systems. It presents the state of the art of representations, algorithms and architectures.

Part IV, MAKING CAPE TOOLS, show what are the current tools (component software; objects, middleware, databases, XML..) and methods (data modelling, UML ..) used by the CAPE software industry to develop the new frameworks and tells about current standards projects and their status. Through examples such as CAPE-OPEN, Plant Data XML, PI-STEP, the reader will get an insight on how these standards are being developed and implemented. This part is mostly written by software specialists and experienced CAPE tools providers.

Part V, USING CAPE TOOLS, highlights, through illustrative examples, the use and/or application of the relevant frameworks (of Part III) and methods, tools & standards (presented in Part IV) to the solution of interesting CAPE problems. It also presents what are the current computer aided tools (process simulators, databases, design, *etc.*) that are currently available in industry and academia.

The chapters in this part has been written by contributors from academia and industry, all known as world experts in their field.

Part VI, NEW FRONTIERS, shows where we are going, through new technologies and new potential areas of applications that make use of the advances in CAPE and software technologies: *e.g.* agents, web-based systems, integrated approaches, process monitoring and fault diagnosis systems. It starts by discussing new business models made possible by current technologies (collaboration frameworks, e-Work and e-Business models). Some of the developments in this part are already available but not widely used yet; some still involve significant research before being considered for applications. Specialists at the forefront of CAPE research have written the chapters of this part.

Part VII, CASE STUDIES, presents the modern use of CAPE tools and architectures on representative process examples. The designers and developers of these examples have written this part.

1.1.2.2 Who should read what?

We hope that the book will be of interest to a wide variety of readers. Here we try to guide you through the book following a rough classification, with an aim to help in making the best of it quickly, depending on your interest.

Managers of Process Engineering and related activities will be mostly interested in the vision and needs Part II, in order to understand and share what the medium and long term goals are; they will probably skip most of Parts III and IV and find additional interest in Part V where we present applications of modern CAPE tools. Enough curiosity will lead them to read bits and pieces of Part VI, at least the Emerging Business Models chapter (6.5). They will also appreciate the two examples of Part VII (chapters 7.1 and 7.2). *Suggested order of reading: II, VII, V, VI.*

Users of computer-aided process engineering software (mostly process engineers) will easily understand the Vision and Needs part (Part II), as they probably share many of the ideas expressed there. They will learn a lot by reading Part III as an introduction to the methods and concepts implemented in the tools that they are using for their daily activities. They will also appreciate the part on standards in Part IV, since it shows what facilities will be shortly available with the future versions of their favourite tools. Some of the applications shown in Part V and some of the New Frontiers developments will be worth learning about. Finally, we expect that they will enjoy the two examples in Part VII. *Suggested order of reading: III, II, IV, V, VI.*

Managers of CAPE software development projects will, as all others, be interested in Part II, Vision and Needs. They probably know already most of Part III, and will find more "meat" in Part IV where we present methods and tools that their development team will be using, and standards upon which they will base some of their future activities. As for Part III, they will probably be already familiar with application examples in Part V. They should definitely read Part VI for the new ideas for products that it brings. *Suggested order of reading: II, IV, VI.*

Developers of CAPE software will need to understand in depth the concepts articulated in Part III, as it will be their final responsibility, to implement these concepts in software. They are certainly familiar with some of Part IV, but probably not at the same level of detail for all chapters of this part, so we suggest that they spend some time on it. Then they could skip most of Part V and spend some time on Part VI, in the same spirit as for Part III, that is, to understand the concepts before implementing them. Part VII will contribute to their motivation for further developments. *Suggested order of reading: III, VI, IV, VII, II.*

Researchers in CAPE and in Process Engineering will probably like the book as a whole; they will probably be experts in some of the chapters and will be even able to comment, criticize or expand some of the ideas expressed, but obviously not for all topics and issues. It is difficult for us to recommend a selection, as it will mainly depend on their specialty. *Suggested order of reading: parts of II, VI, II, III, IV, V, and VII.*

Software specialists from other domains will take the book as an example of developments in a major application sector. They will need to gain some familiarity with the domain from application examples from Parts VII and V, completed by the vision offered in Part II. Then, they will be able to appreciate the technical developments as presented in Part IV and the future trends of Part VI. *Suggested order of reading: VII, V, II, IV, VI.*

If you do not belong to any of these categories, please let us know. We appreciate your feedback!

1.1.3 REFERENCES

Bansal, V., Pistikopoulos, E. N., Perkins. J. D., A unified framework for flexibility analysis and design of nonlinear systems via parametric programming, *Computer Aided Chemical Engineering*, Vol 9, Elsevier, (2001), pp961-966.

Bayer, B., Weidenhaupt, K., Jarke, M., Marquardt, W., A flowsheet centered architecture for conceptual design, *Computer Aided Chemical Engineering*, Vol 9, Elsevier, (2001), pp345-350.

Bühner, C., Schembecker, G., Reactor selection and design for heterogeneous reaction systems, *Computer Aided Chemical Engineering*, Vol 9, Elsevier, (2001), pp357-362.
Harper, P. M., Gani, R., A multistep and multilevel approach for computer aided molecular design, Computers and Chemical Engineering, 24 (2-7), (2001), pp677-683.
Ng, K. M., A multiscale-multifaceted approach to process synthesis and development, *Computer Aided Chemical Engineering*, Vol 9, Elsevier, (2001), pp41-54.
Okada, H., Shirao, T., New chemical process economic analysis methods, *Computer Aided Chemical Engineering*, Vol 9, Elsevier, (2001), pp1059-1064.
Pantelides, C. C., New challenges and opportunities for process modelling, *Computer Aided Chemical Engineering*, Vol 9, Elsevier, (2001), pp15-26.
Perris T. and Bolton L. (2001), CAPE-21 Report, available from www.cape-21.ucl.org.uk

Web-sites:
CO-LaN web site, http://www.colan.org

Microsoft COM Web-Site http://www.microsoft.com/com

OMG CORBA Web-Site http://www.omg.org

CAPE-WP Web-Site http://www.cape-wp.kt.dtu.dk

Part II: Visions & Needs for CAPE Tools

The objective of this part is to provide an assessment of the needs of the process industries in terms of CAPE software. There are three chapters in this part and they address these needs and include a general view, a view on the specifics of batch processes, and the needs for lifecycle modelling, since it is now well accepted that models should follow the lifecycle of processes from conceptual design to operation and de-commissioning. Together with an assessment of the needs, each chapter also highlights the medium term vision, taking into account the needs and what facilities the computer industry will offer. Industrial users who are experienced with CAPE tools and their utilisation in large and smaller corporations are the principal contributors to the chapters of this part.

Chapter 2.1 written by Banks et al. begins by reviewing the environment in which modern CAPE professionals operate. It then considers the potential benefits available from the application of advanced simulation techniques in the different activity sectors and the issues involved in actually realising these benefits. Finally, it looks at some of the implications of the adoption of these techniques for the individuals and organisations involved.

In terms of CAPE, both the 'batch' and 'speciality' characteristics influence the needs for system software as well as the interfaces required between the components of CAPE software. Chapter 2.2 (Malik and Puigjaner) considers these issues in further details in terms of structural characteristics of batch and speciality chemicals industries, management requirements, examples of industries, information needed for decision making and finally, characteristics needed for the supporting software.

Chapter 2.3 (Okada and Shirao) identifies the "life cycle" needs of process industries who are going to use CAPE software as well as other tools during process modelling, simulation, synthesis and process design. The chapter discusses the issue of sharing of information between different tools and the role of the data manager during project execution as they influence the user's lifecycle expectations or needs, such as interfacing with different design tools to keep information flow among different disciplines during not only the process design phase which should be fulfilled by CAPE but also across the lifecycle.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

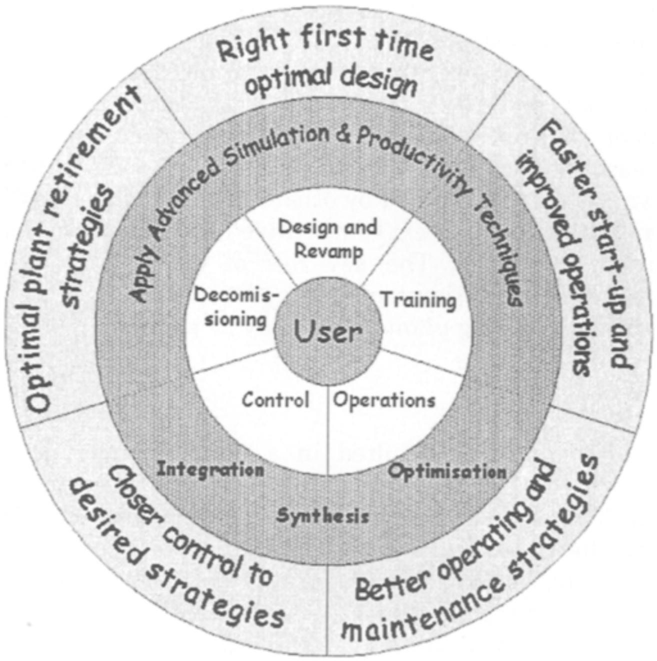
Chapter 2.1: General User Needs for CAPE

P. Banks, K. Irons, M. R. Woodman & B. J. Stenhouse

2.1.1 OVERALL OBJECTIVE

CAPE has great potential to provide business benefits, but, in reality, only a portion of these benefits is currently being delivered. We look at some of the issues that cause this shortfall and establish the unsatisfied user needs implied by each issue. We will be concentrating on industrial users, since they are generally the most demanding, and we will highlight their needs, rather than suggest specific solutions.

The diagram below summarises the scope of the chapter:


Right first time optimal design
Faster start-up and improved operations
Better operating and maintenance strategies
Closer control to desired strategies
Optimal plant retirement strategies
Apply Advanced Simulation & Productivity Techniques
Integration
Synthesis
Optimisation
Design and Revamp
Training
Operations
Control
Decomis-sioning
User

It shows the activities involved in the lifecycle of a plant and indicates the full benefits that are available in each, if a way can be found through the advanced simulation technology layer in between. We assume that users are already receiving benefits from applying standard simulation techniques in each activity sector, so we will focus on the user needs that arise in seeking to move to the new benefits promised by advanced techniques.

The chapter begins by reviewing the environment in which modern CAPE professionals operate. It then considers the potential benefits available from the application of advanced simulation techniques in the different activity sectors and the issues involved in actually realising these benefits. Finally, it looks at some of the implications of the adoption of these techniques for the individuals and organisations involved.

2.1.2 BUSINESS ENVIRONMENT ISSUES

The modern corporation has changed significantly, and continues to change rapidly, from the common model of the 1980s and before. With this ever-increasing rate of change in the corporate environment, the ability to adapt quickly is paramount. Arie DeGeus, formerly head of planning at Royal Dutch Shell, has said "The ability to learn faster than your competitors is your organization's only sustainable competitive advantage." Not quality, not service, not technology, not price, not marketing, not patents, but the ability to adapt more quickly than the competition. This is true across all industry categories. It does not imply that speed is more important than other factors (quality, cost, ease of use, safety, environment, *etc.*), but rather that any competitive advantage can be copied by others in time. The rate at which an organisation can implement its next competitive leap will determine how long it can sustain its advantage. The key role of CAPE is in allowing the organisation to develop and implement improvements in the design and operation of its manufacturing plants as quickly as possible.

2.1.2.1 The Need for Speed

Mergers & acquisitions have resulted in a major change in the mix of organisations participating in the bulk chemical, speciality chemical, petroleum, petrochemical, pharmaceutical and agricultural chemical sectors. Many companies that were major players only a few years ago have either merged with others, split into separate entities, undergone major divestitures, totally changed focus, or disappeared completely. New companies have been formed, alliances and joint ventures abound, and new business models are being tested.

The result of all this has been a rapidly changing mix of competitors, intense competition and rapidly changing business priorities. With all of the shifts in the structure of firms in a given market sector, companies find themselves constantly facing new challenges from organisations that didn't exist a short time ago. Many companies focus on market leadership as their strategic intent, which requires them to compete on a number of levels: price, quality, service, technology, product functionality, etc. Companies cannot only react to this competition, they must determine how to remain a step ahead in at least some of the elements. As each organisation looks for ways to gain competitive advantage, many paths are pursued. The ability of a firm to act quickly can make the difference between market leadership and failure. This reaction speed must be supported in all activities of the organisation, including CAPE.

CAPE practitioners have generally been seen as highly skilled technical specialists, having built expertise over time and multiple project experiences. However, with the factors noted above, there are increasing pressures on this group of professionals. With mergers, acquisitions, joint ventures, etc., staff changes more are far more common than was the case. The effect of these changes is often less continuity in the technical community, whether this means people moving between business units (and therefore between technology areas) or between companies. In addition, technical professionals are frequently asked to participate in diverse activities that are not core to their CAPE role. While this results in a welcome broadening of perspective for each individual, it also means that they become less-frequent users of CAPE tools, resulting in a reduction of their specific CAPE expertise.

With increasing focus on business level objectives, CAPE technical staff can find themselves with less focus on a single company agenda. At the least, there can be difficulties in simultaneously meeting business unit objectives (*e.g.* minimum capital investment for the next new plant or plant expansion) and corporate goals (*e.g.* minimum energy consumption, site integration, emission reductions, *etc.*). This issue is complicated further when CAPE activities involve contractors, whether they are contract engineers working with company staff, or when entire projects are contracted. While project objectives are always discussed as part of the negotiation, contracting, and project management process, that is no guarantee that those objectives are translated into desired individual behaviours when the CAPE work is actually done.

The combined effect of rapid change and competitive pressures means an increased emphasis on capital efficiency and pressure to build and operate plants that reflect a shorter lifecycle for the products they manufacture. These plants must also aid the goals of minimised working capital, asset base, lost production and maintenance budgets, as well as maximum on-line time. The goal here is minimum total cost of ownership. At the same time, there is pressure to squeeze the maximum capability from existing assets, which can actually decrease capital efficiency because older assets are used when newer technology could be more efficient. This is often done when a company seeks to avoid capital expenditure in an effort to increase its Return On Net Assets or Return On Investment.

All of these objectives must be met with lean organisations and stretched staff, who are working under all the constraints and external forces noted here. One manifestation of this changing environment is the widespread practice of outsourcing all or part of a firm's CAPE activities. The outsourcing drivers are to reduce the costs of having a staff of internal experts, to be able to respond to changes in demand for CAPE activities, and to tap into increasingly specialised expertise available from firms who are "full time" CAPE practitioners.

A final set of pressures on the CAPE community comes in the environment, health, & safety arena. Process industry firms are faced with ever more challenging environmental goals, driven by regulatory, corporate policy, societal, and industry pressures. The growth of a litigious society brings another element of pressure on the CAPE professional to deliver process designs that are economic, flexible, capable of producing the required product mix, and socially responsible. It also adds the need to be able to provide evidence that due diligence was, in fact, practiced.

In this environment, it is clear that CAPE professionals will place heavy demands on computing and CAPE software capabilities, as they strive to deliver the speed of response being asked of them. We will look at these demands, later in this chapter.

2.1.2.2 Business Environment Needs

CAPE activities need to be integrated with the larger business enterprise. Easy incorporation of costing modules that are directly linked to the process simulation will allow rapid convergence to business-optimal solutions, not just engineering-optimal solutions. Such cost estimators need to be able to access both general cost data sets as well as company or technology specific cost information.

Along with cost estimation capability, the larger arena of project economics must be readily integrated with process modelling. Changes in the process design can have significant impact on overall project economics, and the ability to see that impact quickly will result in more flexible and cost effective plants. To be able to run multiple project economic scenarios with different plant design bases quickly, process simulators must easily and intuitively share data with project economics packages. Standard interfaces between process and economic models will give firms the ability to fully understand their options, quickly achieving optimum solutions.

Long-term cost of ownership for any given process design can be heavily influenced by maintenance costs. Maintenance cost data bases must be readily available to process and project economic estimators, and so allow the process design optimisation CAPE effort to truly identify lowest cost solutions. For the same reasons, reliability modelling capability must be linked to the overall project economic model, again using standard interface specifications.

Business leadership needs to design company work processes with an understanding of the strengths and limitations of CAPE. This includes integrating CAPE capabilities into contracting and outsourcing strategies, maintaining enough skill and experience in their staff to be "smart buyers" of third party CAPE services, and recognising the limitations of process models. This means that these staff members must be active, skilled practitioners, otherwise they will be unable to fully understand the capabilities and limitations of CAPE tools and the organisations using them.

Mergers and acquisitions impose a constant requirement for integration of the CAPE activities of the new corporate entities that result. The adoption of open interface standards and a “plug and play” architecture for process simulators, commercial software components and proprietary in-house software, would allow this integration to take place rapidly.

2.1.3 COMPUTING ENVIRONMENT ISSUES

All would agree that the last 20 years have been a period of ever-greater focus on IT (Information Technology). IT was once the province of a select few operators of large-scale computers, but it now pervades every aspect of corporate and academic life. IT is now so all-encompassing that it is assumed to be a key leveraging capability, and obviously so in CAPE. Where engineers once were restricted to remote access to mainframes, computers are now generally available and typically reside on the desktop. When corporate leaders see how much they invest in IT, and when they constantly hear of the impact that they should be getting for that investment, they have high expectations for CAPE.

The tremendous increases in computing power and availability have undoubtedly created an environment in which these expectations can be met. Models can now be created with sufficient scope and detail to have significant business impact. However, such models are inevitably of great complexity; so complex, perhaps, that they can actually lead to computer-aided mistakes! These complex systems can be handled easily by state-of-the-art computing equipment, but that equipment is not always available to each member of the CAPE community. This can create an imbalance between the perceived need for model complexity and the actual computing resources at hand for a broad base of end users. Many in the technical specialist community hold the view that their corporate IT organisations don't take these needs into account when acquiring and distributing computing resources. For example, future needs for computing power may well be met with parallel processing capability and web sharing, but these are issues that have not been addressed by most corporate IT organisations, or even the CAPE community.

2.1.3.1 Computing Environment Needs

In many cases, computing resource needs for CAPE exceed the "standard desktop" computing capability as defined by IT organisations. Corporate IT experts need to work closely with CAPE professionals in defining a computer platform and capability that will effective support CAPE activities. This does not normally mean something beyond a high-end desktop computer, but usually that computer must have much more RAM, higher clock speed, and a larger hard drive than are required for typical corporate computing. However, in future it may mean parallel processing capability.

Corporate computer networks and Intranets must be capable of transferring the large files associated with CAPE so that engineers can work collaboratively on CAPE projects.

2.1.4 CAPE ENVIRONMENT ISSUES

Beginning in the 1980s, powerful simulation/calculation systems have become generally available to the Process Engineering community. Put another way, these process simulators have made it possible to practice CAPE on a wide scale. Without these tools, engineers had to develop, debug, and operate their own computer models. It is only with the advent of process modelling tools that are user-friendly and broadly applicable that CAPE has grown to be within reach of every practitioner. Heretofore, CAPE was restricted to a very small number of engineers working on highly specialised, long time frame projects. Costly and lengthy modelling efforts were only undertaken for the most demanding projects. Because many sophisticated algorithms have now been incorporated in widely available, industry standard CAPE tools, they are allowing practitioners to operate at the full level of their engineering skills. The CAPE professional can make use of state-of-the-art computing and modelling tools that would be far too difficult and time consuming to implement, if the engineer had to write the equations and the computer software "from scratch." Now an engineer can easily perform calculations without having to painstakingly review the methodology and procedures, making routine application of very sophisticated computations the norm.

One significant challenge for any activity involving computers is that large amounts of information can be easily generated and stored, but not easily turned into knowledge. Such “lost knowledge” must be made accessible in order to avoid unnecessary reinvention and the associated waste of scarce resources. The ability to convert information into knowledge and to retrieve it in a timely and efficient manor is a function of an organisations’ work processes and of the CAPE software used.

To deal with the challenge of increasing complexity and a more stringent regulatory framework, many organisations have concluded that the ability to develop an audit trail is an important component of their CAPE software. However, auditing has its own problems of resource needs and time requirements, and can be seen as a luxury in an environment where cost and speed are ever more important. It is the other pressures of health, safety and environment, personal liability and increased outsourcing that make the case for documenting the audit details behind the “user friendly icons” of a process simulator model. . It is particularly important in an open, plug and play environment, such as that offered by CAPE-OPEN (CO) standards (see www.cape-open.org for more information). Here, the modeller can make use of multiple software components from diverse sources, so that a complex model may not contain simply the process simulator supplier’s unit operation or data base components.

Audit documentation is especially key when a model is used for optimisation, as this places maximum stress on the process model, and any anomalies must be understood and corrected. Further, if a project suffers from any lack of continuity of staff, or if a process model is used over a relatively long period of time, the rationale behind design and modelling decisions can be lost. CAPE tools that can easily capture the rationale for decisions and design bases, making documentation “automatic,” bring significant value to an organisation and its projects. Without appropriate documentation, mistakes can easily be made out of ignorance of the reasons for previous CAPE decisions.

Given the time pressures that are inherent in CAPE activities, CAPE tools with an intuitive user interface can reduce the time required to build, document, and optimise a model, as well as the time to learn to use the tool. Such systems obviously reduce the need for specialised training, but at the same time, training will always be required to improve the productivity and quality of work for CAPE practitioners. In the rush to bring new engineers up to speed on a project, or convert an organisation to a newer version or a different brand of CAPE tool, training sometimes manifests itself as a combination of "sink or swim" and "ask the person down the hall". This approach has many limitations compared to effective training programs and tutorials.

2.1.4.1 CAPE Environment Needs

Model sharing methods are needed so that the effort put into developing and refining models can be leveraged across a project team. This means making the model available to plant, process R&D, process design, detailed design and maintenance engineers, as well as plant operators. While the needs of each of these groups are different, putting the results of the process model in their hands will result in significant improvements in operating plant performance.

Facilitating model sharing requires intuitive user interfaces, the ability to store and retrieve modelling results, the ability to easily interpret those results through graphical presentation, the ability to easily see and understand the input data and underlying assumptions, and the ability to easily compare multiple model cases. Visualisation of modelling data and results, as well as the model's structure is required to best accomplish this goal.

"Automatic" documentation of model structure, methodology, and the choices made during model assembly and use is required. This will allow and require users to communicate their thinking by documenting the decision process in model assembly and use. This documentation then becomes the basis for any required conformance audits and for ensuring that subsequent model users and developers understand how and why the process model evolved. A similar process should also be used to document the creation of the final process design case.

Self-documentation requirements extend further when a model is used for optimisation, whether off-line or on-line. Since optimisation involves a number of trade-offs, documenting both the decisions and the rationale behind them is key in capturing the value of the optimisation work. The likelihood that a process model will be used by many CAPE practitioners over several years makes such documentation all the more critical. The CAPE-OPEN environment emphasises this need, as software components will have multiple versions over time, and therefore must be documented fully with each model implementation and upgrade.

The need for intuitive interfaces is obvious, but they do not reduce the need for effective training. Whether this consists of on-line tutorials, classroom sessions, imbedded help commands, or a combination of all three, rapid and low cost training of everyone involved with model development and usage is essential. The training must also be specific to the use, with CAPE experts requiring a very different experience than plant engineers, or operators using a training simulator.

2.1.5 POTENTIAL BENEFITS

There are many published examples that demonstrate that the appropriate application of CAPE tools can deliver substantial benefits across the process industries. These examples cover the complete range of CAPE tools applied in all the sectors identified in the diagram at the beginning of this chapter. However, accurate numbers for the size of these benefits are hard to obtain and even more difficult to attribute.

The main commercial simulation vendors routinely publish success stories for their software, supported by anecdotal evidence from the relevant contractor, or operating company, as to how the benefit was delivered. These are published as a sales and marketing tool, so are obviously written with that in mind, but, even so, they provide a useful indication of the type and scale of benefits that can be achieved.

However, there are many potential benefits that can be obtained through the application of CAPE that currently are not routinely captured. To illustrate where these potential benefits can be achieved, it is necessary to look at the process plant life cycle from process concept through to operations and beyond. Some of the benefits described are already being obtained in a few companies, but not uniformly across the process industries.

2.1.5.1 Design and Revamp

It is possible to deliver process designs that are cheaper to build or, more exactly, capital efficient - *i.e.* that requires less capital, or use the same capital more effectively. The potential for CAPE tools here is:

- To enable leaner and more agile project teams to use con-current engineering, in place of the traditional sequential design process.
- To simplify the flowsheet for the plant (and hence remove equipment, reducing the capital cost) through the use of formal process synthesis techniques.

A recent publication by Shell International Chemicals and the Process Design Centre (Harmsen, J *et. al.*, AIChE Symposium Series No 323, Volume 96, 2000) describes capital cost reductions of up to 50% through the application of process synthesis tools.

The same process synthesis tools can also be used to produce process designs with improved operating costs. There is a common assumption that reducing capital costs will lead to increased operating costs, but the same paper quotes operating cost savings of up to 80%. Looking at the total annualised cost (which combines both capital and operating costs), the savings quoted range from 20 to 60%.

Formal optimisation methods and dynamic simulation are two available techniques within CAPE that are not routinely used, but which offer significant potential to deliver process designs which are safe to operate and easy to control. Optimisation at the design stage offers at least two benefits:

- A formal approach to assessing the financial operational impact of process constraints against the cost of removing them by equipment substitution.
- A framework within which to assess the robustness of the design to uncertainty in the basic design parameters and economic assumptions.

Dynamic simulation offers the ability to:

- Integrate process and control system design, enabling the design to be produced faster and with greater assurance that the process and control schemes are optimised.
- Examine the dynamic behaviour of the plant and proposed control scheme, thus ensuring the process will operate as intended and is controllable.

Finally, all the techniques described above can also be used to produce process designs that minimise impact on the environment. This can be achieved not only through the reduction in energy usage, but also by giving designs which minimise emissions and can cope with upset conditions in the most environmentally-friendly way.

2.1.5.2 Training

The main area of application for CAPE during plant start-up is through the provision of a training simulator. This enables the operators and plant engineers to be trained in the operation of the process before it is commissioned. It is then available during start-up, for troubleshooting and to improve the understanding of the dynamics of the process.

The training simulator subsequently becomes a tool for training new operators before they start work on the actual plant, and to improve the capability of the existing operators in situations of abnormal operations.

Although training simulators are becoming more common, they are still not made available for every new plant. Furthermore, the most effective way of generating a training simulator is to base it on the models used during the design process. This is definitely not currently standard practice, with the result that a great deal of rework is needed to generate an entirely new model for the training simulator.

2.1.5.3 Operations

Once a process plant is operating there are many areas in which CAPE can offer benefits.

Process troubleshooting is perhaps the most common application. Given the low process engineering manning levels present on most process plants today, this activity is most often associated with helping operations staff investigate causes and possible solutions, when the plant is unable to deliver the performance expected of it by the management team. Here, off-line models are used to understand the process and the way in which the operating equipment is under performing. Frequently this involves simulation followed by detailed equipment rating calculations. Most industrial companies routinely use CAPE in this way and are undoubtedly generating benefits from the process. Significantly, some companies are now using process models in a more pro-active way to routinely monitor equipment performance. The objective is to avoid problems before they occur or, at worst, advise on an appropriate intervention strategy.

Going beyond performance monitoring and operational troubleshooting, there is an opportunity to use process models to optimise plant performance to generate increased profit for the business. The benefits of on-line model-based optimisation have been robustly demonstrated in a number of areas, for example ethylene plant optimisation. Here, technologies from a number of suppliers have received wide application, often linked with proprietary models developed by the operating company.

Typically, on-line optimisation is used on plants, which either have multiple products, or where throughput is very large, so that there are large benefits to be obtained by a very accurate optimisation of the plant around the current operating point. Models are usually simplified, but continuously tuned to the plant data, thus ensuring high accuracy in the current region of operation.

Such on-line optimisation can be complemented in many cases by rigorous off-line optimisation. Here the models need to be detailed and based on first principles. All equipment constraints need to be explicitly included. Off-line optimisation can be used to make step-out changes in the operation of the plant, outside the current range of operation of the process. For example:

> An unpublished Dow project involved the consistent application of an in-house optimisation tool to a series of distillation columns. This has resulted in an average 18% reduction in distillation energy requirement and a comparable increase in tower capacity. These towers represent a broad range of product types including speciality chemicals and commodity hydrocarbons.

In BP, a 5% improvement in throughput was obtained at an oil production site by optimisation of operational set-points. In addition, at the same site, a significant summer production shortfall, caused by air cooler limitations, was eliminated. In both cases, the operating strategies proposed by the optimisation were counter to normal practice, but, after the event, fell firmly into the category of "Why didn't I think of that?"

Finally, there are many plants for which neither off-line nor on-line optimisation is necessary, typically because they are single feed / single product processes. In these cases, the creation of high fidelity, rigorous off-line models is often sufficient to provide significant process understanding and thus improve the operation of the plant. An unpublished example from the Chemicals Stream of BP describes the use of such models, which resulted in $0.65 million operational savings per annum, whilst also reducing effluent by up to 50%.

2.1.5.4 Control

Although control is not strictly CAPE, there are many areas where CAPE can be used to assist the control engineer. The use of dynamic simulation to aid the control system design has already been described, but there are other synergies to be obtained, especially in the area of Advanced Control.

The first of these is the re-use of the Process Engineering models in the Advanced Control algorithms. Such re-use is almost certainly not "as-is", because of the speed requirements for the models used in control, but simplified models can be generated based on the Process Engineering models.

The second is the use of the Process Engineering models during the plant trials ("step-tests") required to tune the Advanced Control. This can either be by providing information as to which plant trials are required, thus avoiding the need for unnecessary trials and minimising the risk of upset conditions during the trials, or even by replacing some of the trials with off-line modelling, in certain circumstances.

2.1.5.5 Decommissioning

The optimal decommissioning of plant is an area, often over-looked, where CAPE tools can contribute significantly. The scenario is very common in the oil industry, during the later stages of an oil or gas field's life. Production rates begin to fall away from the design levels of the processing plant, so the equipment items are required to operate below full capacity.

The most important issues concern the operation of rotating machinery: compressors and gas turbine power generators. Low plant throughputs either result in operation away from the peak efficiency point, or else require recycle, with its associated increased energy consumption. Both compressor and gas turbine efficiencies fall dramatically at operations significantly away from the design point.

Where multiple trains exist, equipment can be decommissioned and load transferred to the remaining machines. This can result in significant benefits from reduced energy demand and environmental impact, which are key issues for oil and gas stabilisation plants, particularly with a gas shortfall.

A closely related scenario occurs in oil stabilisation plants, where hydrocarbon fluids from a large number of producing fields are processed centrally. This can result in gas rates and compression requirements varying throughout the year, according to the profile of production units feeding the pipeline system. Appropriate decommissioning or reinstatement of equipment offers significant benefits, but requires careful planning and accurate simulation to allow operations staff to respond in a timely manner.

Of course, decommissioning may sometimes have disadvantages. For example, there may be a loss of processing flexibility, which may affect the ability to respond to plant upsets and equipment failures. Also safety must be assessed, as with any process change. These complex issues all have to be included in the optimal decommissioning study, which is really only feasible with accurate, flexible process models.

These techniques are now beginning to be applied. Optimisation of plant decommissioning has generated benefits for at least two on-shore oil and gas stabilisation processes. Analysis, in these cases, was performed by a combination of process engineering knowledge and the use of good, validated process models, coupled with an element of formal non-linear optimisation. However, the problem should really suit a robust mixed-integer, non-linear optimisation approach, since it has clearly defined limits. It is, in fact, a particular type of process synthesis problem.

Although this topic is of most interest in the oil and gas sector, it could also be applicable elsewhere, if, for example, throughputs are reduced, or there is a cyclical demand.

2.1.6 DELIVERY OF ACTUAL BENEFITS

2.1.6.1 Overview

At its most fundamental level, an engineer uses process simulation to understand his process better. With the aid of this increased understanding the engineer can add value by improving the engineering design, by responding to operational problems in a more informed way and by establishing alternative modes of operation that offer improved performance.

The opportunity to add value from process simulation applies during both the design phases of a project and during the operational phase of the associated plant. It is worth discussing the issues arising from these two phases separately.

2.1.6.2 Benefits during Plant Design

Design moves through a number of phases from synthesis or conceptual design, where models are traditionally relatively simple and the correct "big picture" solution is identified, through to detailed design where flowsheet topology is fixed and exact equipment performance is determined. To facilitate this process the underlying synthesis and simulation software needs to be sufficiently accessible to allow use by non-specialist technology

R&D or project development personnel, but able to grow with increasing modelling fidelity into a full-scale detailed design model. In addition the resulting detailed design model should be able to be taken forward by the operational team as the "operational model," capable of accurately mimicking the control and operation of the plant after start-up.

In summary:

> We need intuitive flexibility in modelling software and presentation to the user, so that the representation of the plant can be easily refined as the design process proceeds.

Process Synthesis

Process synthesis technology has been developing as general simulation package capabilities and hardware performance have improved. However, a general process synthesis solver has not yet materialised and the full mixed-integer, non-linear programming problem that comprises process flowsheet design optimisation, has remained largely a problem for academic study. Certain niche area synthesis programs have matured, however, and found their way into the commercial simulator market.

The desire for a generalised process synthesis package may be unrealistic, given the range of fundamental issues to be addressed at this stage. For example, in upstream oil processing, the key question is the robustness of the design concept to the inherent uncertainty present in the design information; for a new process on a chemicals plant, the key issue may be how best to optimise design, given the integration of reactor and associated downstream separation system. As an additional complication, process synthesis often involves the transfer of proprietary technology from academic research into industry.

The general industry need is for the design engineer to be able to use detailed unit operation models at the heart of a robust synthesis process, such that the resulting flowsheets are valid and can be taken on as the initial plant design basis.

Initial synthesis studies can sometimes be performed with simple process models, as long as more detailed models can be substituted as the process develops. However, in other cases, especially in the chemicals industry, rigorous models, often representing new processes, are required from the start. Both of these situations demonstrate a clear need for easy plug and play within the synthesis activity itself.

In summary:

> The use of computer-aided process synthesis tools is not widespread, but emerging systems need to incorporate easy plug and play capabilities. This will allow the transition from simple conceptual models to rigorous representations for final flowsheet definition.

Detailed Design

In the ideal design process the simulation model lives and grows alongside the development of the actual operating plant. It is true to say that this "life-cycle vision" for simulation has been an aspiration of many operating companies for a number of years. However, in most cases it has remained more of an aspiration than a reality, despite some significant advances in the under-lying technologies. The reasons for this are manifold, but relate primarily to the ease of development and maintenance of the models and the varying relationships involved through the asset life-cycle. Another factor is that the design process often involves a number of different people, each of whom is responsible for only one piece of the design, so that once that is completed, they move away to the next project.

The vision is only likely to happen if the simulation model can be developed without the need for continual re-working. A prime example of this concerns the roles of steady-state and dynamic modelling. Typically the initial models developed of a new process are steady-state to allow the basic design parameters and overall economics of the system to be assessed. However, transient effects can have a key role in the optimum process design; the earlier in the design process these can be assessed, the better will be the resulting design. This parallel role of steady-state and dynamic simulation in the design process is greatly facilitated by easy movement from the steady-state environment to dynamics and vice-versa. Typically, these modelling tasks are conducted by different engineers, but the underlying requirement is for them to share a common model and representation of the plant. Management commitment to this process is essential for it to take place.

Many engineers are involved with process modelling during a plant's life cycle. Operating companies have eventual ownership of the plant design and responsibility for safe operation. However, contracting companies are commonly involved in the detailed design specification of the plant. They are also increasingly involved in the conceptual design phases, as operating companies minimise internal engineering resources on projects. This means the operating company must be able to rely not only on the abilities of the staff in the contracting companies, but also on the design procedures used and the accuracy and reliability of the recommended engineering design tools. In particular thermodynamic methods, physical property correlations and modelling assumptions need to be well documented and sufficiently transparent to allow a high degree of technical assurance to be easily established.

Design projects typically deliver a design package consisting of documents – PFDs, PIDs, data sheets – that define the process design. These are normally delivered electronically as a collection of unconnected design documents. A much more effective project deliverable for the end-user/operator would be a process design package in electronic format containing process graphics, audit trails and automatic cross-referencing of design data.

Design inevitably involves trying to optimise more than one objective. A good example of this is the requirement to design a plant that is the cheapest to build, but also the most environmentally-friendly and profitable to operate. Historically this has involved compromise based on engineering ingenuity and the examination of a large number of design cases. Mathematical optimisation at the design stages offers an improvement over this situation, as sensitivity analysis can be used. However, with conventional optimisation techniques, integer decisions can not easily be handled and multi-objective optimisation can be handled only by expressing all but one of the objectives as a constraint on the search space. Although useful, this approach can lead to sub-optimal designs and there is a clear business need for true multi-objective optimisation, if this can be delivered in a usable fashion.

In summary:

> The concept of a model for life is now quite feasible, but will require management commitment to deliver it.
> The design package can be delivered electronically and should include the model, supporting documentation and decision audit trail. This is increasingly important as in-house engineering staff numbers diminish and contractors deliver designs and projects.
> Practical optimisation tools are available to improve design quality, but real problems often require multi-objective optimisation, which is still to be achieved.

2.1.6.3 Benefits during the Operational Phase

Once a plant is operating, benefits will only be delivered when an operator changes control set-points on the plant and moves it to a new and better operating position. Achieving this introduces a number of new issues related to model development and usage.

Model Fidelity

Firstly, it should be possible to take the engineering design model forward as the operational support model for the plant. Typically this will require some re-configuration of the model, since design product specifications may well need to be replaced by new specifications representing operational set-points, in order to mimic the way the plant is operated and controlled.

Before an operations department will take action based on a model's predictions, they require confidence that the model is a valid representation of the plant. Usually this confidence is built up over time, based on repeated evidence of the model's capabilities to predict and explain. In order to reproduce operating plant behaviour, in particular, operation of the plant as plant capacities are maximised, the fidelity of the process simulation model generally needs to be increased. A particular example of this is to take account of equipment hydraulic capacity limitations that would not normally appear in the design model. In general, operational optimisation models require all aspects of the mechanical design of the plant to be included, as it is not possible to predict beforehand which constraints will be active at the optimum solution. The implementation of such third-party capacity models requires the simulation program to be open, with standardised interfaces, to minimise software development requirements and to simplify maintenance and upgrade capabilities.

It is worth stating at this stage, although it should be obvious to most people, that high fidelity modelling must be underpinned by the availability of high fidelity data. Detailed models require validation against plant performance and this is reliant on the availability of good process measurements linked into parameter estimation in the model. In fact, redundancy of data measurements on the plant is extremely useful for model validation, which is an iterative process, in practice. Initially the model is validated against data taken from the plant, but once accepted as a faithful representation of the operating plant, the model becomes capable of validating the plant data, by highlighting where measurements are likely to be in error. Even off-line operational models require convenient access to plant data. For instance, every time a strategic optimisation is run, up-to-date information on equipment performance must be fed into the model, so that effects like exchanger fouling are correctly represented. If this is not easily achieved, the model predictions are likely to be compromised.

In summary:

> A design model must be extended significantly for use in operational support. General unit operation models must be enhanced to turn them into models of the particular pieces of equipment that have been

installed. An open plug and play architecture is essential for this to be done with the minimum of effort.
Process models must have easy access to accurate plant data, so that they can reflect actual equipment performance at the time of model execution.

Using models to update operational guidelines

The techniques of offline simulation modelling and optimisation can be used to develop improved operator guidelines on how to utilise spare degrees of freedom to maximise plant performance. We can also use them here to illustrate the way in which user needs extend beyond the achievement of the central technical prediction, in order to deliver actual benefits.

Typically, operator guidelines are developed from previous experience coupled with knowledge of the design basis of the plant. Use of a high-fidelity optimisation model can be very valuable to generate new insights into plant operation and to challenge the accepted thinking of the best way to operate. This is because an optimisation model can look at the simultaneous interactions of many degrees of freedom on the plant. This can vary from 4 or 5 degrees of freedom on a typical chemical plant to 20 to 30 on a large refinery unit. Human operators can typically only cope with 2 to 3 variables at a time. However, operations departments will normally be justifiably resistant to implementing counter-intuitive solutions.

A good example of this situation came during a model-based optimisation study of a complex crude/vacuum distillation unit in the UK, shown in Figure 1. It distils under vacuum the heavy atmospheric residue from 3 crude oil distillation towers (COD1, COD2, COD3) to recover further distillates to feed refinery conversion processes. It is generally regarded as a straightforward process, which is operated at the lowest possible pressure with base steam set as a ratio to vacuum residue (TAR) yield. However, in this study, model predictions suggested that optimum distillate recovery on the vacuum unit would be achieved at an operating pressure some way above the current operating point. This was clearly counter to the normal operational philosophy of the unit and was greeted initially with considerable scepticism. It was not until a convincing engineering explanation of the prediction was developed, that plant trials were authorised and the benefit demonstrated.

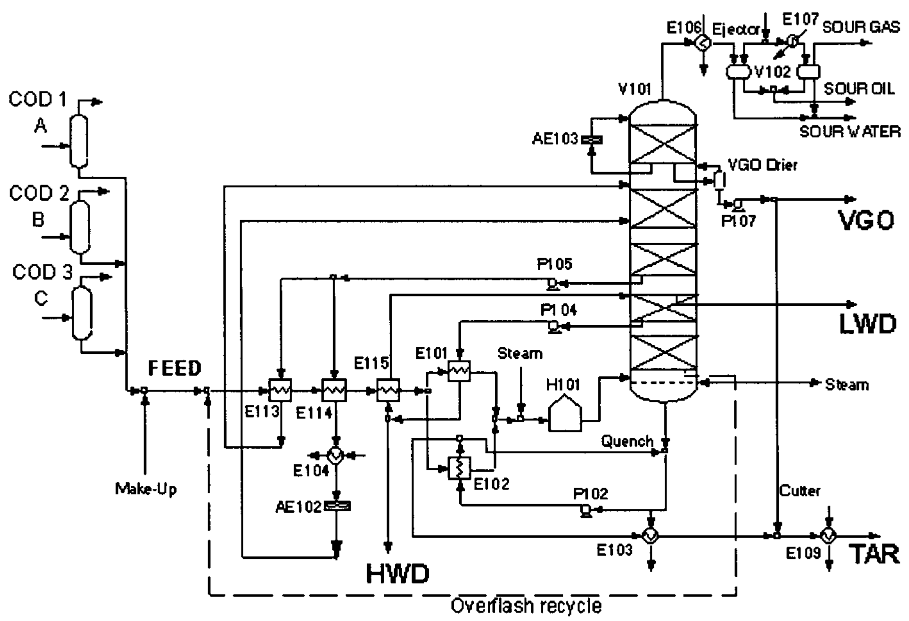


Figure 1. Flow diagram of a complex crude/vacuum distillation unit

The eventual explanation was quite simple: Vapour induced flooding of the vacuum tower packed beds was resulting in instability and loss of stable operation at high vacuum. However, increased distillate yields could be obtained at higher pressure by a corresponding increase in base stripping steam, but without initiating column flooding. Although this was all quite straightforward to understand, after the event, it took considerable tenacity and ingenuity on the part of the engineers involved to turn the mathematical messages from the optimiser into process engineering insights. This meant that a long time elapsed between making the original prediction and implementing it on the plant, during which considerable benefit was lost. Although it did not happen in this case, there is always the further danger that the longer this time delay is, the more likely it is that something else will happen and the prediction will never get implemented. The key factor that led to implementation was that the operations engineers were fully involved with the CAPE engineers throughout. They understood the capabilities of the model and had confidence in its predictions.

However, this is still not enough. Recommendations from an optimisation, such as those above, can only be implemented on a plant that operators have under good steady control and that they have the ability and confidence to move away from the normal operating region. This introduces a number of new issues, not least the safety impact of moving the plant to a new operating point, particularly where the new point is outside of the operator's comfort zone or region of past experience. It is worth stating that the specific insight that led to the benefit described above was only true for this particular tower with its interaction of feedstock, performance requirements and design constraints. The conclusion ceased to be correct later in the tower's life, when it was revamped to replace the packed beds that caused the hydraulic limitation. Different physical limitations then constrained the plant, which is why it is important to include all aspects of the mechanical design in the optimisation model.

To summarise, we have shown that, although the primary requirement for the generation of improved operating strategies is the availability of a reliable optimisation system and a good model of the plant, there are several other user needs that must be satisfied to deliver the final outcome. The most important of these additional user needs, is the ability to extract a convincing process engineering explanation for any counter-intuitive predictions, quickly and systematically. A robust and fast optimisation model will almost certainly be central to this process, as will some form of visualisation software. Optimisation algorithms gather significant information about the feasible operating region and the problem curvature, both on route to the solution and at the optimum point. The challenge is to extract this mathematical information in a form that provides physical answers to the question "why is this operating point better than all the others?" This question then rapidly leads to further questions about how much better the optimum solution is over the next, say, half-dozen best solutions and which of these steady-state solutions is easiest for the operators to implement from the current operating position. A lot of benefit can be lost in moving from the current operating position to the new one, and some degrees of freedom are more difficult to move than others. Again, tools to help answer these questions quickly and systematically, will increase the number of these studies that are carried through to actual benefit.

Other additional needs are links to the plant database and detailed heat exchanger rating programs that enable fouling to be estimated and the performance of all the exchangers in the simulation to be correctly represented, in a convenient and consistent manner with a minimum of manual intervention.

This has been a detailed look at only one possible application of advanced simulation technology. The same holistic approach must be taken to the others to ensure that all the steps needed to deliver the target benefit are first identified and then facilities put in place to enable them to be carried out conveniently. If this is not done, the benefits will almost certainly not be delivered.

Thus:

> New counter-intuitive operating strategies can be derived from off-line optimising models. However, the benefits will only be delivered, if tools exist to enable the engineering rationale and safety of the predictions to be easily determined. This is true of any optimisation application.
> Links are required to the plant database and detailed equipment rating programs.

2.1.6.4 Linking models to Advanced Control

Modern, multi-variable controllers (MVC's) are capable of handling large-dimensionality problems for control and optimisation, based on linear models derived by plant step tests. This technology is essentially associated with strategy enforcement. A number of the degrees of freedom available to the controller are tied to regulatory control; the remainder can be used for optimisation. However, due to the linear nature of its underlying model, the controller generally requires a pre-defined strategy to be implemented for these additional degrees of freedom, to ensure correct plant optimisation. An off-line simulation/optimisation model, such as the one in the example above, can be used in this role to define the strategy for the controller.

A key element here for developing acceptance of Advanced Control by operations staff, would be the ability to utilise an off-line, non-linear dynamic model to mimic the action of the MVC on the plant and to test and refine the developed strategy logic. In addition, the step test derivation of the linear model for the controller is a time consuming activity, which could be minimised if a basic linear model could be readily created from the off-line dynamic model.

In summary:

> The implementation of Advanced Control underlines the need for straightforward transition from steady-state to dynamic plant models. It also indicates that the ability to be able to create a linear model from the off-line dynamic model would help minimise the step tests needed to create the controller model.

2.1.6.5 On-line model-based parameter estimation and optimisation

Putting a model on-line for parameter estimation and optimisation, requires a modelling architecture that allows flexible problem definition together with robust and fast solution. Historically, an equation-based architecture has been considered to be best suited for this task. However, if such models are put on-line to optimise the process automatically, then the same models should be usable off-line for engineering studies and wider-ranging, strategic optimisations. It is in this role that equation-based models have fallen down, in the past, in their ability to solve when not close to the solution, their general robustness and the difficulty of diagnosis if they fail. The important thing, for an operating company, is that only one model-based representation of the plant should exist and need to be maintained. This means there should be one set of equations and assumptions, which could be solved in different ways in different circumstances. The method of solution should be transparent to the user, so long as it works correctly and reliably.

In summary:

> The model description, in terms of equations and assumptions, should be separated from the solution method. This would simplify model maintenance and allow the most appropriate mathematical techniques to be used for the different functions in the lifecycle of the plant and model.

2.1.6.6 Operator training systems

It should be possible to take a dynamic model developed at the design stages of a project as the basis for an operator training system. Since these are used to develop the skills of new operators and to provide training in the management of unexpected situations, the models should represent the plant's performance as closely as possible. Once the operator trainer is commissioned, the underlying model should be the same as that used off-line for process control support studies and operational troubleshooting. Apart from minimising model re-working and development, the ability to use a common modelling platform across the range of dynamic modelling applications simplifies the training requirements for operational and process support staff. This has often not happened in the past, because the reliability and speed of solution of rigorous models has not been suitable for operator training. These are issues that must be tackled in the future, to enable our model for life requirement to be realised. It will probably require a combination of hardware and software development.

In summary:

> Operator training systems need high fidelity, non-linear dynamic modelling facilities that solve quickly and reliably.

2.1.6.7 Equipment Maintenance

Traditionally, equipment maintenance is planned on the basis of regular maintenance intervals during continuous operation, supported, in some cases, by equipment performance monitoring. The availability of high fidelity plant data together with high-fidelity plant models introduces the possibility of moving to a regime where maintenance is planned on the basis of how hard a piece of equipment has been operated, as well as for how long. Well-validated process models are key to achieving this, as they can provide a wealth of process information to add to the limited amount of available measured data. Again, this information will need to be presented in way that is intuitively clear to a maintenance engineer, but which is based on the same underlying process models that are being used to operate the plant.

2.1.6.8 Use of tools by Process Engineering Specialists.

The delivery of operational benefits can best be achieved by engineers whose primary responsibility and skill lies in utilisation and operation of the plant, rather than the advanced use of simulation tools. Their role could be business planning, based on a knowledge of what the operating plant is capable of delivering, or short-term operational optimisation. However, to make good use of a complex asset simulation model, these engineers will require the simulation technology to be packaged in a way that allows them to focus on their business tasks, rather than on the underlying technology. The user interface required for each business user of simulation will, in general, be different and related primarily to fitting simulation seamlessly into their work tasks. This implies a certain degree of openness and plug-and play capability for the interfaces themselves, as well as the technical components.

In summary:

> The delivery of CAPE tools to process engineering specialists requires the same flexibility and openness in user-interface systems as in modelling systems. This will allow the underlying plant model to be presented to each different category of user in a way that is tailored to that category.

2.1.7 INDIVIDUAL RESPONSIBILITY

In most countries, professional activities are increasingly governed by legislation, such as, for example, the UK Health and Safety at Work Act. Although the details may differ from one country to another, what this basically means is that whatever the sophistication of the computer programs you may use, the responsibility for the decisions you finally make remains firmly with you. We therefore need to examine the requirements that this implies for common CAPE tools, such as flowsheet simulators and equipment design tools, as well as considering if it suggests new tools that need to be provided.

In the case of common tools, the guidelines developed by the UK Institution of Chemical Engineers provide a useful discussion of the issues involved in the responsible use of CAPE tools (see http://cape.icheme.org/capeinfo/Guidelines/Good_Practice_Section_Menu_Page.html). These guidelines cover the checks necessary before using the results of CAPE tools and hence imply the facilities that the tools need to provide to allow this to happen conveniently. These facilities can be summarised as follows:

Thermophysical data – this is critically important to any simulation and must be carefully selected. Openness is important so that the origin of the data, its range of applicability, quality of fitted model and extrapolation properties can all be assessed. This must be easy to do, as must the application of Sensitivity Analysis.

Engineering model – it must be clear if there are any restrictions on applicability of the model. If these are missing or well hidden, it will be a major source of potential danger.

Input checking – it must be simple to check input data to ensure the right problem is being solved.

Results checking – it must be simple to check mass balances at overall, component and atomic levels, depending on the context. It must also be simple to check energy balances. Error messages must be clear and unambiguous. Convergence criteria must be clear, as must the status of convergence. However, reliable convergence is even more important.

The only certainty is that everything is uncertain – this underlines the importance of Sensitivity Analysis. It must be simple to apply to the overall solution, so that you can easily bound the problem, determine the stability of the solution and assess the allocation of design margins, if appropriate.

In addition to these general points, there is a further point associated with advanced techniques, especially those involving optimisation. The power of these techniques is that they are multi-dimensional and, hence, likely to provide counter-intuitive solutions. This opens up access to benefits that would otherwise be unknown and unattainable, but only if the user is comfortable about implementing the solution. As we have discussed earlier, this means that the software must not only provide the solution, it must also provide a means to understand the rationale behind the solution. In the context of this discussion of individual responsibility, we need to focus on the safety implications. Whilst, for example, the new operating point described in "Using models to update operational guidelines" may be justified in theoretical terms, it may not be a safe or stable operating point. This must be checked first and then a safe trajectory from the current point to the new one has to be established. In other words:

Projects that take advantage of counter-intuitive, multi-dimensional solutions must be provided with tools and procedures to assess the safety of operation in unfamiliar regions.

In the business environment described earlier, one of the trends that we have noted is the likelihood that more changes of personnel will happen in the course of a project and especially in the life of a plant, than used to be the case. In addition, these people are likely to work for a wider variety of companies than before, as outsourcing becomes more prevalent. One of the consequences of this is that good "organic" documentation becomes essential for any model/program/design that is likely to be used or modified by more than one person. "Organic" documentation, in this context, means documentation that is intimately associated with the model/program/design and that is updated whenever anything changes. It will almost certainly be different from formal project documentation, which tends to be rather more static. It will, for example, record all changes and the reasons for the changes and it will log the date and the person who made the change. Without this information it is difficult to judge, at a later date, whether it is safe to make changes, with the result that it is often regarded as easier to start again, or just to leave things as they are. Neither of these options is necessarily the best for the business.

Of course, it has long been the ambition of project managers to have such a record available - and it has long been one of the first things to go when the budget and timescales get tight. What has changed now is that the need has become greater, for reasons discussed above, and the universal use of computers has made it possible to envisage tools that could make it a practical proposition. The sort of tool required will be a challenge to software designers. It must be available on line in the background so that as, for example, a model is being developed, or a plant being designed, the decision points are recognised, the decision captured and the user prompted for the rationale behind the decision. This must be a seamless activity, since any complication or awkwardness will almost certainly result in lack of use. It, therefore, implies that it can work in an intuitive way with other tools, such as flowsheet simulators, without affecting their user interfaces, in both capture and retrieval modes. This provides us with the final user needs bullet point of this section:

CAPE tools must provide the ability to record and retrieve decision points and their rationale in an intuitive way during the creation or modification of a model/program/design.

2.1.8 ORGANISATIONAL IMPLICATIONS

The most important organisational success factor in the application of CAPE is management understanding and commitment. If management provide clear guidelines on the degree of risk that is acceptable in the pursuit of CAPE benefits, provide the resources and training needed to follow the guidelines and maintain the policy in the medium term, then there is an excellent chance that a large percentage of these benefits will be obtained. However, a common scenario is for there to be a message about the pursuit of technical excellence, but a stronger message about the need for cost reduction and downsizing. Faced with the certainty of reducing costs by cutting resources, the possibility of benefits from difficult and new CAPE activities looks unattractive to many middle managers. This is hardly surprising, but again not necessarily in the best interests of the business. Clearly, clarity of purpose and good communication become even more important, as outsourcing increases.

A second success factor needed to implement the issues discussed in the preceding sections, is the management promotion of work practices that allow individual engineers to exercise their individual responsibility effectively. An important part of this is to encourage good networks of CAPE practitioners across a corporation, or, where outsourcing is involved, across the relevant technical community, to encourage best practice and the reinforcement of skills. Thus the user needs in this section are:

Management should communicate a clear and consistent corporate policy on acceptable risk in the pursuit of CAPE benefits.
Organisations should provide tools and promote work practices that assist individual engineers to exercise their professional responsibility in pursuit of these CAPE benefits.

2.1.9 SUMMARY

It is clear from the above discussion that the delivery of benefits from the advanced use of CAPE tools is as much about the work process and the people who execute it, as it is about technology. The technology must be available and it must work, but that in itself is not a complete and sufficient condition for success. Most of the user needs identified above arise from the need to ensure that the people in the delivery loop are kept fully informed and involved. Uncertainties create barriers that extend the elapsed time of a CAPE project and increase the likelihood that the project will not be undertaken.

The current business climate is one of lean organisations, heavily-loaded staff, frequent personnel changes, shorter timescales, increasing health, safety and environmental regulation and a generally litigious society. In this environment, it is clear that all categories of users require a full understanding of the capabilities and limitations of the tools they are using, as well as the meaning of the results. This is particularly true with applications that involve optimisation techniques, since these promise new, counter intuitive solutions that can lie outside the normal comfort zone. User interfaces must be easily tailored to each category of user, so that information from the same underlying model can be presented to all in ways that are intuitively clear. Given the cost and complexity of a modern process model, the concept of a "model for life" is economically attractive. However, to provide the necessary user understanding, such a model will need to carry around its life history in a convenient form. This means recording major decision points, so that the model can be used and adapted with confidence, as the plant proceeds through its lifecycle.

To build and operate a plant using process modelling, in today's short timescales, requires an appropriate infrastructure to provide:

- Easy access to all required data, for instance:
- Cost data for design
- Comprehensive plant measurements for operations
- Effective process control systems to implement new strategies
- IT facilities capable of solving CAPE applications in a reasonable time
- A plug and play, open standard architecture for process simulation, such as that provided by CAPE-OPEN.
- Corporate clarity on the level of technical risk tolerated and consistent management support for staff to pursue benefits within this risk scenario.

If all of the above are present, benefits from advanced CAPE tools will certainly be delivered. As more of these elements are missing, the benefits achieved will drop sharply. In the current pressurised business environment, it does not take much inconvenience to separate a busy engineer from a CAPE application and hence its benefits.

So, to deliver benefits from advanced CAPE tools, you must first focus on the end users; identify ALL the steps that must be taken; make sure that each step can be easily achieved and that it flows seamlessly into the next one; and then provide the appropriate intuitively-packaged tools. Discontinuities or confusion will rapidly destroy the process, no matter how good the underlying technology may be. All this must be done within a supportive corporate culture. It's not easy, but the benefits are real and substantial - and if you don't do it, one of your competitors undoubtedly will.

This Page Intentionally Left Blank

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 2.2: User Needs in Batch and Specialties Chemical Processes

T. I. Malik & L. Puigjaner

2.2.1 INTRODUCTION

2.2.1.1 Distinction between 'batch processes' and 'specialty products'

In expressing these needs, we should distinguish between what is meant by a 'batch' process and 'specialties' chemical processes and clarify where both terms apply. By definition, a batch process occurs in a finite time span, having an initial start condition, a path of transformation (usually involving material and energy transfers) and a final state. This is a definition based on type of processing and does not necessarily reveal the classification of the product itself. On the other hand, the word 'specialty' refers to the type of product being manufactured irrespective of process used to make it. The name, 'specialty' arises from the special functional properties that these products bring when added to other 'bulk' materials. Examples are chemicals that impart special fragrances or flavours to their substrates or modify the rheological behaviour of (bulk) fuel (*e.g.* diesel).

2.2.1.2 Different degrees

Of course, there are different degrees that can be applied to both terms. A batch process can be totally batch or be part batch (where some aspect is continuous *e.g.* the flow of feeds in case of fed-batch). There can also be a mix of batch and continuous in different parts of a flowsheet. Often, for producing bulk chemicals, the reaction and primary (fluid-fluid) separations may be carried out in continuous units whereas the secondary (fluid-solid) separation (*e.g.* in centrifuges), filling lines, palleting and packing may be carried out in batch units (sometimes at such high frequency that we get the illusion of continuity). On-line or off-line compositional measurements will be batch for both types of manufacturing operations. In terms of degree of 'specialty', some materials are used in very minute and dilute quantities whereas others are the main component in a given application albeit with tailored, 'special' properties.

2.2.1.3 When are batch processes used

The choice between a batch process or a continuous process for a given product depends upon many factors including scale of operation, type of transformation required, numbers of different products required from the same plant and economics. Usually these factors conspire to make specialty processes mainly batch based as they do to make them high value added.

2.2.1.4 Influence on CAPE needs

In terms of CAPE, both the 'batch' and 'specialty' characteristic influence the needs for system software as well as the interfaces required between the components of CAPE software. This chapter will consider these issues in further detail.

2.2.2 STRUCTURAL CHARACTERISTICS OF BATCH & SPECIALITIES INDUSTRIES

2.2.2.1 Value of materials processed

The relative value of the materials processed vs. the plant used is much higher for batch and specialties industries as compared to bulk chemical industries. Whereas the bulk commodity chemicals are made in large quantities and usually will be the only product made on a dedicated plant, often there are many related but distinct specialty products that are made using the same equipment.

2.2.2.2 Flexible infrastructure and safety

Thus the batch process plant needs to be sufficiently flexible to be able to accommodate the variation of products as well as be able to cleanly separate one product from the other. The safety characteristics (*e.g.* relief capacity) need to cater for all the processes that are carried out. The sequence in which the products are made can be important from the point of view of safety and product quality (*e.g.* contamination).

2.2.2.3 Mix of plant and equipment

Often, the processes have evolved over a long period of time and the equipment available has a large span in terms of age. (Usually the bulk of a continuous plant is of the same age). Often the products are made in a number of countries (at least for companies that are internationally established) and it is not uncommon for there to be little standardisation between these. Thus the efficiencies can vary quite significantly between different processes.

2.2.2.4 Life span of products

Often the life span of a product is less than in the commodity markets as improved products with more special properties become available through research and development. Thus, a company may have several new products in the pipeline that it will make using the present equipment. Product scale-up is quite an issue and often requires significant experimentation and pilot scale work. These are bottlenecks to the development of new marketable products.

2.2.2.5 Levels of investment made

The investments made in plant and equipment are usually an order of magnitude less than those in the bulk industries. The capital is spent in a more even manner unlike in the bulk companies where a few projects each year may take the bulk of investment.

2.2.2.6 Interaction with regulatory authorities

As there are many more new products and as these are often used in contact with man, the interactions with and the work of the regulatory authorities is much more important for the introduction and safety classification of the new products.

2.2.3 MANAGEMENT REQUIREMENTS

2.2.3.1 Cost efficiency

From the point of view of management, it is desired to make the most efficient use of the resources available at the least cost and to provide products of the right quality to customers on demand (in a highly responsive manner with safety, security and environmental considerations). In order to achieve these goals, it will be required to tract all the materials and the status of any given order at a given time. In order to minimise costs, the supply chain needs to be optimised with inventories minimised. Actions such as maintenance can be implemented at times of relatively lower demand on the equipment. From the point of view of the product cross-contamination, there should be a compatibility matrix that says what products can follow a previous batch.

2.2.3.2 Balance between batch frequency and batch size

There can possibly be some trade-off between the sizes of the batches used and the batch cycle time although there will be a minimum batch cycle time even for the smallest of batches. The optimal balance needs to be found as well as the optimal trajectories through which the batches are passed.

2.2.3.3 Scheduling

The best and optimal schedules in which to make the products need to be available on a daily basis. It should be possible to adjust these on the basis of changes in demand or constraints e.g. as a result of equipment failure.

2.2.3.4 Effective R&D Capability

As these industries are often characterised by rapidly shifting product portfolio, it is part of the management requirement to have an effective capability to develop and scale-up new products, to be able to use the existing equipment effectively for making new products.

2.2.3.5 Standardisation of best practice

Given forecasting and longer-term market demand, the management would like to know what investments in which factories are the best ones to make and indeed if new factories should be started. To what extent should production be independent or inter-dependent between the factories, what strategy should be used for longer-term standardisation of best practice between the factories?

As far as possible, management would like to have the information and make decisions with the minimum expenditure of resources. This implies as much reliance on existing information, data and models as possible in order to minimise expenditure and avoid delays.

2.2.4 EXAMPLES OF INDUSTRIES

2.2.4.1 Relative size of the specialties sector

The total size of the chemical industry in year 2000 is estimated to be about \$1130 Billions. A little more than half of this is for basics and intermediate chemicals. The size of the specialty products is about \$330 Billions out of the total and the areas in which ICI operates are shown in Figure 1 below:

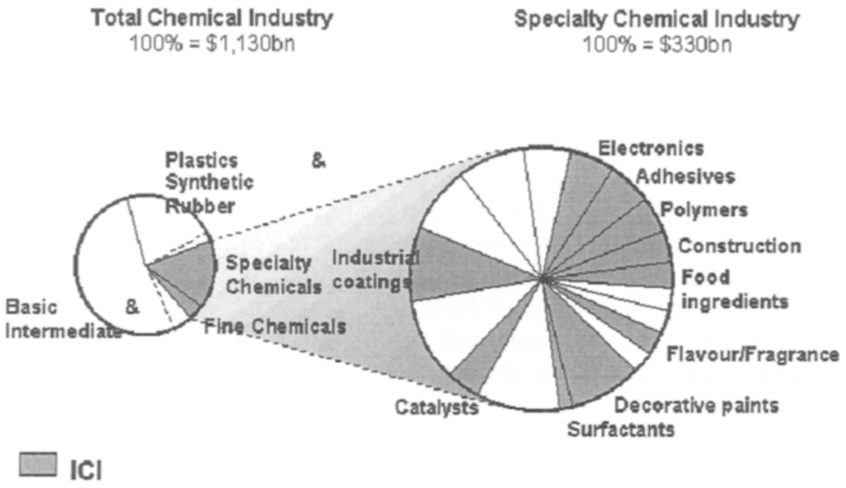


Figure 1. The Global Chemical Industry, Specialty sector and ICI Operations Year 2000 (Original source ICI Presentation Pack 2000)

2.2.4.2 Example - Description of catalyst making process

Here, for illustration an example of a catalyst making process is given as many as 150 different products can be manufactured on the same plant.

The making of catalysts is quite individual in that no two catalysts are made in exactly the same way but certain operations are common. In the 'wet process' the basic liquid and solid components are mixed together using precision measurements. Impurities are removed through filtration and drying where purified solids are dried in high temperature ovens up to 1500 °C. Following this the right shape of the catalyst is obtained in machines similar to those used in the pharmaceuticals industry to make tablets. Shapes vary from thin tiny solid pellets to larger complex geometric shapes. Finally the pellets are packed.

2.2.4.3 Examples of other specialty products requiring reaction chemistry

Examples of products that are made with reaction chemistry taking place in batch reactors include manufacture of synthetic polymers, e.g. manufacture of resin to be used in making paints, manufacture of surfactants, carrying out of the esterification reactions. Sometimes the batch reactions are combined with batch separation units *e.g.* in esterification reactions. Batch separation techniques (distillation or other) are used for example in extracting flavours from natural products.

2.2.4.4 Other Examples of Specialty Products

Further examples include adhesives, emulsifiers, dispersants *e.g.* for mineral oils, de-watering or flocculation agents, biocides *e.g.* for use in pigment slurry industry. There are products to control deposit formation and control in internal combustion engines, wear reducing additives, additives for diesel *e.g.* Cetane improvers, cold flow improvers and lubricity improvers. There are additives for Gasoline e.g. anti knock agents and anti valve seat recession. There are additives for lubricants *e.g.* anti-oxidants, viscosity modifiers, friction modifiers, detergents.

Other examples include the manufacture of anti-foam or de-foaming agents, inks, inhibitors and hydrophilicity agents, the manufacture of automatic gear lubricants, automatic transmission fluids, gelling agents for greases, tackiness agents. The extraction of starch from natural sources such as corn is carried out at quite large scale but the process has significant batch elements.

2.2.4.5 Examples of batch processes carried out in bulk industries

The manufacture of pigments for paints are usually batch including the case for TiO2 which is a bulk product but at least some of the process is carried out in the batch mode. Most development work for continuous process industries is also initially carried out in batch laboratory experiments before continuous pilot plants may be used. Some processes used in continuous plants are cyclical *e.g.* pressure swing adsorption and in terms of process modelling these will lead to some of the similar issues as conventional batch processes *e.g.* connections between steady state models and non steady state models.

2.2.5 INFORMATION FOR DECISION MAKING

2.2.5.1 Decision levels

In order to respond to new forces on the competitive landscape, manufacturing companies are incorporating the Computer Integrated Manufacturing (CIM) model to meet today's market needs.

The levels of decision for decision-making of CIM architectures are described in the Purdue Reference Model represented in Figure 2.

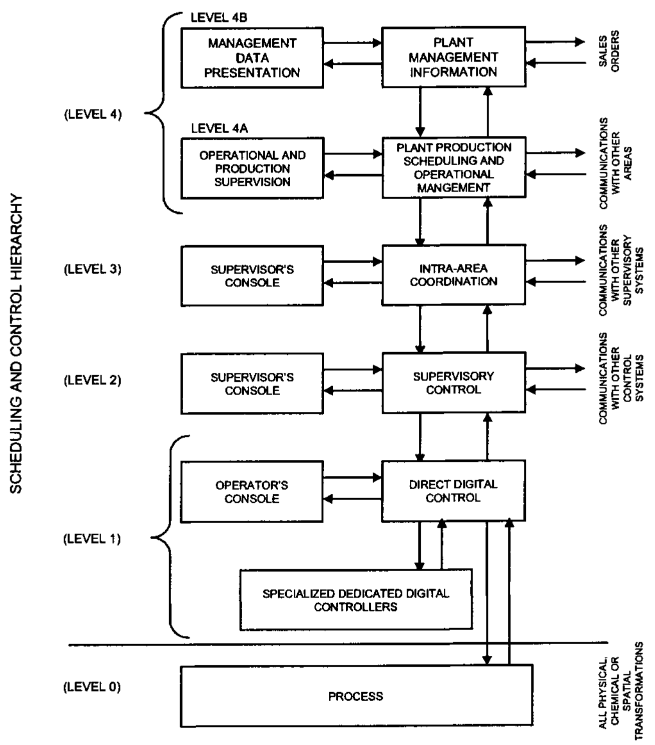

SCHEDULING AND CONTROL HIERARCHY
(LEVEL 4)
LEVEL 4B
MANAGEMENT DATA PRESENTATION
PLANT MANAGEMENT INFORMATION
SALES ORDERS
LEVEL 4A
OPERATIONAL AND PRODUCTION SUPERVISION
PLANT PRODUCTION SCHEDULING AND OPERATIONAL MANGEMENT
COMMUNICATIONS WITH OTHER AREAS
(LEVEL 3)
SUPERVISOR'S CONSOLE
INTRA-AREA COORDINATION
COMMUNICATIONS WITH OTHER SUPERVISORY SYSTEMS
(LEVEL 2)
SUPERVISOR'S CONSOLE
SUPERVISORY CONTROL
COMMUNICATIONS WITH OTHER CONTROL SYSTEMS
(LEVEL 1)
OPERATOR'S CONSOLE
DIRECT DIGITAL CONTROL
SPECIALIZED DEDICATED DIGITAL CONTROLLERS
(LEVEL 0)
PROCESS
ALL PHYSICAL, CHEMICAL OR SPATIAL TRANSFORMATIONS

Figure 2. A hierarchical computer control system structure for an industrial plant. From ISA-dS95.01-1999, Enterprise-Control System Integration. Part 1: Models and Terminology.

2.2.5.2 Here and now decisions/ wait and see decisions

One of the most widely used techniques for decision making under uncertainty is two-stage stochastic programming. In this technique, the decision variables are grouped in to two sets. The *first-stage* variables correspond to those decisions that need to be made prior to resolution of uncertainty ("here and now" decisions). Subsequently, based on these decisions and the realization of the random events the *second stage* decisions are made subject to the constraints of the *recourse* problem. Production decisions, because of their significant lead times, may be contemplated in a here and now decisions scenario. Otherwise, supply-chain decisions can be postponed on the basis of the production decisions and the realization of the demand (wait and see).

2.2.5.3 Decision making and uncertainty

Uncertainty analysis might be incorporated at the different levels of decision-making to improve the probability of performing the expected goals. Preventive maintenance tasks may also be introduced to compensate the use of facilities with low reliability indexes.

As for batch processes, the production schedule has to satisfy the production requirements under certain constraints, and, optimising an objective function usually based on the expected plant profitability. Preventive maintenance increases the plant reliability and, as a consequence, the production robustness. Therefore, batch processes require simultaneous maintenance and production scheduling activity.

Basically, the evaluation of robustness of a schedule is based on the reliability of the equipment unit assigned and of the possibility of finding an existing alternative unit in the case that the unit initially assigned to a task becomes unavailable during the schedule execution.

2.2.6 REQUIRED CHARACTERISTICS ON THE SUPPORTING SOFTWARE

2.2.6.1 CAPE Functionality Required – Medium Term Vision

There is a significant potential to improvements through systematic CAPE applications in the batch and specialty industries. However, due to the complexity and characteristics of these industries, the applications realised thus far are not as extensive as those in bulk and continuous processing industries. Some of the factors that have prevented applications can be circumvented through open interfaces as these potentially allow the collective functionality of several tools to be applied to given problems. Step changes to value generation can be obtained through application of CAPE in the following areas as well as effective, seamless integration between them:

- Modelling of batch/specialties processes.
- Scheduling/planning of batch/specialties processes.
- Information and data handling on batch/specialties processes.

Each one of the above areas is discussed below although the major emphasis is on the modelling part.

2.2.6.2 Modelling of Batch/Specialties Processes

Specific modelling needs arise both due to the nature of the process (being unsteady) and the characteristics and properties of specialty products that are handled.

Use of block-modular packages

Given sufficiently long time scale so the variation of frequency of occurrence of batches is masked, batch process flow sheets can be modelled with block-modular, steady state packages. While such models do not reveal detailed time behaviour within a single batch, they can give valuable material and energy balance information. However, provisions are required within the package to recognise that the plant may be utilised to make several products. The frequency at which batches of a given product are made may vary; indeed different products are included if there is interest in the same.

Interfaces for batch process models

Firstly, the modelling package needs a capability to recognise if the process that is being modelled is batch (both when time varying behaviour or time averaged behaviour are considered). Further, additional functions can be activated and the information about the batch stored in the correct manner for passing across interfaces. The type of information that could be valuable (if available) would be the time varying inputs to the batch, the material and energy balances, the average size of the batch, the frequency of the batch, the utilisation of named equipment for the batches. The package should recognise whether detailed behaviour of a single batch or time-averaged behaviour over several batches is being addressed.

Effective interfacing within a package

Some block modular packages contain batch units in their libraries, thus enabling time dependent behaviour to be modelled within a unit while time averaged values are passed to the rest of the flowsheet. An example of this is when a batch reactor is used connected to a continuous process. The user interfaces in such connected simulators need to improve to more clearly distinguish between time dependent and time averaged results. Also, the standard interface for batch unit models could be utilised for connections of batch units within a package, either with other batch units or with other continuous units. For the first case, clearly, it would be beneficial to be able to pass time dependent information between the units.

Use of automatic dynamic model generators

Commercial, state of the art, modelling packages presently allow automatic generation of dynamic models from a base case steady state model providing additional information required for dynamics is provided (sizes of equipment, control scheme, controller parameters *etc.*). However, this functionality is typically only available for normally continuous units within the base steady state model and not for the batch models.

Representation of physical properties of complex molecules

Often batch and specialty processes are characterised by handling of materials that incorporate complex or polymeric molecules. For example, the mixtures may contain water, inert gases and condensed phases, associating vapours, dissociating electrolyte species, surfactants, dissociating catalysts (initiators), reacting species, macromolecules. There is a need to develop effective physical, thermodynamic and transport property models for these types of mixtures that can contain several phased (Vapour, Liquid 1, Liquid 2, Polymer particles, and other solids).

As the availability of such properties is scant, often to solve a real problem, all the resources need to be utilised simultaneously. Thus, parameters may come from the databanks of one package, while the physical property models are built and regressed in another package and linked to suitable process models elsewhere. Thus standard interfaces that allow different modelling packages to be linked with each other in this co-operative mode are needed.

Representation of the form of solids

Batch processes are often characterised by presence of solids, either on their own or in slurries. Often continuous fluid processes culminate in batch solid processes as we get closer to the final products. It is required to have standard methods to represent the shapes and sizes of the solid particles and to be able to pass the same information across interfaces (both for single solid phases and solids mixed with fluids). The models and the interfaces should recognise the level of model being used (*e.g.* use of average sizes, use of moments or full blown particle size distributions).

Representation of molecular weights

Similar to Particle size, molecular weight can be expressed in many different ways for complex materials (*e.g.* as average, moments or full-blown distributions). These need to be recognised by the modelling package as well as standard interfaces.

Initial Values for batch models

Help needs to be available to be able to set up initial values for batch models without requiring a steady state run. Initial values are particularly important in equation-based packages but for dynamic models, the initial values of the state variables are important inputs that determine the results obtained.

Dynamic Optimisation

The optimisation functionality used needs to be aligned with the type of model being used. If a time averaged model is used, steady state optimisation algorithms may be used and a mix of batch and continuous units may be handled simultaneously. Of course, the level of such optimisation could not consider the dynamic profiles within a batch. In order to optimise the same, dynamic optimisation algorithms are required so that trajectories can be adjusted to optimise given objective functions. There needs to be recognition by the modelling environment that a dynamic batch optimisation is requested. Standard interfaces should be available to pass such information from one package to another.

Batch Design vs. Rating or Simulation

Batch design algorithms allow directly the design and sizing of equipment without simulation at first. An example is the determination of the number of stages required in a batch distillation column. The modelling environment needs to provide these different functions and be able to distinguish between them and provide standard interfaces for transfer of the information across the interfaces.

Provision of control schemes and controller parameters for batch systems

Increasingly 'intelligent' modelling environments are proposing default control schemes for continuous processes. The same should be available for batch processes and the controller settings should take account of the part of the batch cycle the given process is going through.

Batch Models that run in real time

It should be possible to adjust the rate at which the models can run and to be able to link them through to DCS systems through standard interfaces. Such systems can then be used for training in process operation, educational uses or for possible on-line control.

Legality of interfaces

There is a need for recognition of batch scenario by inter-operability standards and effective on-line help in interfacing (*e.g.* when interfacing large library based packages in general, checking/expressing the viability/legality of connections between modules).

Computational Fluid Dynamics

There is a need for interfacing models at different length and time scales. For example, a process model can pass important process variables (*e.g.* mass flow rates) through to a Computational Fluid Dynamics (CFD) model that can return values for heat and mass transfer coefficients or give insights into mixing. There are a number of initiatives that aim to bring high volume fraction flows to the domain of CFD (*e.g.* BRITE European Project).

Lab-scale to Manufacturing Scale-up, dynamic parameter estimation, optimisation, experimental design

Tools are required that can absorb data generated at a lab scale and help in the scale-up to manufacturing scale for the new products (if possible avoiding but at least minimising the intermediate semi-tech stage). This requires the use of dynamic lab scale data to estimate parameters (interfaces to dynamic parameter estimation tools) and subsequent translation of the model to the manufacturing scale utilising the equipment available the information on which should be available on a databank. Conversely, these models can be used to guide the lab-scale or semi-tech scale experimental work. There is a need to combine mechanistic models with data models in helping and guiding experimental design. A convenient way of interfacing the data collected (both off-line and on-line) to the modelling tools is required and there need to be hooks through to the hierarchical scheduling and control tools within the CIM environment mentioned above. This will also allow more use of full-scale plant data to improve the estimates of parameters.

Intelligent Formulation

Most of the end products are complex formulations of the key ingredients that are manufactured and other manufactured or purchased ingredients. An example is the area of Process Flavours where Block flavours are formulated together with other ingredients to give the products that are sent to various customers in the food industries. At present, there are not many tools available that relate product properties to the formulation. This is a complex problem involving several product properties (physical, objective and subjective) and a formulation that will usually have multiple dimensions.

2.2.6.3 Scheduling/Planning of Batch/Specialties Processes

Both short-term (*e.g.* weekly scheduling linked to current stock and orders) and long-term planning (linked to market forecasts) needs to be accommodated. The latter requires an interface with the forecasting and accounting types of software packages. At the shortest time scales, there is on-line scheduling/ re-scheduling that is adjusted immediately in response to new relevant information becoming available. For the medium and longer-term, it should be possible to carry out sensitivities with respect to key assumptions and variables and what-if scenarios. Examples of questions that can be raised are what if the pump capacity from the reactor is doubled so it can be emptied in half the time. What if additional storage capacity is provided?

Clearly, it is beneficial to be able to pass information through from a modelling package through to a scheduling package through a standard interface. However, there should be good management of this information and only that is required for a given scheduling problem should be passed on. It is important to keep scheduling files to reasonable size as it may be required to distribute them from planning to production and through to the plant level. The process models may be helpful in determining the times that are set in the scheduling models. These are usually based on experience but may sometimes be conservative. Through combined use of modelling and scheduling it should be possible to minimise the equipment availability excess.

For multi-product, flexible plants, compatibility matrices between the different products are worth developing and interfacing to the tools. These give rules on the extent to which one product can follow from another with the minimum need for cleaning.

2.2.6.4 Information and data on Batch/Specialties Processes

Presentation of the right information at the right place and time is key to obtaining optimal development and operation of batch facilities. Some of the techniques that can be used for improvement of operations e.g. SPC (Statistical Process Control) rely on utilisation of data from previous batches. There is the prospect to improve or prepare downstream batch operations dependent upon the indicators upstream. This brings in the general point of interfaces and ease of passing of information from CIM environment to modelling environment through to statistical packages for analysis.

Further requirements for Information and data handling are presented separately for an order driven production environment and made to stock production environment. There is also a discussion on the retrofit and strategic development of batch factories.

Special requirements in an order-driven production environment.

An order-driven production environment is characterised by,

- Customer order receipt triggers major components to be procured / fabricated and assembled into the final product,

- Product lead time for satisfying a customer order equals the time required for procurement, fabrication, assembly, packing, and shipping;
- Expensive components are purchased or manufactured for a specific customer order,
- Supply orders are typically pegged to specific customer orders,
- Inexpensive common use items (fasteners, paint, *etc.*) may be inventoried under reorder point,
- High volume operations with repetitive component use typically use standard cost systems, and
- Low volume operations are most often seen using actual product costing.

Automobiles, stamping presses, bass boats and special application motors are examples of products manufactured in a make-to-order environment. In these factories, all functions must be synchronised to serve many customers simultaneously.

In a make-to-order environment, improvement comes from,

- Reducing the through-put time for components and finished goods to improve inventory turns and lower costs,
- Reducing the number of SKUs (stockkeeping Units), that is, the number of one specific product available for sale, without limiting customer selection, and
- Reducing unfavourable material, labour, scrap and performance variances measured by the costing system.

Special requirements in an make-to-stock production environment

Make-to-stock represents a production environment in which,

- Finished goods are made and stocked in anticipation of customer orders,
- Demand to meet inventory stocking levels drives production,
- Time required to pick, pack, and ship product is the lead time quoted to the customer,
- ROP software techniques typically maintain inventory levels for product not subject to lumpy demand, and
- MRP techniques are used for component items and products subject to lumpy demand.

Consumer products are usually manufactured in a make to stock environment. In fact, anything that can be purchased off the shelf was probably produced in a factory using make to stock manufacturing practices, i.e., toothpaste, power tools, light bulbs, food products, fishing tackle. The only customer of these factories production is the warehouse stocking levels. On the other hand the warehouse may have many customers - distributors, dealers, consumers.

In a make to stock environment, improvement comes from,

- Reducing throughput time for finished goods to justify reducing the stocking level.
- Reducing the lot sizes for manufactured and purchased components to reduce inventory.
- Reduce lead times for manufactured and purchased components to reduce throughput time and inventory.
- Reduce negative cost variances for material, labour, scrap, and everything else measured by the standard costing system.

Strategic Development of Batch Factories - System Retrofitting

Development of new products eventually requires plant capacity expansion, which becomes a complex case of redesign of existing facilities (retrofitting). Solution to the retrofit scenario involves strategic decisions with the necessary trade-off between cost and plant capacity expansion requirements.

The problem of optimal retrofit design of multiproduct batch plants contemplates the case of an existing plant with the sizes and types of its equipment already given. Due to changing market conditions and external economical pressures, it is assumed that new production targets and selling prices are specified for a given set of products. The problem then consists in finding those design modifications that involve purchase of new equipment for the existing plant to maximise the profit.

Design options considered in the retrofit design of a multiproduct plant contemplate the addition of new equipment in two ways; (1) operating in parallel out-of-phase to ease bottleneck stages (thus decreasing the cycle time of a product), or (2) operating in parallel and in-phase with the current equipment. to increase the size of the present batches.

The retrofit design problem can be formulated as a mixed- integer-non-linear-programming (MINLP) problem. Solution to this problem may require excessive computing time when real cases are contemplated. The introduction of appropriate heuristic rules should help to simplify the calculation procedure and insure, at the same time, convergence within a reasonable time.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 2.3: Life Cycle Needs

Hiroshi Okada & Tetsuo Shirao

2.3.1 IINTRODUCTION

The main purpose of this chapter is to identify "life cycle" needs of process industries who are going to use CAPE software as well as other tools during process modelling, simulation, synthesis and process design to achieve better quality of the process design work for better, cheaper, faster and even greener [1][2]. Recently a typical question such as how well systems employed are integrated to share information once created at the upstream phase such that it will flow through seamlessly and utilized many times is getting more attention. A role of the data manager is getting important during the project execution. This reflects the user's life cycle expectations or needs, such as interfacing with different design tools to keep information flow among different disciplines during not only the process design phase which should be fulfilled by CAPE but also across the life cycle.

2.3.2 BACKGROUND

To look into life cycle needs, let's take a look at an example of Chemical industry including organic and non-organic chemicals in the highly competitive market. It has been providing the world with a wide range of products and services. As the chemical market has globalised, selecting a product line and a suitable process, and thereafter optimising the size and location of productions is becoming one of demanded business concerns. Strategies and policies are put into action through the development of programs, budgets, and procedures. Most important decision on what sort of product lines are to be produced, together with issues such as environmental protection, safety, trace ability, quality, productivity, life cycle cost are the ones required keen focal point of the management from the origin of every strategy.

2.3.2.1 Time to Market and Cost Pressure

Time to market and cost pressure is becoming a serious issue of managements of any indsusty. And process industries are not exception to this rule. There are two potential issues involeved in here as shown below. One is efficient use of resources, especially at R&D phase of the life cycle, particulaly faster developement time and optimaized allocation of resources including cost reduction. Thus lifting the cost curve upward. Another is invloved in how to deploy and penetrate the market faster, cheaper, clearner, and even greener; thus, shifting sales cureve to the leftward.

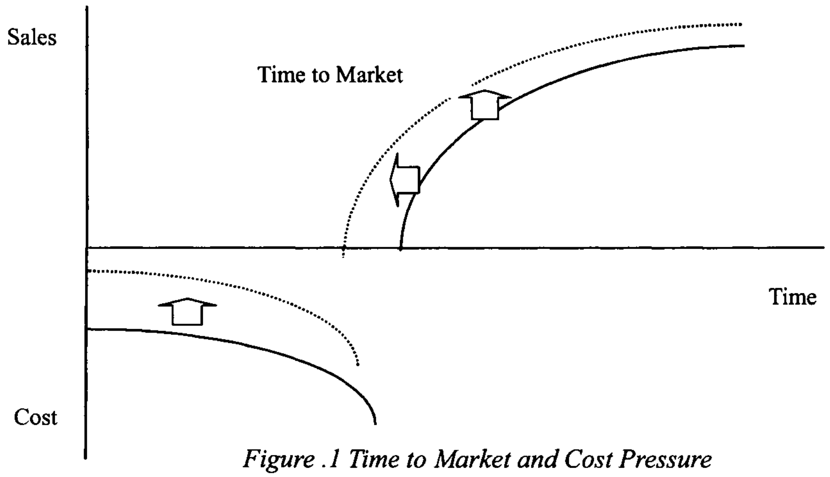


Figure .1 Time to Market and Cost Pressure

2.3.2.2 Driving Forces

Meanwhile, process plant owner and operators, at the same time, are paying more attention to concepts such as "Assets Life Cycle Management" and "Supply Chain Management" for the better management of the business. Where "Assets Life Cycle" is "Life Cycle Model" whilst "Supply Chain" is a Business Model that consists of "Purchase", "Production", "Distribution" and "Sales". Both are actually connected concepts. Beginning with Feasibility Study of the plant, Process Design and Engineering, Procurement will be executed to Construct and Commission the plant. This is called "Plant Asset Creation" in terms of Life Cycle Model, which has been traditionally integrated into Supply Chain Management to perform a role of Production. After the delivery of the plant, the process plant owner and operators will use the process plant as a tool to produce the products and ship to the market. This represents "Plant Asset Deployment". This is to say that "Operation and Maintenance" of the process plant in terms of Assets Life Cycle is "Production" in terms of Business Model where the two concepts are met.

To illustrate this, a new chain of the Asset Life Cycle has to be introduced as shown below:

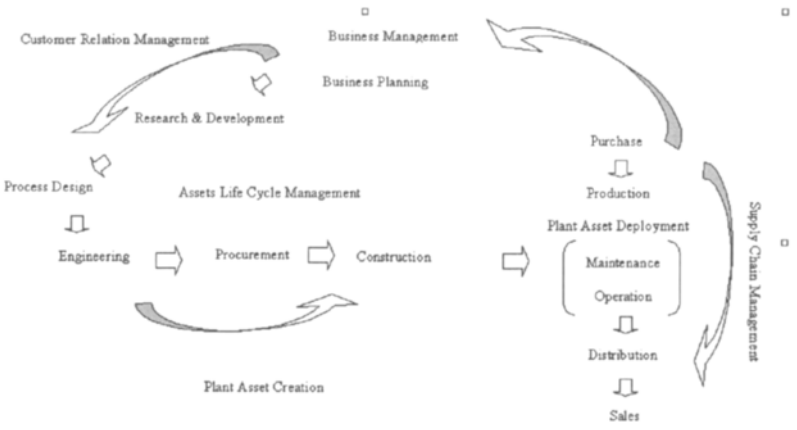


Figure 2. Plant Asset and Supply Chain

The total picture of the process industries consists of "Assets Life cycle," "Supply Chain" and "Business Management" as shown above Figure.2. Information Framework of Plant Asset Creation and that of Supply Chain are to be integrated at Production consists of Maintenance and Operation. IT increasingly provides a strong driver to keep spinning the wheel of this chain. To increase the rotating speed, factors have to be identified so that proper technology can be employed to achieve this goal.

2.3.2.3 Getting things right in the first place: 80/20 rules

It is worth to note that the impact of Process Conceptual Design to Business Decision as a whole to realize a Business Plan of constructing, maintaining, and operating the process plant. Since, in this phase, a business strategic plan will be deployed as the particular product line releasing plan, a decision on what kind of the process shall be designed first will determine the direction. This plan also defines overall cost factors such as Plant Investment, Labour, Raw Material and Fuel cost. It can be summarized that 80 percent of total predicted cost allocated to this business plan has been already locked in this phase, which accounts only the first 20 percent of the project duration. This is called the 80/20 rule, which will often determine the fate of a business decision. Any errors made in the early phase have to be taken care of before pursing into the later phase; otherwise, the greater casualties will be counted in the later. This is the main reason why more attention is given in process industries to the information created during the early phase of the design activities such as process design activity such that the better management of a life can be achieved to yield the fruits. Now with an advanced computing capability emerging Computer-chemistry gives a birth to a new field of Chemistry, which enable to perform molecule level assumption, estimation, and simulation to achieve higher level of results during Research and Development phase. For instances, physical property estimation based on end-user requirements can be performed. Environment impact can be also be calculated based on digitised physical property database in terms of life cycle assessment. End-user's requirements can be studied thoroughly. User requirements as to existing segment or new segment can be also predicated based on information stored in digital reserves. The real contribution of IT is to improve the quality of work such that embed errors can be removed or its effects on the later phase can be minimized due to enemas calculating power covering every possible range of analysis. Therefore "getting thing right at the first place" is what is required to execute any project successfully, and IT is one of the tools to provide the practical solution.

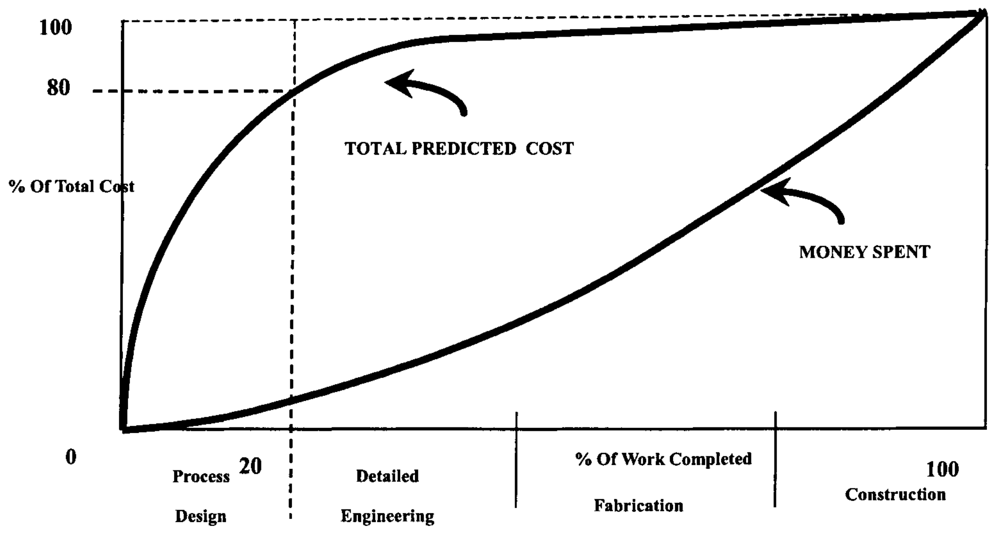


Figure 3. The 80/20 Rule

Primary concerns of engineers who are involved in the development projects in terms of data management and control are summarized as follows:

(a) Exchange and share of data between engineers: This demonstrates a potential problem of coordination. Timely exchange and share of data among participants is the primary concern since this will allow each to proceed with respective scope of work without delay. Also distinguishing between what to be exchanged and what to be shared is also important in terms of data management.

(b) Data consistency and integrity: The potential problems involve incorrect data formats, different data models, not synchronized data turnovers, ill-adherence to stated standards, and future maintainability of data.

(c) Mid- or long-term achieve and retrieval: Availability of data for future reference or modification as required is getting critical.

It should be noted that the above concerns are deemed to be unacceptable and impediment to performance required task efficiency unless proper measurements are taken. Also it should be understood that they would serve as mid- and long-term vehicles for cost saving and not as added cost to the current design practice. In order to ensure consistency, integrity, timeliness, and quality of data as deliverables, a role of data manager should not under-estimated. Especially this becomes a keen issue because internet technology allows one to access to any information one needs, when one needs, to where one use and from where it is stored.

2.3.3 LIFE CYCLE NEEDS

Three aspects of Life cycle Needs including Activity Model, Information Requirement and CAPE software will be explored in this section.

2.3.3.1 Life cycle needs - process industries - activity model

Life cycle models have been discussed in various fashions. Following six major categories of activities are identified as the life cycle model of Global CAPE-OPEN (GCO) project as a result of international harmonizing co-ordination work of GCO activity [3] [4][5]. Descriptions of the respective activity as well as information requirements can be summarized below:

(a) Business Planning: Selecting product lines in accordance with the business strategic planning, quality target, sales volume, cost in consideration of timing, investment target (including investment for development) of the product line are the main concerns in this phase. Any interface has to be capable to handle all of these relevant parameters including sales, target cost, and investment criterion and basically BFD drawings so that Business Planning can be fully executed.

(b) Research and Development(R&D): The purpose of this activity is to establish a process by the Process Development. At this stage of the life cycle, a process simulator will be used to study the process behaviour such that sufficient information will be created to produce a Block Flow Diagram (BFD) drawing and its related data. Economic Analysis in terms of business justification mainly based on raw material and fuel costs will be conducted here. It is very important that every single related attribute data including material safety has to be extracted to hand-over to the next stage of the life cycle. Also managing lessons learned type information such as unsuccessful case studies is important to be achieved for future reference.

(c) Process Design: Conceptual Process Design phase includes activities such as Process Selection, Process Synthesis, Process Analysis and Evaluation. Process Decision will be performed in this phase such that optimised and safe guarded process will be selected though Conceptual and Detail Process Design work with the help of engineering design disciplines. Economic Analysis at this stage will be mainly utilized raw material and fuel costs and plant costs based on earlier process design and case study. CAPE tools are available at this stage.

(d) Commercialisation Decision: The decision will be made based on the analysis of return on investment, profitability, risks in terms of quality, safety and other aspects including the plant location based on off-site conditions and physical distribution, plant safety and reliably. User requirement specifications will be issued after Commercialisation Decision, such that Engineering Design will be started. User requirement specifications will be including Pre-P&ID, Equipment Functional Design, Control and Operability Study, Safety & Health & Environmental Study and Economic study. During this stage, it is essential to bring an idea developed in terms of BFD representation into Preliminary Piping and Instrument Diagram (P&ID) representation. This bridging between them should reflect conversion of functional requirements into mechanical specifications so that the actual plant can be operated as designed.

(e) Engineering design and construction: At this engineering design stage, P&ID will be fixed based on information received from process design and the resulting work of Engineering Design. Material, Equipment and Plant Safety will be built into the actual plant at this stage. Seamless transition of Process Design information to Engineer Design tools is a must.

(f) Operation and maintenance: (Production): After Construction and Commissioning, actual Operation and Maintenance will be applied to maintain the optimum production in response to changing environmental conditions for the process and plant. Handing over all engineering documentation and data to owner and operator is important in terms of data owner ship and responsible for any updates and resulting integration of the system. Especially, data integration of all related data is critical for smooth transition and reuse of them. Change control and revision control of information as changes are made is also important during day-to-day operation.

Life Cycle Model Framework					
Business Plan	Research and Development	Process Design	Commercialisation Decision	Engineering Design and Construction	Operation and Maintenance

Figure 4: Life Cycle Activity

This activity model will provide a whole framework of the process plant life. Further refinement can be done to break into individual work packages, such that typical workflow can be schemed.

2.3.3.2 Life cycle needs - process Industries - information requirements

Based on understanding the needs of the market, a market development plan will be launched and potential demand will be estimated. Then information including process, process material, equipment, cost, safety is processed in every business aspect throughout a life of the process plant engaged by different disciplines using different tools [5]. The below table has been developed to summarize the life cycle requirements of engineers. Process Representation Methods describes the typical drawings used to convey the information graphically. Raw Material Cost and Plant Investment Cost represent typical costing items to be considered at each activity. Operation Methods describe the main focus point in terms of plant operation at each stage. As you will see from Table 1, required granularity of information will be different from phase to phase.

Process Representation Methods represent basic functionality of required operations; thus, they imply different meanings to different disciplines across the life cycle. Table 2 is formulated to illustrate information requirements of process engineers. Plant Function represents various class of equipment concept providing specific service of operations together with respective attributes. Items to be concerned during Equipment Concept development are those used to define specific type of equipment for a given operation.

Table 1. Information requirements of engineers

Business Plan	Business Planning	Research & Develop-ment	Process Conceptual Design	Process Detailed Design	Engineer-ing Design	Operation Maintenance
Process Representatio n Methods	Pre-BFD	Pre-BFD	BFD Pre-P&ID Framework	BFD, Pre-P&ID	BFD, P&ID (AP221, etc)	BFD, P&ID Actual Figure
Raw Material Cost	Estima-tion using Alike process or Literature	Estimation using Alike process or Literature or Experiment	Simulation using Rough mass/ heat balance (Experiment)	Simulation using Mass / heat balance (Experiment)		Simulation using Mass/heat balance or Actual data (Experiment)
Plant Investment Cost	Estim-ation using alike process or Cost DB	Estimation using Alike process or Cost DB	Simulation using Alike process or Cost DB	Simulation using Cost DB or Estimation	Simul-ation using Cost DB or Estim-ation	Simulation using Cost DB or Actual data
Operation Methods			Conceptual operation	Process control & Operability	Plant Control system	SOP Actual operation

BFD: (Block Flow Diagram); P&ID: (Piping and Instrument Diagram); PFD: (Process Flow Diagram) is to be realized as Pre-P&ID; Pre-P&ID Framework indicate P&ID before equipment functional design; Pre indicated preliminary.

Table 2 Information requirements during Process Design Phase

Plant Function	**Attributes**	**Items to be concerned during Equipment Concept development**
Reactor systems	Reaction Heat, Stoichiometry Material Conversion Formula, Composition, Temperature profile, Pressure profile, adiabatic temperature increase, Catalyst, Void ratio, Holding time	Tower, Drum, Heat Exchanger, Piping type. Interior, Abnormal reaction scenario
Column (Tower) systems	Summary data (Column data, Number of trays, Inside diameter, Liquid load, Vapour load), Composition/ Temperature profile, Pressure profile	Identification of Condenser/ related equipment Selection of Interior such as Distiller, Absorber, Extractor, Empty/Packed/Trayed column, Internal Construction: Tray type,

		Packing, Spray
Storage Equipment	Vapour/Liquid separation, Liquid/Liquid separation, Process Information and Holding time to design Interior (Agitator/ Baffle plate/Demister etc) Inlet and outlet, Outlet vapor/liquid, Drain based on Process Information provided	Selection of Drum, Tank, Silo, Vertical/Horizontal, Vertical Tank: Selection of type(Corn roof, Dorm roof, Floating roof etc)
Heat Transfer Equipment	Process Information to design Heating or cooling curve, Heat duty, Heat exchanger(Water cooler, Steam cooler, Process) selection, Phase changes(Total condense, Partial condense, Total vaporize, Partial vaporize) based on Process Information available	Consideration on Heatup or cooldown: Heat source, coolant Selection of Tubular/Spiral/ Plate/Double pipe/Air fin type, Tube side/Shell side fluid
Transfer Equipment	Process Information: Driver, Suction pressure, Discharge pressure, Head, Phase: Solid, Liquid, Liquid based on Process Information *Special Equipment including Ejector, Solid transfer equipment: To be set separately	Selection of Equipment type, NPSH avail, Spare equipment, Vertical/Horizontal and Driver, Gravity (Pump)
Special Equip. / Utility service systems	To be set for each type	
Piping Systems	Stream Information	Selection of Line Number, Size, Pressure/ temperature class, Parts: Valve/ Insulation

The following table (Table 3) illustrates a proposed mapping between Process Function in terms of Blocks represented by various Simulation Packages and Plant Function represented on P&ID. This will help translating an idea of Process Function into Plant Function so that Detailed Engineering can be expanded to provide services required by Process Function for a real world application. As a result, the Conceptual Process Design is conducted solely to create the information to conceptualise a process for producing a certain product line as indicated in Business Plan.

Table 3. Mapping from Block to Equipment Concept

Plant Function	**Aspen Block**	**HYSYS Block**	**PRO2 Block**
Reactor Systems	RSTOIC, RCSTR,	PFR, CSTR	PFR, CSTR
Column Systems	*	COLUMN	COLUMN
Drum / Tank	FLASH3, DECANTER	3 phase separator, Tank	FLASH
Heat exchanger Heater/ Cooler	HEATX, HEATER	Heat exchanger, Reboiler, Heater, Condenser, Cooler, Air Cooler	HX
Pump	PUMP	Pump	PUMP
Compressor	COMPR/ Expander	Compressor/Expander	COMPRESSOR/ Expander
Other transfer equip	*	*	*
Special Equipment	*	*	Cyclone
Piping Systems	FSPLIT, VALVE, *	Tee, Valve, *	PIPE, FSPLIT, VALVE, *
Utility service Systems	*	*	*

* indicates that special rules are required as suggested below:
FLASH2 Block→ Heater/ Cooler or Drum or Piping Systems (VALVE: 1 input and 1 output) ; MIXER Block → Drum or Piping Systems
RADFRAC Block → include Column, Drum, Heat exchanger, etc
SEP Block→ Special Equipment, etc

The following diagram (Figure 5) illustrates the basic idea behind this mapping:

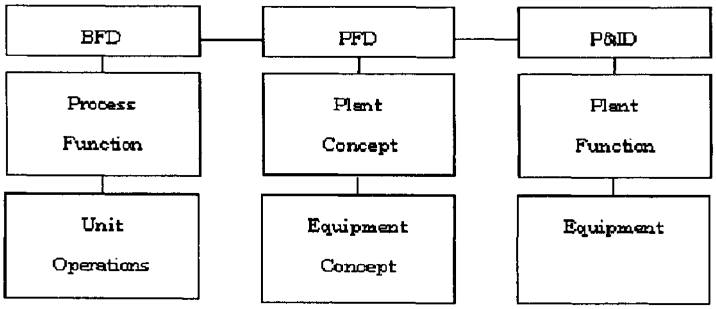


Figure 5. Mapping between Process Functions and Diagrams

The value of information generated during the Process Design phase is always verified and evaluated in terms of the contribution to firm business decisions to produce specific products based on the process designed as shown in Figure 6. Information should be well-measured one so the result will be constructive that the business decision can be profitably made and executed.

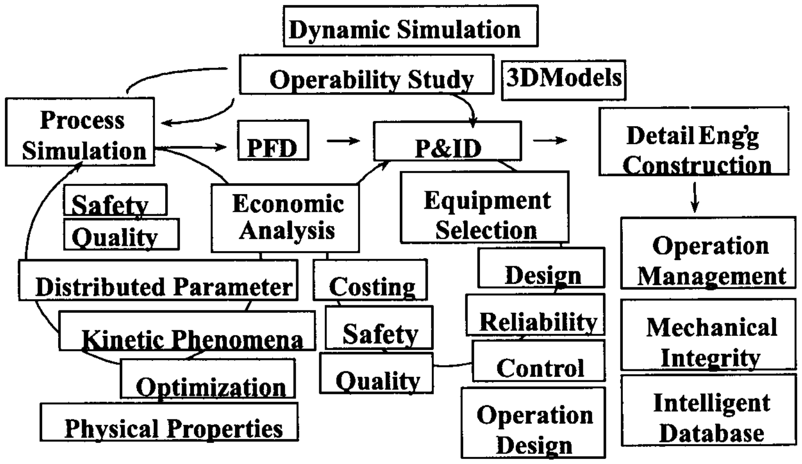


Figure 6. Process Engineering Activity

The information once created should be shared through interfaces among activities consisting the life, including Business Plan, Research & Development, Process Design, Commercialisation Decision, Engineering Design and Construction, and Operation and Maintenance. Therefore, Life cycle is a collection of phases of the life that divides a management effort into several more manageable subordinate efforts and management loop to conduct the business perpetually.

2.3.3.3 Life cycle needs - process industries - CAPE software

Almost every activity will create new information after having been processed. The use of the information is not restricted to the activity that creates and processes it. It is used in other activities in a chain to carry on a business plan e.g. to delivery a new product line. Moreover, it is very important to note that information on the product created has to outlive the life of the product per se. For example, information generated, integrated and archived during the process design will be used to generate all sorts of other information through the life of the process, which are typically twenty to thirty years. This is the major reason why managing the information throughout the life is becoming important issue.

To manage information in order to cope with business requirements, the CAPE systems used must be capable of,

(a) Knowing what information they have, and what it is about,
(b) Interfacing information between organisations and systems without degrading the contents,
(c) Integrating and storing information from different systems,
(d) Sharing the same information among different systems with different users' needs,
(e) Managing the information, including history, for a life.

Of course, any software that is designed to fit the needs of process engineers is no exception to this. To share the earnings among the participants of process design activities depends on how well a system employed is integrated to flow the information once created at the upstream phase of the entire design activities despite of different formats each system uses. Often this shows a bottleneck since availability and grade of the information has been very limited inside the boundary of the application systems used. Most of process engineers have to spend their significant time just to re-key in the information provided at the upstream design activities and at the same time, most importantly, keep track of configuration of each information as each revision is made. One of the solutions to this will be replied by CAPE software by proposing the interface standards among the components of simulation software such as Solver Package, and Unit Operations. Use of such components will promote standard mechanisms for various applications to inter-operate and facilitate the sharing of information between applications. CAPE software will provide various services to the process engineering related activities during the early phase of the conceptual process design thus to reach new quality and productivity levels.

It is worth mentioning the following concerns of the business so that CAPE software to be more sufficient to fit the needs of the business. Differentiation between Engineering/product definition and Product should be noted. Too much focus on engineering/product definition may lead to the conclusion that production is being supported insufficiently. Also actual life means that history is involved. Dealing with organizations and processes already in-place and new paradigm and transition as to where and how business goes forward should be concerned. Change management opportunities and requirements and data ownership are the real issues to be concerned.

2.3.4 GENERAL & SPECIALIZED NEEDS OF LIFE CYCLE SUPPORT BY CAPE SOFTWARE

To match market requirements for a new product line required functionality to be designed based on information available at that time. Completeness and accuracy of information is the key concern of process engineers. To avoid the risk of non-rigidity and degraded decision, information management using various software tools has been schemed and devised. However islands of software tools causing such a greater communication problem, when the management of information is concerned. Bridging the isolated systems through standard interfaces will be one of solutions where great improvement on overall performance is expected based on implied 80/20 rule. It is always suitable to consider the way to leverage the emerging information technology solution to gain the improvement therefore to gain competitive advantages. This is the reason why CAPE Software based on a common understanding of the industry work process and information requirements is becoming important to be explored. In term of life view, general needs, as well as specialized needs will be elaborated as follows.

2.3.4.1 General Needs

In this section general needs will be elaborated. Due to the latest development in IT industry, what engineers want to accomplish can be achieved. Once upon a time when capability provided by IT has been far less than what engineers requested. What can IT or a computer to be more specific, be used type question was the main topic; therefore only the IT layer consists of IT components such as COBOL, Mainframe computers has been discussed. Power of IT has been improved, then solution layer obtained such attention that utilization of the particular application system such as Data Base Management System, CAD system CIM composed of IT components has been focussed. Nowadays, Business Planning based on the strategic decision makes clear the demand of integrating proper application systems into the designed business process. It is important to note that each activity of business process consisting Life cycle Model mentioned utilizes some sort of system components to process data represented by its own data model. Through a set of externally accessible operations, interfaces are so defined to bridge between different system components accordingly to transmit data required. Basic idea of CAPE software set interoperability standards for communication between system components in process design shall provide a solution to overcome interfacing problems of exiting proprietary systems so that it shall be able to improve the efficiency and speed of the whole throughput.

Business Plan	Research and Development	Process Design	Commerciali zation Decision	Engineering Design and Construction	Operation and Maintenance

Life Cycle Model Framework

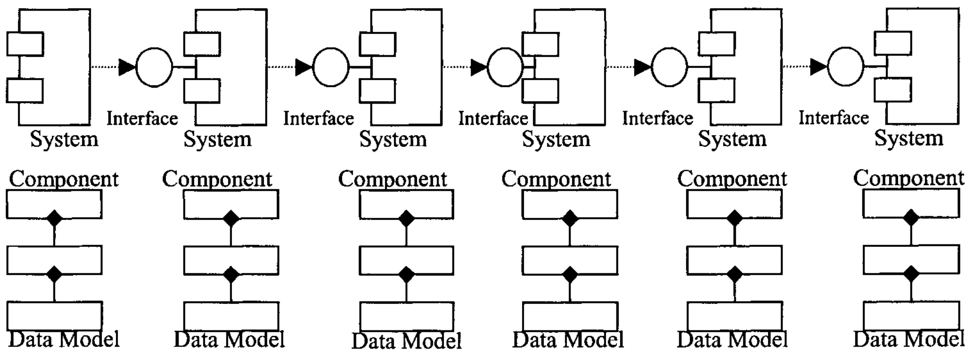


Figure 7: Life Cycle Model Framework

What industries have seen as a change will be represented as the way how each computer aided system is used and integrated to create data and manage information. A critical issue tends to be whether or not different systems are able to communicate each other. This means at the information technology level, data is exchanged and/or shared when it is created and used by different computer systems, disciplines, or even different organizations sometimes. For instance, all the engineering related data, which is hand over to the owner and operator is used to feed the systems such as a maintenance system or an even to accounting system to keep track of the life of each equipment as the vital asset. This is becoming more important when Information Technology becomes far much advanced and is to be utilized to perform more efficiently than the past.

It is also worth noticing that life of computer system is much shorter than that of process plant per se so that solution may work at that point of time, but may not necessarily work at the different point of time as the life passes by. For instance, CAD systems, which are commonly used during the design work phase of the process plant projects, were once running on mainframe computers with various operating systems, then work stations including UNIX systems, now some of them running on PCs with Windows operating systems. Yet the process plant is still standing, but the hardware of the system has been replaced with the successors with different operating systems. When the CAD system itself becomes obsolete, it has to be replaced with the new model. In order to cope with this situation, not only the application system but also data may also have to be converted. This is the reason why managing the "life" is becoming the key issue among process industries.

Process Design is an activity whose mission is to design functional requirements to meet the market needs. Especially Process Synthesis is the key activity to dominate the result of 80/20 rule according to standardization work of MITI funded GCO project [4]. The following addresses Process Design phase specific needs in terms of information requirements:

(a) Evaluation based on data created by various process designs tools is very essential at the early design phase. Interfacing with simulation applications, equipment design applications such as various sizing packages or in-house engineering calculation software of any equipment, as well as down-stream engineering design packages, especially costing applications to provide quantitative analysis of various design alternatives is required to perform any activity with less interference. Performing the complete analysis at the earlier stage as possible will be a key to succeed determining the fate of the business plan as shown in 80/20 rule.

(b) Leveraging process data having been available from the simulation results to generate conceptual process design level as well as conceptual engineering design level of details and representations is one of the sought achievements at process design phase. By bridging between process function and plant function to generate automatically Pre-P&ID from BFD using standard design best practices requires standardization on the mapping and interface between functional components of the process resulting from process design and physical components of the plant resulting from engineering design.

Exchanging and sharing information among existing software, which constitutes the elements of this integrated environments, has been recently taken up as the subject for standardization such as ISO, ISA, IEC, etc., but the available scope is still very rigid and limited to cover the whole life cycle per se. And significant efforts including those for standardization are required for the construction of the integrated environment. In real practice, each application tends to be customized to reflect the individual needs of the users. Moreover, each project may use a different set of the software to solve the project specific needs. Therefore, it is important to consider one overall concept so that difference among user of the systems and procedures of the systems including CAPE software are well considered and seamless flow of information can be secured.

2.3.4.2 Specialized needs

In this section, specialized needs of process engineers will be elaborated. When a process design was first developed, its deliverable will be graphically represented as a BFD. The main interest is to visualize the sequence of operation and to list some attributes such as heat balance and mass balance. Evaluation based on raw material cost and plant investment cost, which is estimated, based on similar process, literature, and experiment sources will be performed to seek the best solution at this stage. It is worth doing at this stage of Life cycle Model to archive not only the base case but also every single case with relevant information into digital storage that has been considered during the design phase. Later phase of the development, the information can be shared and exchanged based on this repository. Information such as why this value of the parameter has been used over the other type information will become vital in the later days when a similar study has to be conducted, which normally lost. This can be treated as a case library that can be data mined later. Super class – Sub class and inheritance and derivative type relationship can be used to implement this library in object-oriented manner so that emerging new technology can best make use of it as described in other capture.

When the process design finalized, an engineering design will be developed. Drawings such as P&ID and completed datasheets and lists would be issued. Then more detailed cost estimate can be also calculated at that stage since more specific consideration has been identified. Again well-managed digital storage can play a critical role to maintain integrity of the information.

For instance, during Process Design phase, various emerging computer based tools such as simulators, analysis tools, design tools, CAD systems, various data base management systems, spread-sheet applications, word processors are commonly used to generate, store and manipulate necessary electric data as shown below in Table 4 [5]:

Table 4. Design Tools, Data and DB

	Business Planning	**Research & Development**	**Process Conceptual**	**Process Detailed**	**Engineering Design**	**Operation Maintenance**
Design Tools	Linear Programming software	Simulator (Material, Reaction, etc) Fluid dynamics	Simulator (Process) Pinch & Residue Curve Analysis Fluid dynamics	Simulator (Facility) Fluid dynamics HTRI, HTFS Hazop, FTA AXSYS, Zyqad 2D-CAD	Plant design (System & Mechanical) HTRI, HTFS 2D/3D-CAD	Simulator (Process) Pinch-Analysis Fluid dynamics Hazop, FTA 2D/3D-CAD
Data & DB Others	Market data Cost & SHE DB	Reaction DB Material DB Experiment DB Cost & SHE DB	Thermo.DB Experiment DB Cost & SHE DB	Experiment DB Cost & SHE DB	Cost &SHE DB	Analyzer Actual Plant Data Cost & SHE DB

2D/3D-CAD systems are, for examples, Micro Station, PDS, PDMS, Plant Space.

Cost DB: SRI, CHEM SYSTEMS, In house data includes Cost Estimation Tool (IPE, ICARUS2000, etc), SHE: Safety, Health and Environmental issues

Bridging between Process Representation and Economic Analysis reflecting required graininess of information is therefore summarized below in Table 5:

Table 5. Mapping between Cost Estimate and Design Representations

Life cycle Model	Plan & RD	Process Design		Engineering Design	Operation Maintenance
Rough Estimate	Pre BFD	BFD	BFD	BFD	BFD
Design & Estimate			Pre P&ID	P&ID	P&ID
Actual Estimate					Actual figure

BFD: Rough Cost Estimation → Rough Economic Analysis

Pre P&ID / P&ID: Design & Cost Estimation → Design & Economic Analysis

In order to assure the seamless flow of information, an experimental integrated environment for process designing as shown below is schemed for Chemical CALS project in Japan, which intended to demonstrate the power of the seamless interfaces among design tools. The project reported the significant improvement in quality of work done:

The success of the project relies on integrating simulation tools, Costing tools for Feasibility Study, Rough Costing tools for early process design phase, construction phase, various design tools, and database management and CAD system; therefore, the concept of the integrated system has been proved. Next step would be to break the integrated system into functional components such that each component can be plug and play, as it is needed for the project execution.

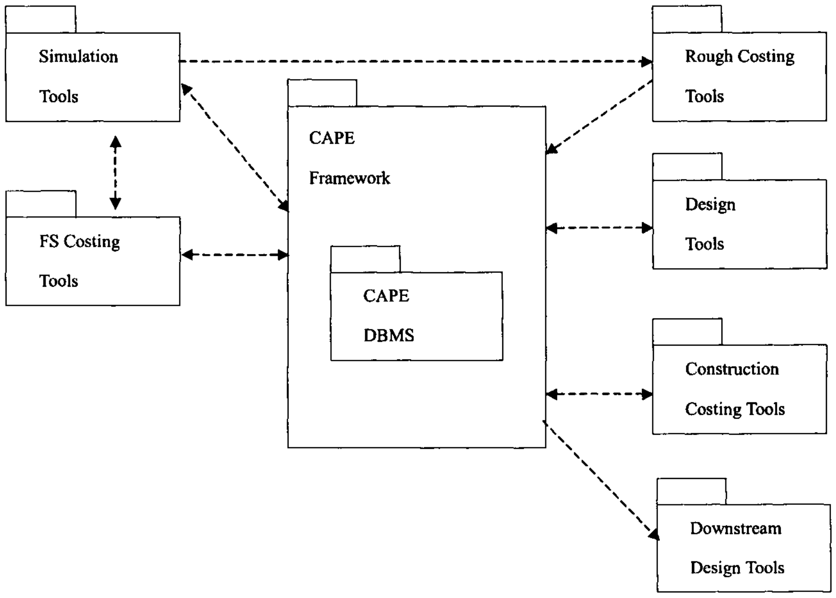


Figure 8: An experimental integrated environment

2.3.5 MEDIUM TERM & LONG TERM VISION - WHERE ARE WE GOING FROM HERE?

It is very important to identify the proper seeds of a business opportunity. Especially the industry who is specialized in producing basic material to other industries facing big challenges such as Market Analysis, Quality Assurance, Environment Protection, Recycling Resources. Everyday various simulations are conducted to define the products to meet the needs based on Market Analysis. Changing the brand mixture, shorter delivery time, and providing customer satisfaction are the key words playing important roles of the industry. Business justification is taking account of market development, pros and cons of business planning, risks vs. growth decision. But none of systems employed today can handle this situation seamlessly. For instance, a commodity product vs. specialized products is a typical issue, but no clear answer can be prepared by the existing system. As time to market and cost pressure is building up, production systems of specialized products are getting attention. An unique idea such as making use of multi-purpose equipment to produce the specialized product instead of designing specific equipment designed for the product is worth attempting since it will reduce the whole time to produce. Whilst commodity type, which possesses longer life, may utilize the specially designed equipment to reduce the overall cost and assure the quality. One concern here is that none of vendors is ready to accept the challenge since it has not yet armed with models capable to handle required simulation. Another growing concern shown in the business is Life Cycle Assessment, which requires process design to consider the environmental impacts. Precise calculation of the impact is getting more attention since sustainability is becoming the hot topic. CAPE software has been demonstrating opportunities, which can be gained from standard interfaces for process simulation components. It is an enabler integrates simulation into the process design such that enhanced competitive and environmental advantages can be obtained. Specialization of general needs is gaining popular votes among the industry. One such example is to bridge between Research & Development and Process Design since Computer-chemistry is getting more attention. CAPE software will give birth to a standard body, internationally supported and recognized, as well as to a network of integration laboratories for process simulation. Furthermore, the specifications should cover Computer Aided Process Engineering, and not just process simulation. Within this framework, existing ISO standards that cover parts of Process Design, Plant Design, and Plant Operation and Maintenance fields, have to be harmonized, in order to avoid duplication and inconsistencies among recognized standards. Whilst another bursting area of concern is Demand Chain Management where the business is required to shift its business paradigm requiring agile and flexible processing. Solution to these should enable CAPE software to contribute to achieve business excellence better, cheaper, and faster even greener than ever.

2.3.6 REFERENCES

[1] Proceedigns ESCAPE-9 & PRES'99 May 31-June 2, 1999, Budapest, Hungary.
[2] Bertrand L. Braunschweig, C. Pantelides, Herbert I. Britt and Sergi Sama: Open Software Architectures for Process Modelling, Proceedings FOCAPD'99, CACHE Publications, (1999).
[3] Henriksen, J.P., Russel, B.M., Static and Dynamic Optimisation of CAPE

Problems using a Model Testbed, Computer Aided Chemical Engineering, Vol. 9, Elsevier, (2001) pp1009-1016.
[4] Hiroshi Okada (JGC), Tetsuo Shirao (MCC), New Chemical Process Economic Analysis Methods, Computer Aided Chemical Engineering, Vol. 9, Elsevier, (2001), pp1059-1064.
[5] Hiroshi Okada (JGC), Tetsuo Shirao, Naoki Dohi (MCC), Chemical Process Analysis Methods, Proceedings of Conference of Japan IMS Research Results, July11-12 (2001), pp 173-176.

Part III: Framework for CAPE tools

The chapters of this part introduce some of the CAPE methods that provide the framework for CAPE tools. There are five chapters, each dealing an important topic with respect to current and future CAPE tools. Although many topics could have been chosen, the particular selection has been made based on their use in current CAPE software. The chapters, covering modelling, numerical methods, simulation, design/analysis, data reconciliation and computer tools for discrete/hybrid systems, present the state of the art of representations, algorithms and architectures. They give concrete examples of how these frameworks can help to match the needs expressed in Part II.

Models are considered to be valuable assets for engineering and decision-making processes because they are not just data but embody a lot of knowledge about the process studied and can be used to generate information about it. The purpose of modelling as a work process is to transform the perception of reality or an idea into a symbolic language, which consists of a set of well-defined concepts with an agreed understanding. Starting with an overview on the modelling process, Chapter 3.1 written by von Wedel et al. provides a review of modelling concepts and languages followed by discussions on modelling tool functionality and modelling tool architecture.

Typical process modelling activities involve the use of a number of different numerical algorithms (solvers). Chapter 3.2 (Tolsma and Barton) briefly describes several common numerical activities and some of the algorithms applied. In particular, the information that must be supplied in addition to the model equations is emphasised.

Process simulation, design and analysis are some of the most common activities in computer aided process engineering. Process simulators are used daily by engineers and scientists almost on a daily basis to solve various CAPE related problems (such as process design) by generating *process information through process simulation and then by analysing the generated results. Chapter 3.3 written by Hocking et al., provide the software side of the story with respect to simulation, design and analysis. Definitions and explanations of the important terms are provided and discussions on problem definitions and the framework requirements for the corresponding software are presented together with examples from commercial simulators.*

Data reconciliation and parameter estimation are important components of model fitting, validation, and real time optimisation in chemical industries. In its most general form, data reconciliation is a minimization of measurement errors subject to satisfying the constraints of the process model. Parameter estimation is the step after data reconciliation in which the reconciled values of the process variables are used to set values for the model parameters. Chapter 3.4 (Arora et al.) discusses optimisation strategies that deal with data reconciliation and gross error detection. Included in this study are two case studies, a comprehensive steady state example and a small dynamic example. Both illustrate these strategies and issues related to this task.

The software tools for batch processes may be classified in two broad classes. The first class involves modelling, simulation, and analysis of the physico-chemical processes that take place during batch operations. The second class is designed to support the decision processes at the different managerial and operational levels of the plant operational hierarchy. In chapter 3.5, Puigjaner et al. review a representative set of tools from the two classes of tools that are available commercially. While there is much innovative research and development in both academic and industrial groups in this domain, experimental software and prototypes have not been considered in the discussion.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 3.1: Modelling Frameworks

L. von Wedel, W. Marquardt & R. Gani

3.1.1 INTRODUCTION

Process Systems Engineering, PSE, has gained an important status for a broad range of chemical engineering activities. A basic requirement for applying the techniques provided by this rather young discipline is based on the notion of a model. For the purpose of the chapter, a (limiting) characterization of a model as a formal system is made. As a requirement, the model represents relevant properties (structural and behavioural) of the system under consideration. The essential feature of a model (with respect to PSE) is that it can be formally evaluated to make statements about a system. This feature allows the use of digital computers, which have become an essential tool for many tasks in process systems engineering that are now characterized by the term computer-aided process engineering. Models must be considered valuable assets for engineering and decision-making processes because they are not just data but embody a lot of knowledge about the process studied and can be used to generate information about it. Models allow virtual experiments through process simulation and/or optimisation that would be costly or even impossible to perform otherwise.

Modelling activities consider a variety of chemical engineering elements on different scales of complexity (Marquardt, *et al.*, 2000; Pantelides, 2001). Model-based studies cover a range from designing molecules (Harper *et al.* 1999; Meniai *et al.* 1998) on one end of a size scale as well as studies of the supply chain between different plants or even sites on the other end (Backx, *et al.*, 1998). The relevant time scales range from microseconds to months or even years, respectively. Between these two extremes, the most commonly employed models today represent thermodynamic phases, single unit operations, or (usually only a part of) a chemical process. Besides modelling physical processes, models of operation modes (especially of batch processes) are also of interest for simulation and optimisation applications.

3.1.1.1 Modelling Process Overview

The purpose of modelling as a work process is to transform the perception of reality or an idea into a symbolic language, which consists of a set of well-defined concepts with an agreed understanding. There are many possible models of a certain perception, but only few of them qualify as being useful to answer the questions relevant in the current modelling context (where a context is understood as a current situation and a desired goal). A modelling process should thus systematically lead to a useful model and discriminate those irrelevant. In the reverse, however, a specific model may be needed for several model-based applications such as identification, simulation and optimisation during the plant development process.

The modelling work process is also important with respect to developing supporting tools for model development because any software tool must focus on the work processes it is intended to support. Modellers are generally unwilling to change the work processes that they have successfully performed for a number of years. Further, a weak focus on the work process is likely to miss the important requirements where support is needed most. Therefore, this chapter briefly summarizes the work of Foss *et al.* (1998), which presents a field study about the modelling process as it is conducted in chemical industries (*cf.* Figure 1). Several issues including documentation, conceptual modelling, model implementation, and model application have been considered in this field study. A lot of detailed information about various facets of modelling processes can also be found in Hangos, Cameron (2001b).

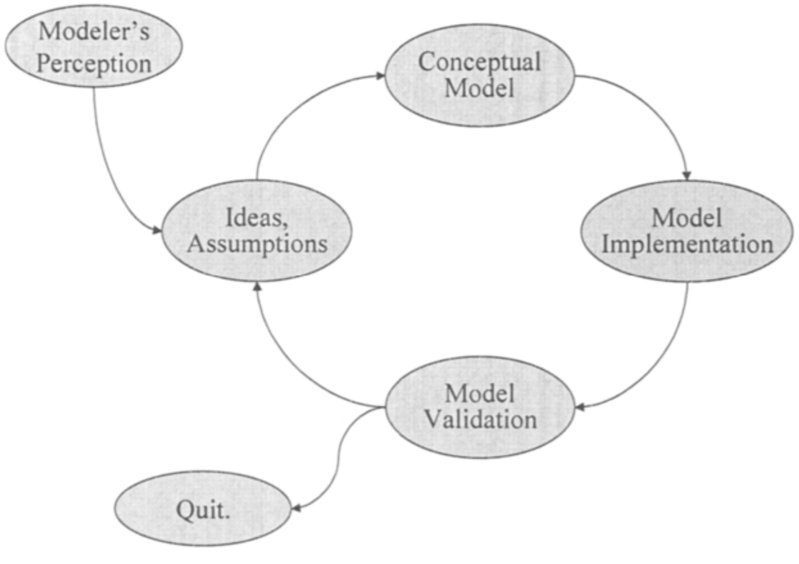


Figure 1. Modelling as an iterative work process

Documentation is described as an important means of preserving and communicating informal knowledge about the models being developed. The modelling process should therefore start with some sort of initial problem specification, reflecting the modeller's initial ideas, requirements, and assumptions on the model to be built. Among these, the purpose of the model should be specified a-priori, for example as a target against which simplifications can be evaluated. This document can further contain any existing information about process, equipment, or material under consideration. A further step towards a good documentation is to trace decisions that have been made during the modelling process.

The interviewees consulted by Foss *et al.* (1998) state that a *conceptual model* of the chemical process to develop a model for is (at least) helpful (if not essential) for discussing with domain experts. This conceptual model specifies the model in terms of the chemical engineering domain together with its assumptions rather than as a set of equation or even a chunk of a programming language. Ideally, a conceptual model could be defined without writing any equations, but, as we will reveal later, current modelling languages do not fulfil this goal. The fact that there is more to a model than just its equations has been further elaborated by e.g. Subrahmanian *et al.* (1993) or Marquardt (1996). The conceptual model can be considered as independent of a realization in a particular simulator, it is rather the model the engineer would sketch on paper before actually using a tool to implement the model.

Given the conceptual model, an *implementation* in a specific numeric application must be developed in order to analyse the model. Different tools may be chosen for this task, depending on the context, their capabilities and the experience of the modeller. Engineering concepts must be translated into the concepts provided by the tool in an appropriate and systematic manner. It is likely that differences in expressiveness between the representation for the conceptual model and the chosen tool may require extra work during the implementation step. The model implementation should be written with respect to numerical performance and robustness (e.g. by reordering equations or scaling variables, *cf.* Chapter 3.2). Such issues are best considered during model implementation rather than on a conceptual, domain-centred level. As an example, a spatially distributed model should be developed using partial differential equations on a conceptual level. If implemented in FORTRAN for example, a discretisation of the model becomes necessary because FORTRAN does not support any partial derivatives.

Often, the development of a conceptual model is skipped and an existing implementation of some model is *reused*, leading to a potential source of errors and inconsistencies in the overall model if the underlying assumptions are not quite appropriate in the modelling context. Inconsistencies can be avoided or, at least, model debugging can be simplified if a conceptual model and documentation of a model implementation are available because reasoning can then be performed on a physical level instead of a level of mathematics or even source code.

Modelling is a highly iterative process where each model developed must be validated against the actual plant behaviour to check whether the modelling assumptions made are a useful abstraction of the reality. If this is not the case, one will have to improve the model or use different techniques to tackle a problem (Figure 1).

Once implemented and validated, the model can be used in a model-based *application*. Degrees of freedom and inputs to the process have to be specified. Depending on these specifications different problem types result, such as simulation, design, or optimisation problems. However, there is no solver package of commercial strength that is able to carry out the full range of model-based applications that are employed during the plant development lifecycle. Also, in general, model implementations of different software packages are incompatible. This makes frequent reimplementations of the models necessary, a costly but unproductive step that can be facilitated if a conceptual model has been developed from the outset.

From an organizational point of view, the modelling process is often carried out by a number of people, contributing to different parts of the model (such as a distillation specialist) or they are active in different modelling activities (such as parameter identification). Hence, the modelling process in such a group must also support coordination of the joint effort on the one hand and sharing/distributing relevant information on the other. Often, the developers and the users of a model are distinct persons, leading to further complications. First, the developer of the model must ensure that the user (who is less familiar with the model and its limitations) must be able to understand and employ it. Further, different people from different domains have diverse understandings of concepts so that a control engineer would probably prefer a view on a reactor model as a transfer function relating measurements and actuators, for example, while the reaction expert is most likely interested in the reaction kinetics and thermophysical properties of the material system.

The discussion above has revealed that a single piece of process equipment (such as a reactor or distillation column) will be modelled for a number of different purposes during the plant development and operation lifecycle. These models are often coded in incompatible tools, which implement different model-based studies that are required in the design lifecycle. These incompatibilities are a major obstacle on the path towards an efficient use of model-based techniques.

3.1.1.2 Multifacetted and Lifecycle Process Modelling

A systematic approach to an understanding of the above issues of process modelling was first undertaken by Stephanopoulos, Henning, Leone (1990b). The term *multifacetted modelling* was characterized by considering process models being developed on different levels (multilevel modelling), for different modelling contexts (multicontext modelling) and are being considered from different perspectives of interest (multiview modelling). During the past decade, those considerations have evolved into what is nowadays called *lifecycle process modelling*, where not only the different facets of process models (in the sense of data) are considered, but also their relationships and the evolution of these artefacts in the sense of a work process (Marquardt, *et al.*, 2000).

A lifecycle-centred view on the modelling process focuses on tracking the evolution of a model from its initial creation, through a number of application and modification cycles. It makes explicit that different versions and views of a model are by no means independent because they describe the same physical piece of equipment so that we can talk about a family of models rather than about a single model.

Such an inclusive representation of information about models enables to capture also the negative experiences (what better not to do), whereas the final outcome of an engineering design process only covers the decisions finally taken, and these are not even made explicit. Further, a comprehensive representation of mathematical models together with the work processes involved is an indispensable contribution to model reuse and understanding, even after a long period of time. These large time scales occur frequently in engineering where a phase of intense design work is followed by a rather long phase of operation. In subsequent reengineering phase (such as a capacity extension of a plant), accumulated knowledge (such as mathematical models) is often lost because design team members have changed their positions or left the company or it cannot be understood because the rationale behind it has not been documented.

3.1.1.3 Requirements for Modelling Frameworks

Summarizing the above statements, the *requirements* on computer-aided support for chemical process modelling are means to,

- Provide a rich set of concepts to describe models using chemical process engineering terms without being tailored to specific simulation applications or even tools,
- Turn domain knowledge into functionality to effectively speed up the development of models for a certain purpose,
- Capture informal information belonging to the model and its development process to improve the communication of and about models in a team of process developers,
- Ease the transition to/between different model-based applications, and
- Make the evolution of model artefacts and the modelling work process explicit to provide a better understanding over the long-term.

Given these requirements, the remainder of this Chapter will review existing modelling frameworks, which are popular in chemical process industries and academia. As the term of a modelling framework is not precisely defined other-

wise, we will use the generic definition of a framework from Merriam Webster's Dictionary as a *basic conceptual structure*, which here refers to modelling tool architectures. Such a framework must cover:
A set of *concepts* that can be used to construct chemical process models,

- One or more *representations* of these concepts suitable for communication among humans and/or between tools,
- *Functionality* to automate parts of the modelling process, and
- An *architecture* that exploits the model representation and organizes modelling functionality in a tool implementation.

Section 3.1.2 will provide an overview of modelling concepts and languages as concrete realizations of the first two issues, whereas the third will be the topic of Section 3.1.3, discussing modelling tool functionalities. Modelling tool architectures to organize the functionality and respective implementations in state-of-the-art tools are discussed in Section 3.1.4. Finally, Section 3.1.5 will pick up the requirements and identify future directions and potentials.

3.1.2 MODELLING CONCEPTS AND LANGUAGES

A process model can have a wide variety of representations. The general representation must be considered with respect to two independent coordinates. Primarily, the model is built using a set of *modelling concepts* each of which must have an agreed understanding among the users of the model. As an example, the concepts unit operation and stream can be used to describe the structure of a chemical process. These concepts can be considered the semantics of the model. Further, these concepts must be notated in an unambiguous *language* to enable the concepts of the model to be communicated. The concepts employed to describe a process model can be used to distinguish the large set of existing approaches into three groups. As *programming languages* we cover those where a model is directly implemented into the solver, *e.g.* as a subroutine. These are not discussed any further because they do not provide any specific means for model representations. *Generic modelling languages* are those that do not provide domain-specific concepts but concentrate on a mathematical (or systems) level whereas *domain-oriented (process) modelling languages* provide modelling concepts, which correspond to domain-specific concepts (Marquardt, 1996).

3.1.2.1 Generic modelling languages

From a mathematical point of view, models employed in CAPE can use a continuous or a discrete representation of system states and time, respectively. Rigorous modelling based on physical insight naturally leads to models, which are continuous with respect to states and time. However, many applications require simpler models, for example to increase robustness or speed up calculations performed with the model. These requirements are often addressed by discretising the model in time or states. The remainder of the chapter will focus on quantitative models as they are obtained from first principles modelling methodologies (Hangos, Cameron, 2001b; Marquardt, 1996).

Mathematical modelling languages

In a mathematical modelling language, the simplest model is represented as a set of variables x and equations f. Whereas the variables represent system states, inputs, and outputs, the equations constrain possible values of the variables and thereby represent knowledge on the domain of interest:

$$\mathbf{f}(\mathbf{x}) = 0 \tag{1}$$

An equation system like (1) if often used to represent a system in its steady state. The equations f constitute of balance equations of the chosen balance envelope, constitutive equations describing equilibrium states (*e.g.* physical and chemical equilibrium) or non-equilibrium processes such as diffusion or reaction. Further, material properties must relate intensive quantities such as specific enthalpy with pressure, temperature, and composition.

The simulation of a model represented in such a way is then essentially a process of finding the roots of these equations (*cf.* Chapter 3.2). However, the equation system resulting from the modelling process may contain more variables than equations necessitating additional specifications for a number of unknown variables in order to achieve a consistent mathematical formulation. Since the formulation (1) of the model does not indicate which of the variables are considered as inputs to the model, it is also referred to as an *open form model.* On the contrary, in a *closed form model* the selection of inputs and outputs are fixed and integrated to a solution algorithm (Marquardt, *et al.*, 2000; Biegler, *et al.*, 2000).

In addition to the steady state, one is often interested in trajectories, which result from a transition of a system from one steady state to another, for example, due to a change in a plant's feed streams. This class of models dynamic models is often represented as a differential-algebraic equation system:

$$\mathbf{f}(\frac{d\mathbf{x}}{dt}, \mathbf{x}, t) = 0 \tag{2a}$$

Such an equation system describes the change of balanced quantities (e.g. mass, energy, or momentum) with time as a result of fluxes entering or leaving the balance envelope and sources in the envelope's interior. The solution of such a system consists essentially of finding the integral

$$\mathbf{x}(t) = \int_0^{t_e} \frac{d\mathbf{x}}{dt} dt \tag{2b}$$

As symbolic solutions of (2a), (2b) are almost always impossible to find for relevant engineering applications, a numerical integration algorithm has to be pursued, starting from (consistent) initial conditions (Pantelides, 1988; Unger, *et al.*, 1995)

$$\mathbf{x}\Big|_{t=0}, \frac{d\mathbf{x}}{dt}\Big|_{t=0} \tag{2c}$$

and recursively determining a series of states $x(t_n)$ (*cf.* Chapter 3.2).

With increasing detail, spatial coordinates are of potential interest to the modeller as well. Then, the model must be able to represent the changes of the state quantities with respect to a change in space. In these equation systems, a variable occurs in derivative terms with respect to three independent spatial coordinates $r = [r_1, r_2, r_3]^T$. Hence, partial differential operators instead of the total differential in Eqn. 2 must be employed so that the model can be formulated as a partial differential-algebraic system equation (PDAE) system:

$$\mathbf{f}(\mathbf{x}, \frac{\partial \mathbf{x}}{\partial t}, \frac{\partial \mathbf{x}}{\partial r_1}, \frac{\partial \mathbf{x}}{\partial r_2}, \frac{\partial \mathbf{x}}{\partial r_3}, t, r_1, r_2, r_3) = 0 \tag{3}$$

Note, that any formulation with less coordinates (even Eqns. 1) can be derived form equations (3) by a formal integration over the coordinates, which shall be excluded from the model (Gerstlauer, *et al.*, 1993). In contrast to DAE models, consistent initial and boundary conditions have to be provided which is a nontrivial task (Martinson, Barton, 2000).

Distributions of a state quantity are not only of interest with respect to spatial coordinates but sometimes also with respect to substantial coordinates •ᵢ such as the size of particles in a particulate phase. In this case, not only partial derivatives occur, but also integrals that correspond to closing constraints over the distributed quantities. Hence, integro-partial differential-algebraic equation (IPDAE) system are obtained:

$$\mathbf{f}(\mathbf{x}, \frac{\partial \mathbf{x}}{\partial t}, \frac{\partial \mathbf{x}}{\partial r_1}, \frac{\partial \mathbf{x}}{\partial r_2}, \frac{\partial \mathbf{x}}{\partial r_3}, \int \mathbf{x} d\varphi_i, t, r_1, r_2, r_3) = 0\,. \tag{4}$$

A further level of complexity is added when the systems considered no longer have a fixed structure but the equations used to describe them change during the solution process. There are many examples in process engineering applications exhibiting such behaviour, such as phase changes as a function of temperature/pressure or flow pattern changes as Reynold's number increases. To describe the different modes of a system, the overall horizon of the simulation is considered to be partitioned into *N* subintervals. A different set of equations f_i as e.g. in (2) or (4) holds in each mode *i* (Barton, Pantelides, 1994):

$$\mathbf{f}_i(\ldots) = 0,\ i = 1\ldots N. \qquad (5a)$$

Each of those equation sets f_i can have its own set of variables. The boundaries of the *N* intervals are not fixed a-priori but must be determined during the simulation itself by means of monitoring so-called transition conditions *L*. These are associated to a state *i* and describe the condition for a transition to state *j* to occur:

$$\mathbf{L}_{i,j}(\mathbf{x}_i, \frac{d\mathbf{x}_i}{dt}, t) = 0, i = 1\ldots N,\ j = 1\ldots N. \qquad (5b)$$

These can be logical conditions involving logical operations such as AND, OR, and NOT. Each interval can be considered as a standalone simulation process, and must be initialised as such so that initial conditions for interval boundaries must also be computed during the simulation process. In a general formulation, this would involve equations that hold at the interval boundaries to relate quantities before and after the switching event to *e.g.* specify that the mass content of a tank is continuous over the transition:

$$\mathbf{T}_{i,j}(\mathbf{x}_j, \frac{d\mathbf{x}_j}{dt}, \mathbf{x}_i, \frac{d\mathbf{x}_i}{dt}) = 0. \qquad (5c)$$

Initially, equation systems (1)-(5) had to be coded directly into the solver framework and no other modelling facility than a programming language such as FORTRAN was available. Languages such as GAMS (Brooke, *et al.*, 1998) or MathML (Ausbrooks, *et al.*, 2001) simplify the specification of the mathematical formulation but do not provide means for structuring the possibly large set of equations.

Systems modelling languages

Systems-oriented modelling language use some sort of a *system model* as a basic concept. Such a system model can be connected to other models to yield a model structure. According to systems theory, a model is usually regarded as decomposable into a number of subordinate models, so that aggregation hierarchies of arbitrary depth can be developed. The main advantage of such a representation is, that the modeller can limit the complexity of the modelling effort by focussing on individual parts of the overall model. Further, parts of a model can be reused as such. Each system model contains a set of equations of type (1)-(5) and can be connected to others by specifying connection equations of the form

$$x_l = x_k$$

These denote that an output of some model is equal to the input of another one. Modelling languages that simplify the development of large-scale models have been available since 1967 and have improved since then. Starting with CSSL (Continuous systems simulation language, Augustin *et al.*, 1967), and its derivations like ACSL (Advanced Continuous Simulation Language, Mitchell and Gauthier Ass., 1992), Omola (Nilsson, 1993) and Dymola (Elmqvist, *et al.*, 1996) were important stages particularly well known in the control engineering community.

These domain-independent (also called multi-domain) modelling languages have resulted in a standardization effort called Modelica (Modelica Association, 2000). Modelica aims at providing a standard language that is based on object-oriented concepts to simplify model development and reuse. The basic element in Modelica is a *class definition*, which may occur as e.g. a *model class*, a *variable class* (a type), or a *module* for structurization purposes. These facilities can be used to introduce domain-specific models, as shown in the example (Figure 2).

```
   partial model SeparationStage
   MaterialFlow phase_1_in, phase_1_out;
   MaterialFlow phase_1_in, phase_1_out;
equation
end SeparationStage;
partial model StagedSeparationApparatus
   parameter Integer no_stages "the number of stages";
   SeparationStage stages[no_stages] "the stage models";
equation
end StagedSeparationApparatus;
partial model CountercurrentStagedSeparationSection
      extends StagedSeparationApparatus
equation
   // connect the phase_1 ports
   for i in 1:(no_stages-1)
      connect(stage[i].phase_1_out, stage[i+1].phase_1_in);
   end for;

   // connect the phase_2 ports in countercurrent direction
   for i in 1:(no_stages-1)
      connect(stage[i].phase_2_in, stage[i+1].phase_2_out);
   end for;
end CountercurrentStagedSeparationSection;
```

Figure 2. Example of reusable models formulated in Modelica

This Modelica fragment shows an abstract definition of a separation stage that has simply inlets and outlets for two phases. These are represented by special connector classes (not shown for the sake of brevity). The coupling of connectors is achieved by the *connect* statement as shown in the CountercurrentStagedSeparationSection model. On the other hand, we define a model for a staged apparatus by specifying a fixed number of stages. For a staged separation model with a countercurrent flow pattern an abstract topology can now be modelled without specifying what the individual pieces are. The model of the countercurrent separation apparatus inherits from the more general staged apparatus model using the *extends* keyword. Note, that all of these models have been declared using the Modelica keyword *partial*, which specifies that these models have to be further extended to be used in a concrete model. This is shown in a further example of Modelica :

```
model VLTray extends SeparationStage
   ...
equation
   // adds two phases in equilibrium
end VLTray;
model DistillationColumn
   extends CountercurrentStagedSeparationApparatus
      (no_stages=no_trays, stages=VLTray[no_stages])
   Real reflux_ratio;
   Integer no_trays "the number of trays of column";
   Condensor condensor;
   Reboiler reboiler;
equations
   // connect tower with condensor and reboiler
end DistillationColumn;
```

Figure 3. Extending generic models in Modelica

A vapour-liquid separation tray is described as a special kind of a general separation tray, again employing Modelica's inheritance mechanism using *extends*. A distillation column model is then defined as a staged apparatus with a countercurrent flow pattern where the individual stages are two-phase systems. By separating the flow pattern (concurrent, countercurrent) from the actual separation phenomena (VL/LL equilibrium) a number of different separation models can be rapidly built without specifying identical information over and over. This way of modelling demonstrates the power of the two orthogonal concepts of an *aggregation* and a *specialization hierarchy* for the reuse of models, here in the chemical engineering domain. Further, it can be seen that the definition of new constructs such as variable and model types can be used to develop libraries for a particular domain, so that the language can be tailored at least for the mathematical representation of process models. A formal specification of non-mathematical properties such as a geometric description of the model has not been achieved so far. The process engineering community also produced modelling languages, which were more or less tailored to particular requirements in chemical process modelling and simulation, but have been used in other domains as well. SpeedUp (Aspentech, 1997) as an early instance introduced an interface to a physical properties calculation engine, which relieved the modeller of the burden to code physical property models in the form of equations shown above. Further, this interface allowed to include models which were coded as algorithms rather than equations which enabled the inclusion of e.g. local property models or phase stability tests which are impossible to code in an equation-oriented form shown above.

Ascend (Allan, *et al.*, 1996), although offering mostly generic modelling concepts has primarily been used in the chemical engineering domain. It offers a powerful, object-oriented modelling language together with interesting documentation features (Allan, 1997). AspenTech's Custom Modeler (Aspentech, 2001), a follow-up product to SpeedUp has improved on the modelling language of SpeedUp, *e.g.* by adding object-oriented features such as hierarchical decomposition. So far, gPROMS (Process Systems Enterprise, 2000) is probably the only known language, which supports all of the domain-reliant features mentioned above and is also available as a commercial product. Besides the specification of model equations (1)-(5) gPROMS also allows the specification of an *operation model*, which is needed to represent external actions imposed on a process model during its operations. This so-called task language has made gPROMS a popular tool for detailed batch process modelling where the batch recipes can be represented by such an operation model. A task is composed of elementary tasks and timing structures. *Elementary tasks* describe single actions imposed on the process (changing a valve position, specifying changes in the variables), whereas the *timing structures* can be used to position the elementary tasks with respect to time, specifying sequential or parallel execution, for example. Finally, elementary tasks and timing structures can be assembled to a *task*, a complex operation procedure, which allows the reuse of an operation procedure in different contexts of a model. The following example shows how a continuously stirred tank reactor (CSTR) can be modelled together with its operating procedure. It is operated in plain batch mode, where each batch consists of a phase where two reactants are fed in sequence, and, after a subsequent reaction phase, the product is drained from the tank again.

```
MODEL Cstr
    # not shown
END
TASK FeedReactant
    PARAMETER
        Reactor AS MODEL Cstr
        Inlet AS INTEGER
        FeedRate AS REAL
        FinalVolume AS REAL
    SCHEDULE
        SEQUENCE
            RESET Reactor.Fin(Inlet):=FeedRate; END
            CONTINUE UNTIL Reactor.TotalVolume > FinalVolume
            RESET Reactor.Fin(Inlet):=0; END
        END
END
TASK Drain
    # analogous
END
```

Figure 4. Operation model specified in gPROMS

The *FeedReactant* task basically sets the reactor feed stream specified by the variable *Inlet* to the desired FeedRate and continues the (dynamic) simulation until a *FinalVolume* has been reached. Then, the feed stream into the reactor is switched off again simply by setting the feed rate to zero. The task description to drain the product from the reactor again is very similar and therefore not shown here.

```
TASK DoReactionBatch
    PARAMETER
        Reactor AS MODEL Cstr
    SCHEDULE
        SEQUENCE
            FeedReactant(Reactor IS R1, Inlet IS 1, FeedRate IS 0.1,
                TerminationVolume IS 2.5);
            FeedReactant(Reactor IS R1, Inlet IS 2, FeedRate IS 0.2,
                TerminationVolume IS 4);
        CONTINUE UNTIL  (R1.X(3) > 0.25)
        DrainProduct(Reactor IS R1);
END
```

Figure 5. Aggregation of operation models in gPROMS

Performing a reaction in batch mode is then described by reusing the more granular tasks *FeedReactant* and *DrainProduct* defined above. The batch time is determined by the concentration of product 3, which has to reach a certain level. Obviously, the *DoReactionBatch* task could be further aggregated with other tasks, e.g. for a subsequent separation of the products in semi-batch or batch mode. In addition to the reuse for Models, gPROMS provides reuse also for task descriptions based on object-oriented descriptions.

3.1.2.2 Domain-oriented Modelling Languages

A natural modelling approach for a chemical engineer is to represent his/her perception of the process in terms of concepts for communication among engineers instead of directly using mathematical equations as shown above. Such a modelling process allows the modeller to stay close to his/her *mental model* of the process so that it is easy for a domain expert to express this knowledge through a suitable modelling language. With respect to the modelling process overview sketched in the introduction, the modeller would ideally be able to directly represent the conceptual model in a tool, abstracting from mathematical details presented in the former sections.

Flowsheeting tools/languages

The most prominent instances of the class of commercial process modelling tools are flowsheeting tools such as Aspentech's Aspen Plus (Figure 6), Hyprotech's Hysis.Plant, or Simsci's Pro II. All of these offer a library of precoded models on the granularity of a unit operation. These models can be configured through a limited set of choices, but the underlying phenomena and the incorporated chemical engineering knowledge are fixed and cannot be inspected or even modified from the outside.

In the chemical engineering domain, the *structure* of a conceptual model of a plant can be found by selecting the balance volumes and the streams connecting them, first. Usually, these are chosen identical or at least close to the physical equipment boundaries so that the reuse of models for common pieces of process equipment is simplified. On this coarse level the process is described in terms of e.g. unit operation models and stream models. This correspondence between the mental model and the modelling concepts offered by the tool is probably an important reason for the success of flowsheeting tools.

After choosing a suitable model structure, the *behaviour* of each balance volume must be specified. On a conceptual level this can be done by choosing equilibrium and non-equilibrium phenomena, the coarse behaviour of the materials processed, and a suitable geometry. Flowsheeting tools offer a set of fixed choices for those items, thereby hiding a lot of mathematical and physical complexity from the user at the cost of reduced flexibility. Although suitable for process simulation and/or optimisation, this type of tools/languages cannot be used for efficient solution of process synthesis problems involving structural optimisation, for example, the determination of the optimal process flowsheet.

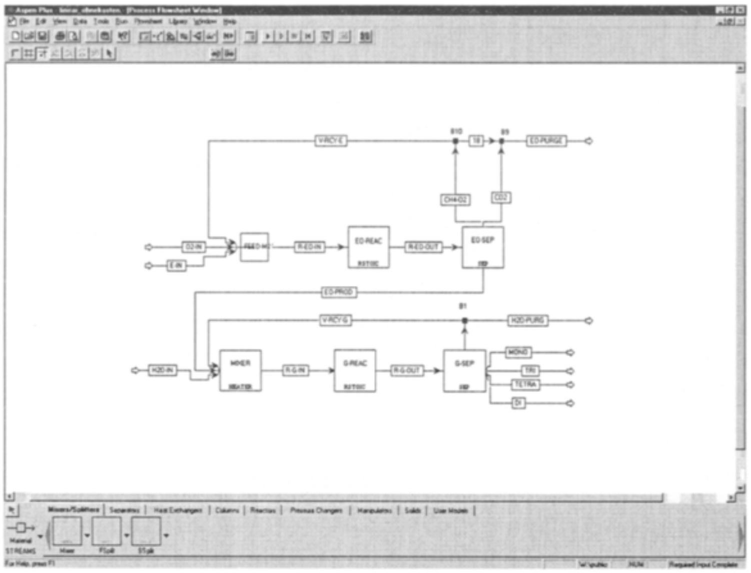


Figure 6. Aspen+ user interface with modelling concepts on a flowsheet level

Process modelling languages

A broad range of process modelling languages on a more granular level of detail with increased flexibility has been a topic of interest to several academic groups throughout the 90s (Stephanopoulos, *et al.*, 1990a; Marquardt, 1992; Drengstig, *et al.*, 1997;, Jensen, Gani, 1999). In these languages, modelling concepts are provided on varying level of granularity (multilevel modelling), usually ranging from individual phenomena such as diffusion and reaction up to models of unit operations and their connections to a plant model. This level of versatility is primarily achieved by a hierarchical decomposition approach, which allows the specification of models not only in terms of flowsheet elements, but on arbitrary levels below. As an example, a large plant model could be broken down into a reaction and a separation section together with their recycle structure. Each section can be further decomposed into logically coherent models, e.g. a distillation column with its boiler and condenser or a reactor with a pre-heater would be found on this level. Going further, the distillation column will be refined into a set of trays and each tray is in turn decomposed into a liquid and a vapour phase model, for example.

As an example, MODEL.LA (Stephanopoulos *et al.*, 1990a) introduces the *generic process unit* as a central modelling element and various subtypes thereof to span modelling contexts concerning different levels of granularity. These range from the *plant* level down to the *sub-unit* level, where the latter represents e.g. individual phases within a unit operation. For representing the process structure, MODEL.LA offers *ports,* which denote connection points between models and streams, which connect them. The process behaviour can be modelled using *variables* and *constraints.* Finally, MODEL.LA provides the notion of the modelling scope, *i.e.* through a set of conditions which must hold for the modelling context in which the model shall be used, thus supporting multifacetted modelling of processes by providing different models to be used in different modelling contexts (Stephanopoulos, *et al.*, 1990b). This language features also a partial support for documenting the model. From a formal point of view, MODEL.LA is strongly based on the notion of semantic networks, which were early approaches in knowledge representation (Russel, Norvig, 1994). A fragment of MODEL.LA describing a continuously stirred tank reactor is shown in Figure 7.

```
((JACKETED-CSTR IS-A UNIT)
 (THE COMPONENTS OF JACKETED-CSTR IS VESSEL)
 ((VESSEL IS-A SUB-UNIT)
  (THE PURPOSE OF VESSEL IS "...")
  (...)
  ENDMODEL)
(THE COMPONENTS OF JACKETED-CSTR IS JACKET)
 ((JACKET IS-A SUB-UNIT)
  (THE TYPE-OF-MODEL OF JACKET IS LUMPED)
  (THE COMPOSITION-CHARACTERISTICS OF JACKET IS HETEROGENEOUS-
COMPOSITION)
   (THE PHASES OF JACKET IS LIQUID-1)
  (THE PRESSURE-CHARACTERISTICS OF JACKET IS
     (SET.OF (HOMOGENEOUS-PRESSURE ISOBARIC)))
  (THE THERMAL-CHARACTERISTICS OF JACKET IS
     (SET.OF (HOMOGENEOUS-TEMPERATURE NON-ISOTHERMAL)))
    (...)
 (THE OUTPUT-CONVECTIVE-PORTS OF JACKET IS JACKET-OUT)
 ((JACKET-OUT IS-A CONVECTIVE-PORT) ENDMODEL)
 (THE PHASES OF JACKET-OUT IS LIQUID-1)
(THE INPUT-CONVECTIVE-PORTS OF JACKET IS JACKET-IN)
((JACKET-IN IS-A CONVECTIVE-PORT) ENDMODEL)
 (...)
  ENDMODEL)

 ENDMODEL)
```

Figure 7. CSTR description in MODEL.LA

VeDa (*Ve*rfahrenstechnisches *Da*tenmodell) used a frame-based notation to represent process engineering models from a system-theoretic perspective (Marquardt, *et al.*, 1993). VeDa uses several meta layers (Baumeister, 2001) to achieve a flexible model representation and builds on devices as major processing compartments and connections which denote the flows between devices. Recently, VeDa has become part of CliP (Bayer, et al., 2001), extending the scope to process design in general. The concept of using meta layers to organize the data model has been further extended and the extensibility has been taken care of by introducing so-called partial models (sort of modules in the data model). CliP currently comprises partial models for mathematical modelling, a function-oriented view of the process, the realisation of the process, economics and cost engineering as well as control engineering concepts.

Another approach, based on a graphical notation, has been presented by Drengstig, *et al.* (1997). TRAV (Transport, Reaction, and Accumulation View) uses a graphical notation of not only the process structure (or topology), but also of the phenomena related to energy and chemical species. Different symbols are available for accumulation, transport and reaction/generation and related by arrows indicating flows among these effects.

Discussion

In the field of software engineering and construction, object-oriented information modelling (Alhir, 1998) turned out to be a useful technique to represent domain-specific concepts and their relationships. Although all of the languages mentioned have an implicit information model, few development efforts were explicitly based on such an implementation independent description of the process engineering domain. Rather, the preferred approach was to directly define a human- and machine-readable representation as text, usually defined by some *grammar* (Aho *et al.*, 1986), along a particular tool being developed. This has led to a variety of languages which do all express tool-specific concept in one or another way and are thereby insufficient to use for communicating models across a broad range of model-based applications. Recent standardization efforts such as Modelica or MathML aim at a completely tool-independent representation of models so that they offer the potential to integrate a large number of applications for modelling and simulation.

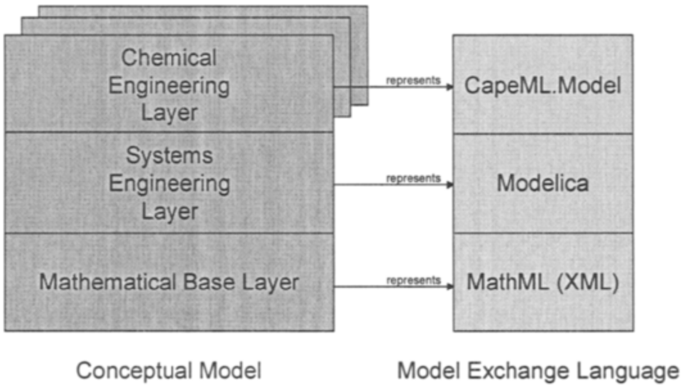


Figure 8. Relationships of conceptual models and languages

In the future, we would favour that a three-level information model together with a suitable set of model representation languages emerges (*cf.* Figure 8). This vision encompasses a mathematical base layer, which provides a rich set of generic mathematical modelling concepts on the lowest level. The systems engineering layer introduces the systems-oriented concepts decomposition and coupling based on object-oriented principles. On top of these, domain-specific extensions such as a chemical engineering layer are to be defined on the highest level by introducing specific concepts (*e.g.* for phenomena or unit operations) using the object-oriented principles defined on the layer below.

A domain model for the concepts on these layers proposed could be specified using UML. For each level, standardized representations suited to different perspectives can then be defined as representations to facilitate tailored exchange of information among humans or tools, textual as well as graphical ones. Using XML, the MathML standard could become a standard representation of the mathematical view of a model. Modelica could serve as a standard representation for a systems-based, multi-domain modelling approach offering rich capabilities of object-oriented principles. Finally, a family of languages tailored to individual domains is required to enable an efficient model development in any of these domains. An XML-based approach towards such a body of standards with a focus on process engineering models called CapeML.Model is currently undertaken as part of the Global CAPE-OPEN project (Braunschweig, *et al.*, 2000).

Whereas the process flowsheeting languages and corresponding concepts are in widespread use today, the more academic process modelling languages have not found much interest in industrial applications so far. Two important possible drawbacks are that

- although focusing on process engineering concepts, a good mathematical understanding of modelling is still required because a complete generation of a process model from a specification of physical and chemical concepts has not been achieved up to date, and
- implementations of these languages (see further below) have not been available in industrial strength software.

More recent approaches still have to prove their theoretical advantage as opposed to generic modelling languages. Generic modelling languages such as gPROMS, ACM, or GAMS have proven to be powerful, flexible tools besides the more easy to use flowsheeting process modelling tools.

3.1.3 MODELLING TOOL FUNCTIONALITY

Rapidly, computer-aided support has been identified as an important means to speed up the modelling process. The intention of a chemical process modelling tool is to support a certain modelling methodology, where a methodology in general is commonly understood as a combination of a certain work process and a notation of domain-specific concepts as introduced above (Oliver, *et al.*, 1997; Jarke, Marquardt, 1995). Both, process and notation should be tailored to each other, the process describing when to use which concept for which purpose.

However, as of today it is impossible to automate the complete modelling process. Instead, some parts of the modelling process are formally understood, whereas others must be considered as a *creative activity* and therefore require user interaction (Jarke, Marquardt, 1995; Lohmann, Marquardt, 1996). Hence, it is important to identify those parts of the process, which can be automated in order to formalize and implement them. As an example, finding the balance volumes in a chemical process is an activity, which requires a bit of experience and an understanding of the model. On the other hand, setting up the balance equations given the balance envelopes, the phenomena occurring in them, and their interconnections is a pretty straightforward task, which can easily be automated (Gerstlauer *et al.*, 1993; Jensen, Gani, 1999).

The following sections present an overview of those parts of the modelling process, which are well understood so that they could be formalized, making them well-suited for implementation in a tool. We structure the functionality of a modelling tool according to its support to *create and manipulate* conceptual and/or mathematical models, to *analyse* them for *e.g.* correctness and solvability, and to finally *translate* them into an efficient representation suitable for the solver. These issues will be discussed in more detail in the subsequent sections.

3.1.3.1 Model Creation and Manipulation

Functionality to actually perform the creation of the model and certain modifications of it are very interesting as they do not just automate some tedious task but actually introduce domain knowledge as part of the modelling tool functionality. Hence, tools relying on such a paradigm can be classified as *knowledge-based.* The knowledge-based systems are those that in addition to the features found in the generic modelling languages, contain "built-in rules" that *assist in setting up model equations.* Thus, the goal of the knowledge-based systems is to let the user describe new models either through phenomena-based information and/or through experimental data and let the "built-in rules" abstract this description into a set of equations. In the development of the model equations, several types of knowledge bases may be used, such as:

- Fundamental model objects
- And-Or graphs
- Algorithms that screen for closure constraints
- Algorithms that assist in the development of new empirical/semi-empirical models

Also, other knowledge representation mechanisms can be used to aid the model construction step such as an expert system. However, few of those have actually been applied to chemical process modelling. Sørlie (1990) for example employed a blackboard architecture in order to assist the modelling process by a predefined set of rules which are supervised by a scheduler.

Fundamental model objects

Fundamental modelling objects represent core concepts from the application domain of chemical engineering. In many knowledge-based process modelling tools, these objects are associated with algorithms that assist the user in performing a phenomena-based description of a new model. This phenomena-based description may either be used by searching for model equations that match the specified description (search-based methodology) and/or it may be used to eliminate parts of a reference model that is stored in a model library, thereby, incorporating the concepts of model reuse and model transformation.

Fundamental modelling objects can be classified in terms of *fundamental region objects* and *fundamental connection objects,* similar to devices and connections introduced by Marquardt (1996). This classification is based on the concept of dividing the model equations into those that represent a defined boundary (region) in terms of balance equations, constitutive equations and constraint (or conditional) equations. Also, two or more boundaries or regions may be interconnected through connection equations. New fundamental objects can be derived from equations that are related to objects belonging to an existing set of reference (fundamental) model objects. Ideally, these fundamental modelling objects may also be predefined in terms of a library. The elements of this library, *fundamental modelling templates,* can control the addition of user-defined equations, for example, when the library (or reference) constitutive equations belonging to a reference model object are not suitable. A set of rules is therefore necessary to generate (retrieve) the necessary equations from the reference model objects and to check for consistency between boundary (mathematical/phenomena) description and generated/retrieved equations. A "search-based" methodology therefore need to be applied to assist the user in locating and retrieving the most appropriate reference fundamental objects among the feasible alternatives available in the library. Figure 9 shows the classification of the model equations belonging to a fundamental model object.

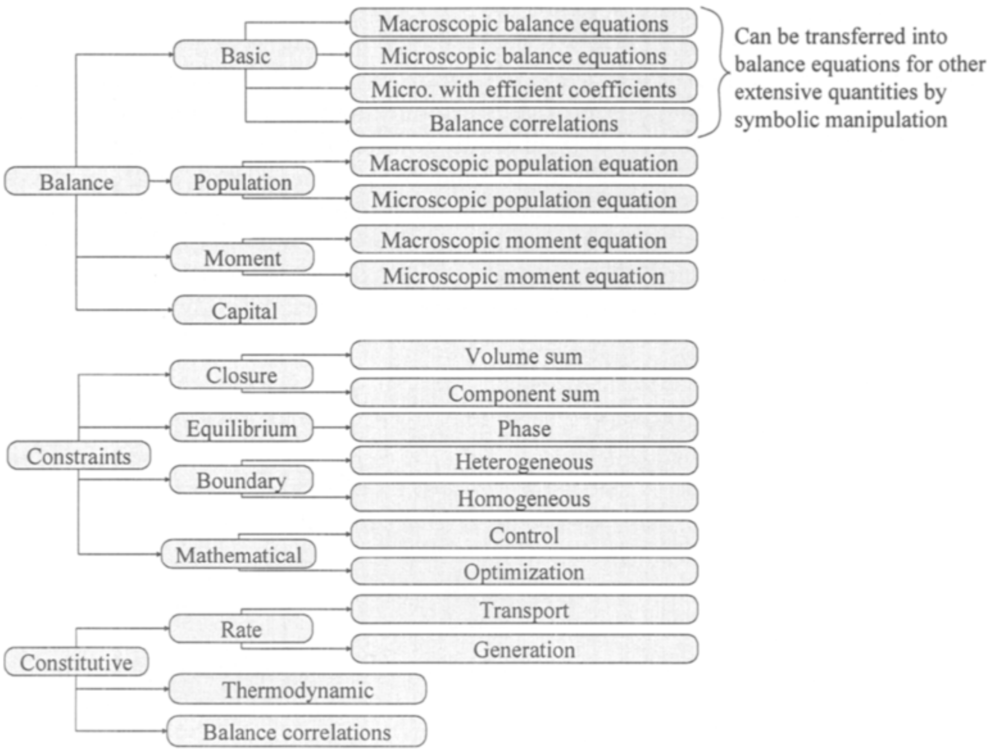

Balance
Basic
Macroscopic balance equations
Microscopic balance equations
Micro. with efficient coefficients
Balance correlations
Can be transferred into balance equations for other extensive quantities by symbolic manipulation
Population
Macroscopic population equation
Microscopic population equation
Moment
Macroscopic moment equation
Microscopic moment equation
Capital
Constraints
Closure
Volume sum
Component sum
Equilibrium
Phase
Boundary
Heterogeneous
Homogeneous
Mathematical
Control
Optimization
Constitutive
Rate
Transport
Generation
Thermodynamic
Balance correlations

Figure 9. Classification of fundamental modelling object equations

A new model can be generated by following the inheritance path for each class of equations (balance, constitutive and constraint) according to the boundary description. The end of each path is connected to a reference model object. If the reference model object matches the boundary description, it is retrieved, otherwise, it must be modified or a user-defined reference model object has to be introduced.

And-Or graphs

A similar, yet fundamentally different approach, is the representation of process modelling knowledge using an And-Or graph (Bogusch, Marquardt, 1997). Whereas the approach presented in the previous section uses a specialization hierarchy to represent knowledge of the process engineering domain, And-Or-Graphs use a composition approach to represent the relationships of variables and equations. Such a graph is partitioned into a set of variable type nodes and a set of equation type nodes (Figure 10). Directed edges from variables to equation describe alternative refinements for a variable type (Or-edges). One of these equations has to be selected as part of the process model. As an example, a diffusive mass flow J can be refined by a variety of phenomenological relations such as Fick's law of diffusion, Stephan-Maxwell equations, *etc.* Directed edges from equations to variables describe which variables are constrained by an equation. Each of the variables in an equation occurring in a process model must either be further refined using the And-Or graph or it must be specified as a fixed parameter of the model. Based on such a representation of domain knowledge, model development can be seen as an activity of finding a suitable path through an And-Or-Graph, where a decision according to modelling assumptions and objectives must be made at each Or-branch. In addition to the specification of the graph, a set of *pruning rules* can be useful in order to eliminate choices, which are not suitable in the current modelling context. The choice of geometric relations should be constrained by the choice of the chosen geometry, for example.

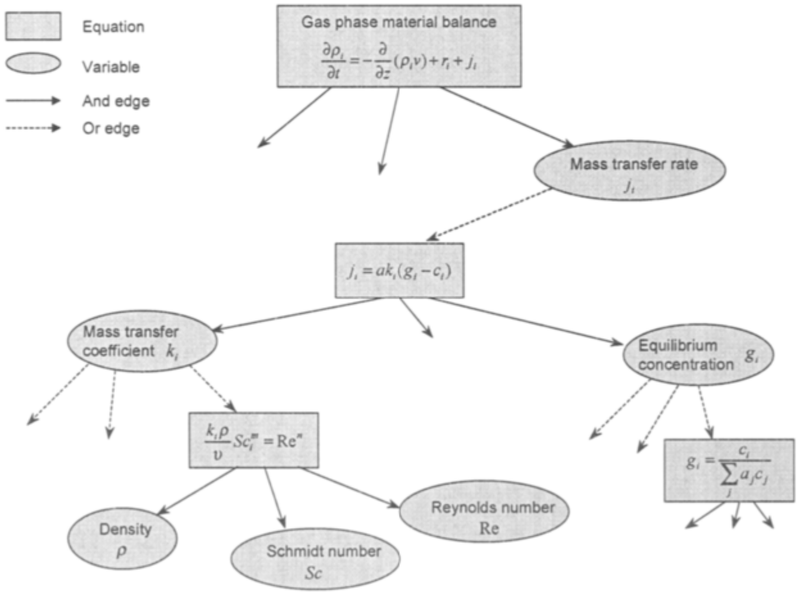

Equation
Variable
And edge
Or edge
Gas phase material balance
$\frac{\partial \rho_i}{\partial t} = -\frac{\partial}{\partial z}(\rho_i v) + r_i + j_i$
Mass transfer rate j_i
$j_i = a k_i (g_i - c_i)$
Mass transfer coefficient k_i
Equilibrium concentration g_i
$\frac{k_i \rho}{\upsilon} Sc_i^m = \mathrm{Re}^n$
$g_i = \frac{c_i}{\sum_j a_j c_j}$
Density ρ
Schmidt number Sc
Reynolds number Re

Figure 10. Part of an And-Or-Graph representing domain knowledge

Screening for closure constraints

Such algorithms screen the developed equation system for variables that should be constrained with closure equations. That is, if all the variables within a closure equation are in the equation system, then this kind of algorithm should add the closure equation to the equation system in order to indicate that the variables within the constraints are not independent.

- Development of new empirical/semi-empirical models
- The following algorithms may be used to assist in the development of new empirical/semi-empirical models:
- Reference constitutive relations – used to create a new phenomena model
- Dimension analysis – the new set of equations need to be mathematically consistent

- Statistic/plotting tools that assist in analysing experimental data in order to determine an empirical constitutive relation (phenomena model)
- Model analysis – identify linear equations, decompose equations into a set of summation terms, identification of terms through intensive variables, *etc.*

3.1.3.2 Model Analysis

Model analysis is of high interest during model development. Model analysis tasks can be distinguished in mathematical criteria (which are derived *e.g.* from solvability criteria and therefore partially depend on the solver type being chosen) and domain-specific criteria, which are relevant in a physical or chemical context.

A number of those criteria have been listed by Jensen (1998), Bogusch *et al.* (2001), or Piela (1989). Starting with physically motivated criteria, there is the analysis of physical dimensions of the variables occurring in an equation. Obviously, the domain of the terms on the left and right hand side must be identical, i.e. mass per time for a mass balance. Further, the list of substances within each balance volume can be constructed by propagating them along the structural connections in the model if permeability of walls etc. is provided by the modeller (Bär, Zeitz, 1990). Several mathematical problems can already be identified on a process engineering level. If *e.g.* too many balance equations/closure equations are set up, the resulting model will have a rank deficiency because two equations will be linearly dependent.

The analysis features which fall into the mathematical domain a far more comprehensive. Here, degree of freedom analysis, structural solvability and index analysis (Jensen, 1998; Bogusch *et al.*, 2001; Russel & Gani, 2000; Unger, *et al.*, 995) have been developed and implemented in several chemical process modelling tools.

3.1.3.3 Model Transformation

All of the model formulations presented above need an additional solution algorithm to actually find a solution from some given input such as the parameter values for an algebraic equation system and the initial conditions in case of a differential-algebraic equation system. Without the solver algorithm, the model as formulated above can only be used to check whether a given state is among the possible states of the system under consideration. This functionality of the model is used by the solver to iteratively determine a solution of the model, together with derivatives of the model equations f with respect to the states x in order to guide the iteration process. In order to connect the model to such an algorithm, the model equations and derivatives must be available as an executable piece of code. Such a model representation is also referred to as *procedural model representation*, as opposed to a *declarative model representation* where the model is given as data, which does not exhibit any functionality. We call the transition from the declarative to the procedural representation a *model transformation* (of a specific kind).

The model resulting from the modelling process is often not well suited to be used by the solver directly. A number of simplifications and optimisations are usually applied to the model as it is transformed. For example, a *discretisation* based on the method of lines approach can be employed to make PDAE systems solvable by standard DAE solvers (Pfeiffer, Marquardt, 1996). *Optimisations* of the equation structure can be performed, e.g. by reordering equations, eliminating explicit equations, or factoring out common equation structures or subexpressions (Carpanzo, Maffezoni, 1998; Allan, Westerberg, 1999; Morton, Collingwood, 1998). The final step of a modelling tool is to transform such a declarative model into a procedural representation to be used by the solver. Generally, there are two distinct approaches.

First, the model can be translated into some programming language (*e.g.* FORTRAN or C++), effectively generating the procedure to be used by the solver (Räumschüssel, 1997). During this step, automatic derivative techniques (Bischof, *et al.*, 1997; Grievank, *et al.*, 1996) can be used to supply Jacobian and sensitivity information required for the solution step. The resulting code must then be compiled and linked together with the solver or it is transformed into a reusable software component. Ideally, such a component is equipped with standard interfaces according to the CAPE-OPEN standard so that it can be used in combination with any standard-compliant solver in a plug-and-play manner.

An alternative approach to achieve a procedural model representation is to transform the model into a computation tree, where the operations are represented as nodes and the variables are represented as leaves of the tree (Keeping, Pantelides, 2000). This tree can then be evaluated in depth-first or breadth-first manner resulting in values of the residual expressions. Additionally computing the chain rule along the computation of the tree also yields Jacobian information. A possible advantage of this approach is that domain-specific knowledge can be taken into account in order to speed up the transformation process. This is particularly important as process models quickly grow into sizes where generic compiler optimisation capabilities get stuck due to the sheer size of the model (Allan, Westerberg, 1999). Further, it does not require the user to have a particular compiler installed on his/her machine.

The MoT modelling toolbox (Russel and Gani, 2000) also provide feature to translate a model into a programming language such as C++ according to a specified format and/or direct solution of the model equations without the need for a particular compiler installed on the machine.

3.1.4 MODELLING TOOL ARCHITECTURES

A set of concepts useful for chemical process modelling has been presented and functionality, which is suitable for an implementation has been discussed above. This section will now concentrate how these two can be successfully used in a modelling tool implementation.

The coarse organization of the required functionalities into a software tool is generally called architecture. Architecture of a software system is a term on which most people have an agreed understanding until it comes to a precise definition. There seems to be a common understanding of architecture as a blueprint of a software system, a description that facilitates the construction and understanding of software artefacts. Further important objectives to be achieved by an architecture are the management of change and the mediation of different views of the stakeholders involved in a software project (Buschmann, *et al.*, 1996; Kruchten, 1995). By common sense, architectures consist of building blocks (called modules or components), their interfaces, and the relationships between the building blocks (Alhir, 1998). Here, the limit of common agreement is reached. There are ongoing debates about whether an architecture is conceptual (providing a basic understanding of a possible realization of the system) or implementational (describing a concrete implementation of the system). Also, it is not agreed whether an architecture also describes the control flows within the systems or distinguishes different roles of its building blocks.

As it is impossible to describe a modelling tool architecture in detail without a concrete example, we will focus here on the properties of architectural patterns. Such a (architectural) pattern *addresses a recurring design problem that arises in specific design situations, and presents a solution to it* (Buschmann, *et al.*, 1996; Gamma, *et al.*, 1994). Being a generic rather than a specific solution makes patterns interesting to discuss and compare modelling tools apart from specific implementations. Here, we will focus on general properties in terms of functionality, integration aspects and extensibility, which emerge from these patterns.

In general, two broad classes of modelling tool architectures can be distinguished. The first class is batch-oriented and requires a complete problem specification from the beginning and a description what to do with it. All remaining activities will be performed automatically without further user interaction. On the contrary, the interactive modelling tool aids the user in constructing the model and specifying the description of what to do with the model. Open process modelling tools finally are tailored towards integration tasks and easily fit into engineering environments.

3.1.4.1 Batch-oriented Modelling Tool Architectures

A *batch-oriented modelling tool* can be characterized as a single tool that reads a model definition from some input file (which is usually written using some text editor). Then, a sequence of steps with this model input is performed until the final goal, a solvable model formulation, is achieved. Often, the final solution step is even integrated into the tool itself. Such an architectural pattern, where data is processed in a number of stages, one after the other, is commonly called a *pipes and filters* architectural pattern (Buschmann, *et al.*, 1996). Each processing step for data is called a *filter* because it consumes input data and enriches, refines, or transforms it, and finally outputs the modified data again. Further, so-called *pipes* are established; they are responsible to establish a communication link between filter components and to synchronize the execution of active filters.

A pipes and filters architectural pattern for process modelling and simulation tools is depicted in Figure 11 and will be explained subsequently. A tool, which understands a textual model representation (*e.g.* in Modelica or gPROMS language), requires a pair of a scanner and parser at the beginning of a suitable pipes and filters combination. These tools stem from the compiler construction domain (Aho, 1986). In fact, many batch-oriented process modelling tools have a lot in common with a compiler. The *scanner* is required to separate the input language into syntactical elements, such as keywords, symbols, etc. Based on this syntax tree, the *parser* recognizes constructs defined in the language and assembles a token tree to describe the input in terms of the grammatical elements of the modelling language that are usually described in extended Backus-Naur form (EBNF) (e.g. Modelica Association, 2000). Although scanner and parser could be assembled in a pipes and filter manner as well, compiler construction has optimised the interaction of these two tools so well, that they can be considered as a single unit within the overall architecture.

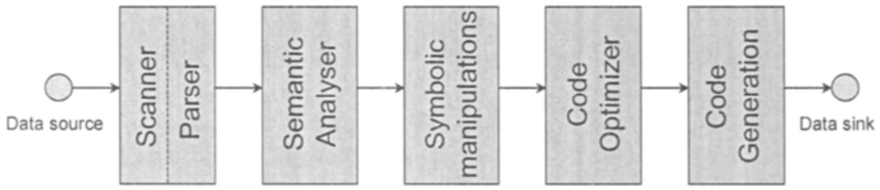


Figure 11. Pipes and filters architecture of modelling tools

The token tree obtained from the parser is usually further analysed for semantical correctness, this can be considered as a first analysis step. The parser only verifies that the language elements are used in a proper configuration (*e.g.* an operator must occur between two variables), but does not check e.g. whether variable names being referred to are actually defined. This is accomplished by a separate module, a *semantic analyser* as shown in Figure 11. Afterwards, modifications such as symbolic discretisation can be performed and the resulting code can be optimised with respect to numerical performance and robustness. Then, a model transformation in the sense of a code generation step (similar to a compiler) can be applied. The final *code generation* filter could e.g. output the contents of a Modelica model e.g. as a set of C++ class definitions which can then be compiled into an executable chunk of code or a CAPE-OPEN compliant component (Geffers *et al.*, 2001) to be evaluated by the solver. In terms of compiler construction the final stage where code is generated would be referred to as a *back end.*

An interesting property of such an architectural style is that it is often possible to reuse individual filter components or to reconfigure them in different ways. As an example it is common in compiler construction to use different back ends for different target formats. Using this approach, one could build a batch-oriented modelling tool that can, given a model in a certain input representation, output the model in a variety of different target platforms. Figure 12 shows an extended modelling tool architecture according to the pipes and filters architecture which uses different back ends to generate code not only for C++ but also for GAMS and Modelica. Also, different analysis and/or optimisation stages may be employed, depending on the problem to be solved.

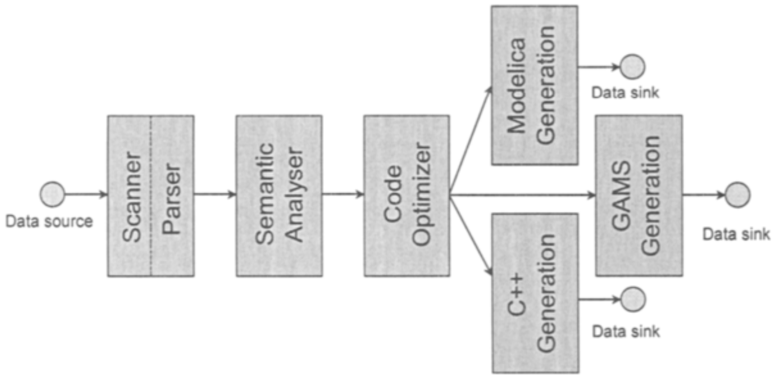


Figure 12. Pipes and filters architecture with different back ends

Tools using an architecture resembling the pipes and filter structure are often developed along a particular modelling *language*, such as gPROMS or SpeedUp. This language is implemented in the scanner/parser module and describes the interface of the tool to the user. With respect to the functionality classification mentioned above, the pipes and filters architecture focuses on the aspects of model analysis and model transformation but usually provides only weak support for model creation and manipulation.

3.1.4.2 Interactive Modelling Tools

Interactive modelling tools on the other hand work quite differently. The main difference is that they do not require the user to specify the full model right from the start, but they support him/her also during the model creation and manipulation process. Hence, an interactive modelling tool can be applied to a process where generative functionality can be applied only partially to the overall modelling process. This feature makes interactive modelling tool architectures particularly well suited for the use of process modelling languages. As already mentioned above, process modelling languages often require further user input since it is impossible to generate complete process models from a simple domain specifications.

An interactive modelling tool frequently switches between a user-driven and an automated mode. As long as user input is required, the user drives the modelling process. Then, he/she can launch automated functions, which can make quick progress along the modelling effort. These automated functionalities usually cover model analysis and the transformation of the model into a form suitable for the solver. If any of those steps fail, the modeller will have to go back into the model manipulation mode to correct any problems reported by the tool. Also, a failing solution process might force the modeller to reconsider his model.

In recent times, also the actual generation of model equations has been further elaborated (*cf.* Section 0). Further, it is important to note that an interactive tool can provide analysis functionality also in the user-driven mode. The tool can respond to any user input and provide immediate feedback if any incorrect information has been entered, such as a reference to a variable unknown so far. Although a tool may not now how to write correct equations on its own, it can very well identify equations, which are incorrect with respect to their physical or mathematical dimensions.

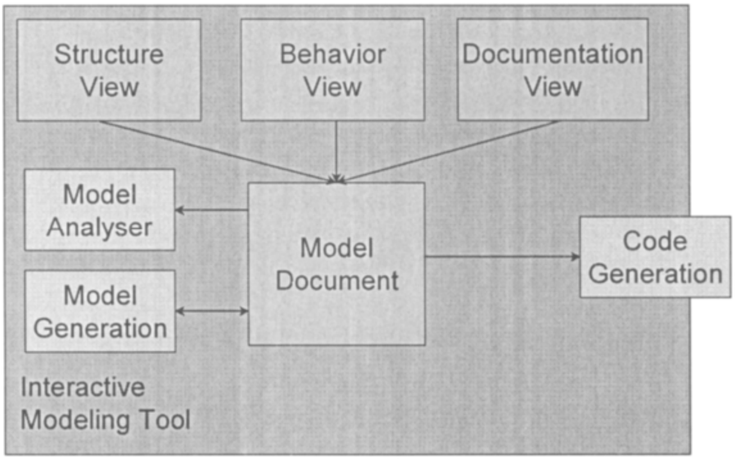


Figure 13. Document-view architecture of a modelling tool

A useful architectural pattern to describe an interactive modelling tool is the so-called *document-view architecture* (Buschmann, 1996) as shown in Figure 13. It consists of a central storage of the data being manipulated by the tool called a document and one or more views which offer representation and manipulation facilities of the data. The *model document* is usually an implementation of a set of modelling concepts as presented above. *Views* for the model aspects structure, behaviour, or documentation are often implemented using a graphical user interface and a graphical language to represent the modelling concepts employed. These view/manipulation modules can be distinguished into model manipulation, analysis and model transformation features according to section 3.1.3.

It should be further noted that the implementation of an interactive modelling tool is far more complicated because analysis steps have to be applied potentially after each user interaction and things analysed to be correct may be no longer correct after the next mouse click, whereas in a batch-oriented tool correctness needs to be verified only once.

A further difficulty in the development of interactive modelling tools arises in the integration with a simulator. In terms of the pipes and filters architecture presented above, the interactive modelling tool could assume the role of a data source for the subsequent simulation tool, thereby integrating the pipes and filters architecture. In practical applications, the interactive modelling tools rely on the generation of a model as a text file, which is then used in a batch-oriented tool for model transformation and simulation. Many of the academic initiatives in developing interactive modelling tools work this way, *i.e.* ModDev (Jensen & Gani, 1999), the recent work around MODEL.LA (Bieszczad, 2000), or ModKit (Bogusch, *et al.*, 2001) which are all able to output a model definition file by means of a *code generation step* which is suitable for use in e.g. SpeedUp or gPROMS. But when the simulator flags a difficulty or error (such as a numerical singularity during the integration of a dynamic model), it is difficult for the interactive modelling tool to take over control again and support the user in circumventing the problem.

Interactive tools are usually developed as monolithic applications, e.g. the interactions between the different view modules are implicitly coded into the tool. Hence, it is impossible for others to extend the tool or to use it as a part of some automated process in a larger software environment. In terms of lifecycle integration aspects, where such tool integration challenges arise frequently, it would be favourable if the model data and the control over the tool could be accessed from the outside (Nagl, Marquardt, 2001; Wasserman, 1990).

ModDev (Jensen, 1998) is a knowledge-based modelling system that is integrated with ICAS (Gani *et al.* 1997) and is able to generate process models based on the description given above. It employs a graphical user-interface to convert the modeller's perception of the process in terms of phenomena, accumulation, and constraints, and aggregates them to form models of the unit operation defined in terms of a boundaries, connections, and states. In ModDev, fundamental modelling objects (as explained in section 0) are used to create generic building blocks. The fundamental modelling objects and the generic building blocks are then aggregated to form the desired process model. The equation set representing the process model is then analysed and translated for integration with a solver. The translated model may be used as an equation set in equation-oriented simulators or as a block in a flowsheeting system.

ModKit (Bogusch, *et al.*, 2001) is an interactive modelling tool based on the process modelling language VeDa (Marquardt, *et al.*, 1993) that specifies relevant domain knowledge. In ModKit several interactive *tools* contribute to supporting the modeller in setting up the model equations. These tools comprise editors for the model structure, for properties such as geometry or phenomena, for model equations and variables, and for the model documentation. In addition, analysis tools such as structural solvability and index analysis are made available including a graphical representation of the equation system structures (incidence matrices) involved. Besides these tools to manually enter the model topology or equations, ModKit offers a so-called *guidance mode*, where predefined parts of the modelling process are used by the tool to guide the modeller through the modelling process, proposing him creation, modification, and analysis steps wherever suitable. The model resulting from the modelling process can be exported in gPROMS and SpeedUp formats.

Further academic initiatives to be mentioned are Model.La (Bieszczad, 2000) or Design-Kit (Stephanopoulos *et al.*, 1990a) – these are also knowledge-based tools that assist the user to generate/develop models based on a description of the process, boundary, phenomena and so on.

On the commercial side, interactive modelling tools have not attracted too much attention so far. Aspen Custom Modeler (AspenTech, 2001) is one of the few commercially available modelling tools, which offers interactive functionality to specify and analyse a process model. However, no actual functionality to generate any kind of model equations (apart from connection equations describing the process topology) has been made available on a commercial basis.

Open modelling tools

As a consequence open architectures are a current topic in the development of modelling tools (Braunschweig *et al.*, 2000). Such environments provide rich interfaces to enable external pieces of software to be plugged in to allow a flexible configuration of the tool according to the user's needs. Further, these tools often provide their functionality to the outside through a so-called application programming interface so that they can be used as a part of an integrated engineering process where models are involved as shown by Gani et al. (1997) or Hackenberg, *et al.* (2000). Middleware such as databases access technology (*e.g.* ODBC, ODMG, or JDBC standards) or distributed object computing (such as DCOM and

CORBA, *cf.* Chapter 4) have proven an important means to provide functionality across the boundaries of tools, enabling modelling tools to be tightly integrated with other engineering processes as part of a software environment.

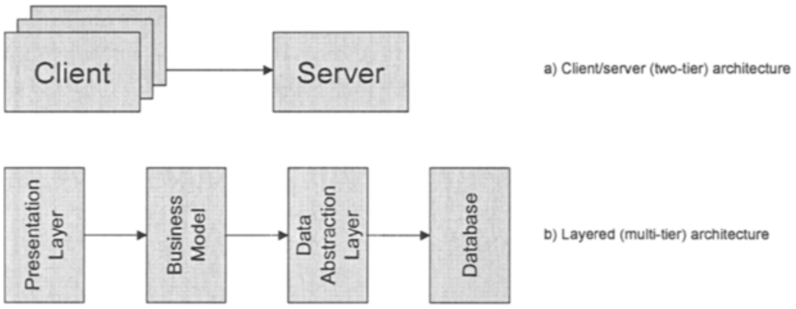

Client
Server
a) Client/server (two-tier) architecture
Presentation Layer
Business Model
Data Abstraction Layer
Database
b) Layered (multi-tier) architecture

Figure 14. Client/server and multi-tiered architectures

The resulting architecture resembles (in its simplest form) a *client-server* structure, where e.g. a set of client programs share information through a central server (Figure 14). Often, the server provides only data access (as it is the case using ODBC) so that clients must implement everything ranging from presentation to business logic. As a client can in turn provide services to other modules, such an architecture can be further structured into several layers (also called *tiers*), where each layer should provide a well defined abstraction of some service. Separating e.g. the presentation services, business (modelling) logic, data maintenance, and persistence leads to a *multi-tiered architecture.* A simple client-server relationship is also referred to as a *two-tier architecture.*

As an example, the *model repository* Rome (von Wedel, Marquardt, 2000) offers its functionality to maintain and manipulate process models to its environment based on a two-tier architecture. The bottom layer uses an object-oriented database for persistence, whereas the business model layer presents modelling concepts to the environment via a comprehensive set of CORBA interfaces.

In order to gain more flexibility within such a layered architectural model, a *component architecture* can be employed (Mowbray, Malveau, 1997). The building blocks of this architecture are smaller-grained components (as compared to the layered architecture) with well-defined but limited functionality, which exhibit client/server relationships among each other. These services ideally share a standardized interface definition that allows a reconfiguration of the system. The set of services (*e.g.* model analysis functionality) can be extended in a simple manner and services can be exchanged for better or cheaper alternatives. The flexibility of such a component architecture is leveraged by the dynamic reconfiguration possibilities provided through the component software techniques.

The recent implementation of ModKit uses such a component architecture to provide a set of editors to the modeller which can be flexibly extended or reconfigured without further modifications of the remainder of the modelling environment (Marquardt, *et al.*, 2000). ModKit uses the model repository Rome for persistence and basic model manipulation functionality and is being controlled by a modelling process guidance system so that a four-tier architecture results, where the interactive layer can be described in terms of a component architecture.

Such an architectural model fully satisfies the requirements on open modelling system with respect to flexibility, extendibility, and information sharing among modellers. The dynamic flexibility of components allows to tailor the system to the particular needs of its users and the networking capabilities of recent software technologies provide an essential basis for connecting modellers and model users working remote from each other. Hence, the modern software technologies must be considered as a key enabling technology towards the development of an open model server as suggested by Britt and Pantelides (1994).

3.1.5 SUMMARY OF PROBLEMS & CHALLENGES

Compared to the first implementations of tools for process modelling and simulation an enormous amount of progress has been achieved. Today's modelling tools provide modelling languages, either based on process engineering concepts or mathematical perspectives that are suited to represent structural and phenomenological aspects of chemical process engineering. With respect to the requirements sketched above, a number of issues must be considered still open today.

The set of modelling concepts should be improved towards a *formal theory* to automatically generate, manipulate, and reason about models. Such a theory will have a strong impact on the capabilities of modelling tools, but is still an open research issue. It will enable inexperienced modellers to effectively use model-based techniques for a wide range of applications. A step forward towards such a theory has been proposed by Hangos and Cameron (2001a) very recently.

Informal documentation about the rationale and intention of models is not satisfactorily maintained and exploited by modelling tools. All the modeller can do is attach some comments to his model. Using some sort of methodology such as IBIS (Rittel, Kunz, 1970) would be at least desirable, but is not sufficient. In the long term, it must become comprehensible why modifications of a model have been made. Further, it must be specified whether the expected outcome from these modifications has been met.

Regarding modelling functionality, improvements must be made by formalizing the *modelling process* and identifying further parts to be supported by tools in an automated manner (Lohmann, Marquardt, 1996; Marquardt, Jarke, 1995). This will lead to a reuse of modelling knowledge, based on experience made in other modelling contexts.

The vision of a *model server* (Britt, Pantelides, 1994) must be further refined to provide shared access to models between modellers as well as to enable a model to be obtained in different representations automatically. Modern software technologies can be considered sufficiently mature to implement such an environment complying with a model server architecture. However, this requires a major contribution from software vendors.

One of the major problems to be tackled on the way to a model server will be the *maintenance of reusable process model libraries.* Frequent modifications necessary due to the iterative engineering processes employed in process design and retrofit lead to a large number of models, which differ only slightly. All of these must be made available in a comprehensible manner and structured properly in a library so that they are available for reuse on demand of the modeller. This requires on the one hand powerful algorithms to systematically deduce the structure of the model library based on the information specified in the models as well as useful search mechanisms, which find a model, tailored to the current modelling context.

Open standards such as Modelica and CAPE-OPEN are important means to achieve an economic implementation of a model server and to enable its interoperability with a wide range of clients, such as simulators and interactive or batch-oriented modelling tools. Customers can more easily plug in their own solutions and research results can be integrated and assessed more quickly.

3.1.6 REFERENCES

Aho, A.V.: *Compilers – Principles, Techniques, and Tools.* Addison-Wesley, 1986.
Alhir, S. S.: *UML in a Nutshell.* O'Reilly, 1998.
Allan, B. A., A. W. Westerberg: Anonymous class in declarative process modeling. *Ind. Eng. Chem. Res.* 38, 692-704, 1999.
Allan, B.A.: *A More Reusable Modeling System.* Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA (USA), 1997.
Allan, B.A., V. Rico-Ramirez, M. Thomas, K. Tyner, A. Westerberg: ASCEND IV: *A portable Mathematical Modeling Environment.* Carnegie Mellon University, ICES Technical Report, October 1996.
Aspentech: *Aspen Modeler 10.2 – Reference.* Aspen Technology, Inc., Cambridge, MA (USA), 2001.
Aspentech: *SpeedUp User Manual.* Aspen Technology, Inc., Cambridge, MA (USA), 1997.

Augustin, D., J. Strauss, M. Fineberg, B. Johnson, R. Linebarger, F. Sansom: The SCi Continuous System Simulation Language (CSSL), *Simulation*, Vol. 9, No. 6, December 1967.

Ausbrooks, R., S. Buswell, S. Dalmas, S. Devitt, A. Diaz, R. Hunter, B. Smith, N. Soiffer, R. Sutor, S. Watt: *Mathematical Markup Language (MathML) - Version 2.0.* Available online at: http://www.w3.org/TR/MathML2/, 2001.

Backx, T., O. Bosgra, W. Marquardt: *Towards Intentional Dynamics in Supply Chain Conscious Process Operations.* Contr. To: FOCAPO '98, Snowbird, Utah, 5-10.7., 1998.

Barton, P.I., C.C. Pantelides: Modeling of combined discrete-continuous processes. *AIChE Journal* 40 , 966-979, 1994.

Bär, M.; M. Zeitz: A knowledge-based flowsheet-oriented user interface for a dynamic process simulator. *Computers and Chemical Engineering,* 14, 1275-1283, 1990.

Baumeister, M.: *Ein Objektmodell zur Modellierung und Wiederverwendung verfahrenstechnischer Prozessmodelle.* Dissertation, RWTH Aachen, 2001.

Bayer, B., C. Krobb, W. Marquardt: *A Data Model for Design Data in Chemical Engineering.* Technical report LPT-2001-15, Lehrstuhl für Prozesstechnik, RWTH Aachen, Available online at: http://www.lfpt.rwth-aachen.de/Publication/Techreport/2001/LPT-2001-15.html, 2001.

Biegler, L.T., D. Alkaya, K. Anselmo: Multi-solver Modeling for Process Simulation and Optimization. In: M.F. Malone, J.A. Trainham, B. Carnahan (Eds.): Foundations of Computer-Aided Process Design, *AIChE Symp. Ser.* 323, Vol. 96, 125-137, 2000.

Bieszczad, J.: *A Framework for the Language and Logic of Computer-Aided Phenomena-Based Process Modeling.* PhD Thesis, Massachusetts Institute of Technology, 2000.

Bischof C., L. Roh, A. Mauer-Oats: ADIC: An extensible automatic differentiation tool for ANSI-C. *Software-Pratice & Experience,* 27, 1427-1456, 1997.

Bogusch, R., B. Lohmann, W. Marquardt: Computer-aided process modeling with ModKit. *Computers and Chemical Engineering* 25, 963-995, 2001.

Bogusch, R., W. Marquardt: A Formal Representation of Process Model Equations. *Computers and Chemical Engineering* 21, 1105-1115, 1997.

Braunschweig, B., C.C. Pantelides, H.I. Britt, S. Sama: Process modeling: The promise of open software architectures. *Chemical Engineering Progress* 96, 65-76, 2000.

Britt, H.I., Pantelides, C.C.: Multipurpose process modeling environments. *Proc. Conf. on Foundations of Computer-Aided Process Design* '94, 128--141. CACHE Publications, 1994.

Brooke, A., D. Kendrick, A. Meeraus, R. Raman: *GAMS – A User's Guide.* GAMS Development Corp., 1998.

Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: *Pattern-oriented Software Architecture – A System of Patterns.* John Wiley, 1996.

Carpanzano, E., Maffezoni, C.: Symbolic manipulation techniques for model simplification in object-oriented modelling of large scale continuous systems. *Mathematics and Computers in Simulation* 48, 133-150, 1998.

Drengstig, T., S.O. Wasbo, B.A. Foss: A formal graphical-based process modeling methodology. *Computers and Chemical Engineering* 21 (suppl.), S835-S840, 1997.

Elmqvist, H., D. Brück, M. Otter: *Dymola – User's Manual.* Dynasim AB, Lund Sweden, 1996.

Foss, B., B. Lohmann, W. Marquardt: A Field Study of the Industrial Modeling Process. *Journal of Process Control* 8, 325-337, 1998.

Gamma, E.: *Design Patterns – Elements of Reusable Software.* Addison Wesley, 1994.

Gani, R., G. Hytoft, C. Jaksland, A. K. Jensen: An integrated computer-aided system for integrated design of chemical processes. *Computers and Chemical Engineering* 21, 1135-1146, 1997.

Gerstlauer, A., M. Hierlemann , W. Marquardt: *On the Representation of Balance Equations in a Knowledge-Based Process Modeling Tool.* Contr. to CHISA '93, Prag, September, 1993.

Geffers, W., L. von Wedel, J. Wyes, W. Marquardt: Integration von deklarativen und ausführbaren Modellen in offenen Simulationsumgebungen. In: K. Panreck, F. Dörrscheidt (Eds.): *Frontiers in Simulation: 15. Symposium Simulationstechnik*, Society for Modeling and Simulation International, 2001.

Griewank, A., D. Juedes, J. Utke: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software* 22, 131-167, 1996.

Hackenberg, J., C. Krobb, G. Schopfer, L. von Wedel, J. Wyes, W. Marquardt: *A Repository-based Environment for Lifecycle Modeling and Simulation.* Contr. To: JSPS International Workshop on Safety-Assured Operation and Concurrent Engineering, Yokohama, 3-5.12., 2000.

Hangos, K.M., I.T. Cameron: A formal representation of assumptions in process modeling. *Computers and Chemical Engineering* 25, 237-255, 2001a.

Hangos, K.M., I.T. Cameron: *Process Modelling and Process Analysis.* Academic Press, 2001b.

Harper, P. M., R. Gani, P. Kolar, T. Ishikawa: Computer-aided molecular design with combined molecular modelling and group contribution, *Fluid Phase Equilibria,* 158-160, 337-347, 1999.

Jarke, M., W. Marquardt: Design and Evaluation of computer-aided modeling tools. In: J.F. David, G. Stephanopoulos, V. Venkatasubrahmanian (Eds.), Intelligent Systems in Process Engineering. *AIChE Symposium Series 312* Vol. 92, 97-109, 1995.

Jensen, A.K.: *Generation of Problem Specific Simulation Models within an Integrated Computer Aided System.* PhD Thesis, Department of Chemical Engineering, DTU Lyngby, 1998.

Jensen, A. K., R. Gani: A Computer Aided Modelling System, *Computers and Chemical Engineering,* 23 (suppl.), 673-678, 1999.

Keeping, B., C.C. Pantelides: Novel methods for the efficient evaluation of stored mathematical expressions on vector computers. In: S. Pierucci (Ed.): *European Symposium on Computer Aided Process Engineering 10*, Elsevier, 2000.
Kruchten, P.: Architectural blueprints – The "4+1" view model of software architecture. *IEEE Software* 12, 42-50, 1995.
Lohmann, B., W. Marquardt: On the Systematization of the Process of Model Development, *Computers and Chemical Engineering* 21, Suppl., S213-S218, 1996.
Marquardt, W., A. Gerstlauer, E.D. Gilles: Modeling and Representation of Complex Objects: A Chemical Engineering Perspective. *Proc. 6th Int. Conf. on Industrial and Engineering Applications of Articifical Intelligence and Expert Systems*, Edinburgh, Scotland, 219-228, 1-4.6.1993.
Marquardt, W.: An object-oriented representation of structured process models. *Computers and Chemical Engineering* 16 (suppl.), S329-S336, 1992.
Marquardt, W.: Trends in computer-aided process modelling. *Computers and Chemical Engineering* 20, 591-609, 1996.
Marquardt, W., L. von Wedel, B. Bayer: Perspectives on Lifecycle Process Modeling. In: M.F. Malone, J.A. Trainham, B. Carnahan (Eds.): Foundations of Computer-Aided Process Design, *AIChE Symposium Series* 323, Vol. 96, 2000, 192-214.
Martinson, W.S., P.I. Barton: A differentiation index for partial differential-algebraic equations. *SIAM Journal on Scientific Computing*, 21, 2295-2315, 2000.
Meniai, A.-H., D. M. T. Newsham, B. Khalfaoui, *Chemical Engineering Research & Design*, 76(A11), 942-950, 1998.
Mitchell and Gauthier Ass.: *Advanced Continuous Simulation Language (ACSL). Beginner's Guide.* Mitchell and Gauthier Ass., 1992.
Modelica Association: *Modelica - A Unified Object-Oriented Language for Physical Systems models. Language Specification.* Available online at http://www.modelica.org, 2000.
Morton, W., C. Collingwood: An equation analyser for process models. *Computers and Chemical Engineering*, 22, 572-585, 1998.
Mowbray, T.J., R.C. Malveau: *CORBA Design Patterns.* John Wiley & Sons, 1997.
Nagl, M., W. Marquardt: *Tool integration via cooperation functionality.* Contr. To: ECCE 3, 3rd European Congress of Chemical Engineering, Nuremberg, 26-28.6.2001.
Nilsson, B.: Object-*Oriented Modeling of Chemical Processes.* Dissertation, Lund Institute of Technology, Schweden, 1993.
Oliver, D.W., T.P. Kelliher, J.G. Keegan: *Engineering complex systems with models and objects.* McGraw-Hill, 1997.
Pantelides, C.C.: The consistent initialization of differential-algebraic systems. *SIAM Journal of Scientific Computing* 9, 213-231, 1988.
Pantelides, C.C.: New challenges and opportunities for process modelling. In: R. Gani, S.B. Jorgensen (Eds.): *European Symposium on Computer-Aided Process Engineering – 11*, Elsevier, 2001.

Pfeiffer, B.M., W. Marquardt: Symbolic Semi-Discretization of Partial Differential Equation Systems. *Mathematics and Computers in Simulation* 42, 617-628, 1996.
Piela, P.C.: Ascend: *An object-oriented computer environment for modeling and analysis.* PhD thesis, Carnegie-Mellon-University, Pittsburg, USA, 1989.
Process Systems Enterprise: *gPROMS Introductory User Guide.* London, UK, 1997.
Räumschüssel, S.: *Rechnerunterstützte Vorverarbeitung und Codierung verfahrenstechnischer Modelle für die Simulationsumgebung DIVA.* Dissertation, University of Stuttgart, 1997.
Rittel, H., W. Kunz: *Issues as elements of information systems.* Working Paper No. 131, Univ. of California, Berkeley (CA), 1970.
Russel, B. M., R. Gani: MoT – A modelling test bed, In: *ICAS Manual* (CAPEC report), Technical University of Denmark, 2000.
Russel, S., P. Norvig: *Artificial Intelligence – A Modern Approach.* Prentice-Hall, 1994.
Subrahmanian, E., S.L. Konda, S.N. Levy, Y. Reich, A.W. Westerberg, and I. Monarch: Equations aren't enough: Informal modeling in design. *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing* 7, 257-274.
Stephanopoulos, G.; G. Henning, H. Leone: MODEL.LA – A modeling framework for process engineering – I. The formal framework. *Computers and chemical engineering* 14, 813-846, 1990a.
Stephanopoulos, G.; G. Henning, H. Leone: MODEL.LA – A modeling framework for process engineering – II. Multifacetted modeling of processing systems. *Computers and chemical engineering* 14, 813-846, 1990b.
Unger, J., A. Kröner, W. Marquardt: Structural Analysis of Differential-Algebraic Equation Systems - Theory and Applications. *Computers and Chemical Engineering* 19, 867-882, 1995.
Von Wedel, L., W. Marquardt: ROME: A Repository to Support the Integration of Models over the Lifecycle of Model-based Engineering Processes. In: S. Pierucci (Ed.): *European Symposium on Computer Aided Process Engineering-10,* Elsevier, 535-540, 2000.
Wasserman, A. I.: Tool Integration in Software Engineering Environments, In: F. Lang (Ed.), *Software Engineering Environments,* Springer, 138-150, 1990.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 3.2 Numerical Solvers

J. E. Tolsma & P. I. Barton

3.2.1 INTRODUCTION

Typical process modeling activities involve the application of a number of different numerical algorithms. For example, steady-state simulation of a process flowsheet involves the solution of a system or systems of nonlinear equations. Dynamic simulation of a process transient involves numerical integration of a dynamic system, represented by a system of ordinary differential equations (ODEs) or differential/algebraic equations (DAEs). If the model is a DAE or an ODE started from steady-state then the modeler must provide consistent initial conditions also obtained by solving a system of nonlinear equations. If the modeler is interested in the influence of time invariant parameters in the model on the state variable trajectories a parametric sensitivity analysis may be performed. Optimization typically follows simulation and requires the application of a suitable optimization algorithm. The modeler may be interested in steady-state or dynamic optimization.

Most numerical activities involve the application of a number of different solvers. In addition, the numerical algorithms themselves may be further decomposed into a set of elementary algorithms. For example, iterative methods for solving systems of nonlinear equations often require the solution of a linear equation subproblem, predictor-corrector methods for solving nonlinear stiff ODEs and DAEs involve the solution of systems of nonlinear equations during the corrector phase, and control parameterization strategies for dynamic optimization involve numerical integration and parametric sensitivity analysis subproblems.

This chapter briefly describes several common numerical activities andsome of the algorithms applied. In particular, the information that must be supplied in addition to the model equations is emphasized.

3.2.2 SOLUTION OF NONLINEAR SYSTEMS OF EQUATIONS

A common problem in CAPE, as well as essentially every discipline in science and engineering, is finding a solution to a system of nonlinear equations. Examples include steady-state simulation of a process flowsheet and computation of consistent initial conditions for dynamic simulation.

A large number of algorithms and numerical codes exist for this task. An appropriate algorithm depends both on the nature of the problem being solved and whether additional information concerning the system of equations is available. For example, whether or not partial derivatives of the system of equations are readily available may have a large impact on the selection of the algorithm used to solve that problem.

We consider the following system of n nonlinear equations in terms of n variables:

$$f(x) = 0 \tag{1}$$

for which we desire a solution x^*. A common method for solving this system is to replace equation (1) at some approximation to the solution, x^k, with the affine equation:

$$L^k(x) = A^k(x - x^k) + f(x^k), \tag{2}$$

where A^k is some suitable matrix, which may be the same for all k [43]. We can easily compute a solution of (2), provided A^k is nonsingular, for a new approximation to x^*. Repeating this procedure yields the iteration formula:

$$x^{k+1} = x^k - (A^k)^{-1} f(x^k) \qquad k = 0, 1, 2, \ldots \tag{3}$$

which will converge to a solution of (1), with appropriate selections for A^k, provided x^0 is "sufficiently" close to x^*. How close depends of the nature of f and choice of A^k

Choosing A^k to be $\nabla f(x^k)$, the Jacobian matrix of f evaluated at the current approximation x^k, yields the n-dimensional Newton's method with iteration formula:

$$x^{k+1} = x^k - \nabla f(x^k)^{-1} f(x^k) \qquad k = 0, 1, 2, \ldots \tag{4}$$

The advantage of Newton's method is that it has *quadratic convergence* properties when x^k is sufficiently close to the solution. Quadratic convergence of the sequence of iterates implies that the following inequality holds at each iteration:

$$||x^{k+1} - x^*|| < K||x^k - x^*||^2, \tag{5}$$

where $K < \infty$. The error at the $(k+1)$-th iteration is proportional to the square of the error at the k-th iteration, indicating convergence is rapid when near the solution. However, Newton's method is not likely to even converge to a solution if a good initial approximation is not available. This issue may be addressed through a line search approach. Given the search direction from Newton's method at step k, $p^k = -\nabla f(x^k)^{-1} f(x^k)$, a line search strategy attempts to find some scalar $\alpha^k > 0$ such that

$$||f(x^k + \alpha^k p^k)|| < ||f(x^k)||, \tag{6}$$

where $||\cdot||$ denotes some norm. A number of different line search strategies exist ranging from the minimization of the norm of f in the Newton direction to simply selecting an α^k such that inequality (6) holds (such a scalar can always be found since p^k is a descent direction in the norm of f).

The advantage of Newton's method is the excellent convergence properties when the initial approximation is close to the solution. The disadvantage is that the method requires the Jacobian matrix, which may be difficult to obtain analytically. Alternatively, the Jacobian matrix may be approximated with finite differences. For example, using the forward finite difference formula, the Jacobian entry $\partial f_i / \partial x_j$ may be approximated by:

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(x_j + \delta_j) - f_i(x_j)}{\delta_j}, \tag{7}$$

where the perturbation, δ_j, is selected based on the behavior of f evaluated at the current point. Selecting appropriate values for δ_j is often difficult; too large of a perturbation results in significant truncation error and a value too small results in excessive round-off error. Without any additional information, the approximation of the Jacobian matrix with forward finite differences requires $n+1$ function evaluations. This may be too costly in many applications. The cost of the finite difference approximation may be reduced if the sparsity pattern of the Jacobian matrix is available explicitly. The sparsity pattern, which indicates the dependence of the equations on the variables, may be represented by an incidence matrix containing entries equal to zero or unity. If the i-th equation depends on variable j then the i, j-th entry of the incidence matrix is unity, otherwise it is zero. Given the incidence matrix of a system of equations, structurally orthogonal columns for the Jacobian matrix may be identified [20]. Two columns are structurally orthogonal if they do not contain nonzero entries in the same row. The cost of evaluating the Jacobian by finite differences may be reduced by simultaneously computing multiple structurally orthogonal columns of the Jacobian matrix [21]. If the matrix is large and sparse then this saving is often significant. In addition to reducing the cost of finite difference approximation of the Jacobian matrix, an explicit representation of the sparsity pattern allows sparse linear algebra techniques to be employed in computing the direction of the step [24]. Exploiting sparsity in the linear algebra (e.g., only performing mathematical operations on the nonzero entries of the Jacobian matrix) not only reduces memory requirements and computational cost, but also allows the engineer to solve readily problems that otherwise would be exceedingly difficult or impossible to solve.

Obtaining the Jacobian matrix is not an issue if a symbolic representation of the system of equations is available, as is the case within modern equation-oriented modeling environments. In this case, the partial derivatives are readily computed via symbolic differentiation or through automatic differentiation (AD) techniques applied to the equation graphs. The latter approach has been shown to offer significant advantages over symbolic differentiation both in terms of speed and memory requirements [55]. If the system of equations are formulated as a computer program, then AD may be applied to the source code to obtain the desired derivatives [10, 56, 31]. In contrast to symbolic differentiation, which is applied to a symbolic representation of the system of equations, automatic (or algorithmic) differentiation has been designed to differentiate algorithms represented by computer code, such as C or Fortran. The actual statements in the code representing the system of equations are differentiated to produce the desired derivative information. A number of variants of AD have been developed to *accumulate* the partial derivatives within the code. However, a detailed description is beyond the scope of this chapter. The code may be differentiated by either using the *operator overloading* features of some modern programming languages (e.g., C++ and Fortran-90) or through the construction of new code. In the latter approach, new source code is constructed automatically from the original source to compute the desired derivative information. This is called an *automatic code transformation technique*. It should be noted that

AD is designed to differentiate *algorithms*. For example, it can differentiate codes containing IF statements, DO loops, iterative procedures, and complex hierarchies of subroutine and functions calls. In addition, the code may depend in complex ways on user supplied parameters, but the automatically generated derivative code is still able to adapt to changes in the parameter values and compute correct derivative information.

Sometimes a model is composed of a hybrid of equations formulated symbolically, and equations that are evaluated by procedure calls to external code. For example, external physical property packages are often interfaced to equation-oriented simulators via such procedure calls. In this case, Jacobians may be assembled by differentiating the symbolic equations, and applying finite differences to the equations evaluated by procedure calls. On the other hand, AD techniques may be applied to the external procedures to generate derivative code. Then appropriate calls to these new procedures can furnish analytical partial derivatives. In conjunction with differentiation of the symbolic equations, this yields an analytical Jacobian for the entire model. An implementation of this is described in [58].

Even if analytical derivatives are readily available, the cost associated with evaluating and LU factoring the Jacobian in Newton's method may be reduced by not evaluating and factoring the Jacobian at every iteration. The cost saving associated with this *deferred-update* Newton's method is offset by weaker convergence properties. However, this approach is often attractive if a good initial approximation to the solution is available. Alternatively, the matrix A^k in recursion (3) may be some derivative-free approximation, for example the secant-update formulas of Broyden [14] and Schubert [52]. Although convergence is weaker than Newton's method, not having to provide numerical values for the Jacobian matrix is advantageous in some cases. This is particularly true when solving flowsheeting problems with a sequential modular simulator where finite difference derivative evaluation may be prohibitively expensive. The advantages of applying AD techniques to codes representing unit operation models in order to obtain the derivative values efficiently is described in [2].

The line search strategies mentioned above are attempts to increase the convergence region of Newton's method. Alternatively, this problem may be addressed with a *trust region* strategy. The Levenberg-Marquardt method [36, 39] computes search directions defined by the following optimization problem:

$$\begin{aligned}\min_{p} \quad & \|f(x^k) + \nabla f(x^k)p\|_2 & (8)\\ \text{s.t.} \quad & \|p\|_2 \le \Delta^k, & (9)\end{aligned}$$

where Δ^k is called the *trust region radius* which limits the size of the step. The solution to this optimization problem is obtained by solving

$$\left(\nabla f(x^k)^T \nabla f(x^k) + \lambda^k I\right) p^k = -\nabla f(x^k)^T f(x^k), \tag{10}$$

where λ^k is determined by some strategy. This method has proven to be quite robust. However, since the step is obtained by solving a linear least-squares problem and λ^k is determined by some iterative procedure, the method can be costly. In addition, any sparsity that may be present in $\nabla f(x^k)$ is lost through fill-in when the search direction is computed. Consequently, the approach is not suitable for the large-scale systems common in process modeling.

Powell's "dog-leg" strategy avoids some of the costs of the Levenberg-Marquardt method by restricting $x^k + p^k$ to lie on the "dog-leg" consisting of the straight line from x^k to the minimum point of $\|f(x^k) + \nabla f(x^k)p^k\|_2$ in the direction of the steepest decent of $\|f(x^k) + \nabla f(x^k)p^k\|_2$ and the straight line from this point to the point obtained from a Newton step [48]. The trust region in this approach is readily identified, making the approach more efficient than the Levenberg-Marquardt method though not as robust.

Similar to the trust region approaches described above, the successive linear programming approach [23] solves the following minimization problem:

$$\begin{aligned}\min_{p} \quad & \|f(x^k) + \nabla f(x^k)p\|_1 & (11)\\ \text{s.t.} \quad & \|p\|_\infty \le \Delta^k. & (12)\end{aligned}$$

This choice in norms for the objective function and constraints results in the linear program [8]:

$$\begin{aligned}\min_{p,q,r} \quad & \sum_i (q_i + r_i) & (13)\\ \text{s.t.} \quad & \nabla f(x^k)p + q - r = -f(x^k) & (14)\\ & q, r \ge 0 & (15)\\ & p \ge -\Delta^k & (16)\\ & p \le \Delta^k. & (17)\end{aligned}$$

The advantage of this approach is that sparsity can be fully exploited and physical bounds on the variable values may be easily incorporated into this framework.

Compared to Newton's method, these trust region strategies have a larger region of convergence, but are computationally more expensive. They also suffer from the fact that they may converge to stationary points in the norm chosen for the system of equations, which are not necessarily the solutions we desire.

Continuation methods are also often used in an attempt to widen the region of convergence of an iterative method. In this approach, the system of equations to be solved is "embedded" within another mapping and an additional variable is introduced [43]. This new mapping has the general form:

$$h(x, \lambda) = 0, \tag{18}$$

where $h : \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$, $x \in \mathbb{R}^n$, and $\lambda \in \mathbb{R}$. The mapping has the following properties:

$$\begin{aligned} h(x,0) &= g(x) \\ h(x,1) &= f(x). \end{aligned}$$

The system of equations $g(x)$ is chosen such that a solution x^0 to $g(x^0) = 0$ is easily determined. The basic idea of continuation methods is to start at $h(x^0, 0) = g(x^0) = 0$ and follow the path defined by $h(x, \lambda) = 0$ to $h(x^*, 1) = f(x^*) = 0$, a solution we desire, by varying λ. Certain restrictions must be placed on $h(x, \lambda)$ for this path to exist, and even if it does exist, additional restrictions, exceedingly difficult to verify in general, must be placed on $h(x, \lambda)$ to ensure the path followed passes through $\mathbb{R}^n \times 1$ [1]. Nevertheless, continuation methods have been successfully applied in many cases. In particular, the approach is often successful when the solution is near singularities, that is, points where the Jacobian matrix is not invertible.

In all of the approaches described above there are no theoretical guarantees a solution will be found. Techniques based on interval Newton/generalized bisection, on the other hand, guarantee solutions will be found in some region defined by bounds on the variables, if a solution exists [40]. There are a number of variants of interval Newton methods, though they all require the evaluation of the *interval extension* of the system of equations and Jacobian matrix. An interval in this context is simply a closed, connected region on the real line defined by upper and lower bounds on the variable. For example, let $X \in \mathbb{IR}$ denote an interval, then

$$X = [X^L, X^U] = \{x : x \in \mathbb{R}, X^L \leq x \leq X^U\}. \tag{19}$$

An interval vector from the set $\mathbb{IR}^n$ is a vector where each of the n elements are intervals in $\mathbb{IR}$. Given a function $f(x)$, the interval extension of the function, denoted by $F(X)$, has the following property:

$$f(X) = \{f(x) : x \in X\} \subset F(X). \tag{20}$$

That is, the interval extension of a function evaluated on X contains the image set of the function on X. The construction of the interval extension from a function is not unique and may be performed in several ways, each producing a tighter or looser approximation of the image set. A common approach is the *natural interval extension* where each of the elementary mathematical operations involved in the computation of f are replaced by the corresponding elementary interval extension of the operation. For example, let $\circ$ define some elementary binary operation (e.g., $+,-,/$), the interval extension of $\circ$ is defined by

$$X \circ Y = \{x \circ y : x \in X, y \in Y\}. \tag{21}$$

The interval extensions for unary operations and "elementary" functions such as sin and log are defined in a similar manner. The code for evaluating the function may be readily converted to code evaluating the natural interval extension, albeit with significant effort in many cases. However, this conversion may be automated using the operator overloading features in some languages (e.g., Fortran 90 and C++) or through automatic code transformation techniques [56].

There are a number of variants of the interval Newton/generalized bisection approach and a detailed description is beyond the scope of this chapter. The disadvantages of these approaches is that they are typically quite computationally expensive and are thus not suitable for larger systems of equations. Furthermore, they require the user to provide interval extensions of the system of equations and Jacobian matrix. This, of course, is mitigated through the use of the automated techniques mentioned above, provided the user has access to the code evaluating the function of interest.

Several algorithms for finding solutions of nonlinear systems of equations are described in this section. Each of these methods are appropriate for certain types of problems and thus the user may have to select different solvers based on the particular problem at hand. The following section describes structural techniques that may be employed to further assist the user in solving a system of equations.

3.2.3 STRUCTURAL ALGORITHMS

As mentioned above, the availability of an explicit representation of the sparsity pattern of the system of equations may be used to reduce the cost of finite difference approximation of the Jacobian matrix. In addition, the sparsity pattern allows sparse linear algebra techniques to be employed, dramatically increasing the size of problems that may be solved. Automatic code transformation techniques, similar to those employed for derivative evaluation, can be employed to generate automatically code to evaluate the sparsity pattern for an arbitrary user-supplied code [56]. This section describes other *structural* techniques that can be applied to this information in order to assist the modeler in constructing a valid model and in subsequently solving the model.

Given the sparsity pattern (or incidence matrix) for a system of nonlinear equations, it is possible to find row and column permutations such that the permuted incidence matrix is in block lower triangular form (provided the matrix is not irreducible). Using the concept of *matchings* in a bipartite graph, Dulmage and Mendelsohn developed a canonical form of the block lower triangular decomposition [25]. The permuted incidence matrix is decomposed into a set of diagonal blocks. Nonzero entries may appear below the diagonal blocks, but only zeroes lie above the diagonal blocks. Three types of blocks appear on the diagonal in this decomposition: over-determined blocks which have more rows than columns, structurally fully-determined blocks which have the same number of rows and columns, and under-determined blocks which have more columns than rows. The diagonal block information may be used by the user to identify subsets of equations that must be deleted from the model to make it well-posed (from the over-determined blocks) and subsets of variables that require additional specifications or equations (from the under-determined blocks). With this information, the modeler is given a tremendous amount of information about how to modify an ill-posed problem. Algorithms to determine the Dulmage-Mendelsohn decomposition are described in [47].

Once a well-posed problem has been formulated, all diagonal blocks will be structurally fully-determined. When permuted in this form, the overall system of equations may be solved by solving sequentially the subsets of equations associated with the diagonal blocks. Not only is this more efficient (for example, the sum of the cost of the linear algebra required for each of the smaller blocks is much smaller than the cost associated with solving the full system of equations), but experience has shown this to be a more reliable approach than solving the entire system simultaneously [45]. Furthermore, each of the algorithms described in the previous section may not be suitable for every system of equations. By solving the overall system as a sequence of smaller subsets of equations and variables, an appropriate solver may be selected for each block based on its size and difficulty to converge.

Structural techniques may also be used to identify high index DAEs. Before numerically integrating a DAE, the user must provide a consistent set of initial conditions (this is explained in more detail in the following section). A high index DAE contains implicit constraints on the state variables that a consistent set of initial conditions must satisfy. These additional constraints may be uncovered by differentiating subsets of the equations in the DAE. The algorithm described in [44] uses structural criteria to identify subsets of the equations that must be differentiated in order to obtain constraints that must be satisfied by a consistent set of initial conditions. However, the approach described in [44] may indicate too few or too many equations must be differentiated, resulting in an incorrect determination of the index of the DAE [49]. Nevertheless, this algorithm has proven to be quite successful, particularly when applied to the DAEs that arise in process modeling. In our experience, this information is very useful in suggesting how a modeler might reformulate a model to make it low index.

The discussion above illustrates many uses for the sparsity pattern of the system of equations. If the equations are available in symbolic form, then the sparsity pattern may be readily generated and exploited. Otherwise, the sparsity pattern may be constructed by hand by the user or through automatic code transformation techniques [56].

3.2.4 NUMERICAL INTEGRATION OF DIFFERENTIAL AND ALGEBRAIC EQUATIONS

Studying the transient behavior of a batch process or the startup, shutdown, or effect of a disturbance in a continuous process requires the numerical integration of a nonlinear dynamic process model. In this situation, the process model consists of a system of ODEs or DAEs. Since many dynamic process models encountered in CAPE are most naturally described by DAEs, the solution of these dynamic systems will be described here. The general form of a DAE is:

$$f(\dot{z}, z, t) = 0, \tag{22}$$

where $f : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \longrightarrow \mathbb{R}^n$, t is the independent variable (usually time) and z and $\dot{z}$ are the state variables and their derivatives with respect to t. The states z and derivatives $\dot{z}$ are functions of t in an n-dimensional function space.

Before numerically integrating the DAE, the user must provide a set of consistent initial conditions, which satisfy:

$$f(\dot{z}(t_0), z(t_0), t_0) = 0. \tag{23}$$

This is a system of n algebraic equations in terms of $2n$ variables (t_0 is known). In order to compute the consistent initial conditions, equation (23) is augmented with additional algebraic equations defining the initial conditions and the resulting system is solved, possibly using one of the algorithms described previously. The number of additional equations that must be introduced is equal to the number of dynamic degrees of freedom, n_d. For most index 1 DAEs [12], n_d is equal to the number of time derivatives of the state variables that appear explicitly in the DAE. Hence, it is only necessary to solve for $n + n_d$ unknowns, using the n DAEs and the n_d additional equations defining the initial conditions. If the index is greater than unity, then as described above, additional implicit constraints appear in the DAE that may be uncovered through differentiation [44]. These additional constraints reduce the dynamic degrees of freedom.

It may be necessary to compute consistent initial conditions even in the case of an ODE. For example, often it is necessary to study the response of a process at steady-state to a disturbance. The consistent initial condition of steady-state is computed by solving the following system of nonlinear equations:

$$\begin{aligned} \dot{z}(t_0) &= g(z(t_0), t_0) && (24) \\ \dot{z}(t_0) &= 0. && (25) \end{aligned}$$

This is identical to a steady-state simulation.

Several methods have been developed for numerically integrating DAEs and there are a number of powerful codes developed based on these methods [9, 32, 13, 4]. Linear multi-step methods, specifically, the methods based on backward differentiation formulas (BDF), are the most popular and will be described here. Many of these codes are restricted to index-1 DAEs or higher index systems with special structure [12].

The q-step BDF consists of replacing the time derivatives, $\dot{z}$, by polynomials which interpolate the previous $q + 1$ computed solutions:

$$f(\frac{1}{h}\sum_{i=0}^{q} \alpha_i z_{k-i}, z_k, t_k) = 0, \tag{26}$$

where subscript k denotes the current time step and q is the integration order. The method reduces to the implicit Euler method when $q = 1$. The BDF method advances the time trajectories using a predictor-corrector approach: a *predictor* polynomial is used to obtain an estimate for the next point on the trajectory, followed by the iterative solution of equation (26), where the time derivatives have been replaced by *corrector* polynomials. Typically, a deferred-update Newton's method is employed during the corrector phase, using the LU factored iteration matrix from some previous time step. Modern codes attempt to balance the cost associated with the iteration matrix evaluation/factorization with the reduced order and time stepsize caused by using an outdated iteration matrix.

Several implementations of the BDF method are available, each with different heuristics for order and stepsize selection. The codes available attempt to make the stepsize large enough for efficient integration while still maintaining accuracy and stability [12].

In addition to the linear multi-step method described above, one-step methods are often used for the numerical integration of dynamic systems. In particular, the *implicit Runge-Kutta* (IRK) methods have been successfully applied to DAEs. An M-stage IRK method applied to DAE (22) has the following form:

$$f(\dot{Z}_i, z_{k-1} + h\sum_{j=1}^{M} a_{i,j}\dot{Z}_j, t_{k-1} + c_i h) = 0 \qquad i = 1, 2, \ldots, M \tag{27}$$

$$z_k = z_{k-1} + h\sum_{i=1}^{M} b_i\dot{Z}_i \tag{28}$$

where $h = t_k - t_{k-1}$, $\dot{Z}_i$ are the *stage derivatives*, and $a_{i,j}$, b_i, c_i, $i, j = 1, \ldots, M$ are the constants associated with the method [12]. The stage derivative $\dot{Z}_i$ is an estimate for $\dot{z}(t_{k-1} + c_i h)$. Although BDF methods tend to perform better on average than IRK methods, these one-step methods can potentially have an advantage if there are frequent discontinuities along the integration domain. Unlike the BDF method which uses a small stepsize and low integration order after each discontinuity, the IRK method can be started at a higher integration order, resulting in faster restarts.

3.2.4.1 Hybrid Discrete/Continuous Systems

The section above describes the numerical integration of DAEs. This discussion assumed that DAE was a smooth function throughout the entire integration domain. A common situation for interesting and realistic problems, however, is where the functional form of the DAE changes during the course of a dynamic simulation. These type of models are referred to as hybrid discrete/continuous systems.

Hybrid discrete/continuous systems (or simply hybrid systems) are dynamic systems that exhibit both discrete and continuous behavior. The continuous behavior of the model is usually described by one or more ODE or DAE systems. The discrete behavior, which occurs at particular points in time known as *events*, includes phenomenon such as nonsmooth forcing, switching of the vector field, and jumps in the state. We can broadly distinguish between two types of events: time events and state events. A time event is an event where the time of occurrence is known a priori. In contrast, a state event occurs when some condition involving the state variables is satisfied (e.g., the level in a tank reaches a certain height) and the time of occurrence is in general not known a priori.

Hybrid system models appear in a wide variety of disciplines. They appear directly when modeling discrete control actions imposed on a system (e.g., a safety interlock system), disturbances introduced into a continuous system and sequence controllers. In addition, state events are typically introduced by modeling abstractions (e.g., physical properties of fluids are often represented by piecewise smooth semi-empirical relationships), physical phenomenon such as irregularities in vessel geometry that can cause discontinuities in the relationship between holdup and level, and changes in the number of thermodynamic phases present. Within a code these may appear as nonsmooth intrinsic functions, such as MIN and MAX, as well as the more obvious IF statements.

It is well known that numerically integrating systems containing hidden discontinuities is inefficient and sometimes results in integration failures [33]. In the worst case, ignoring discontinuities can lead to incorrect results that may escape the modeler's attention [19].

A number of approaches have been developed to handle properly the discontinuities that occur during the numerical integration of a hybrid system. These approaches range from replacing the discrete aspects with a smooth approximation, to modifying the integrator itself to attempt to identify and handle the discontinuities. Replacing discontinuities with smooth approximations is laborious and error prone, particularly when modifying legacy code, and the modified model may not capture properly the dynamics of interest. A better alternative is to handle the discontinuities explicitly during numerical integration [18, 46]. To perform the state event handling correctly, a significant amount of additional information must be made available by the user. This additional information is described below.

The following hybrid system formalism, adopted from [5, 28], will be used in the remainder of this section to highlight additional information that must be provided by the user for proper state event handling. State space is divided into a set of *modes*, $S = \cup_{k=1}^{n_m} S_k$, where each mode S_k is characterized by:

1. A set of variables $\{\dot{x}^{(k)}(p,t), x^{(k)}(p,t), y^{(k)}(p,t), p, t\}$ where $x^{(k)}(p,t)$, $\dot{x}^{(k)}(p,t)$ are the differential variables and their derivatives, which are functions of p and t in an $n_x^{(k)}$-dimensional function space, $y^{(k)}(p,t)$ are the algebraic variables, which are functions of p and t in an $n_y^{(k)}$-dimensional function space, $p \in \mathbb{R}^{n_p}$ are the time invariant parameters, and $t \in \mathbb{R}$ is time, the independent variable,

2. A set of equations, $f^{(k)}(\dot{x}^{(k)}, x^{(k)}, y^{(k)}, p, t) = 0$, where $f^{(k)} : \mathbb{R}^{n_x^{(k)}} \times$

$$\mathbb{R}^{n_x^{(k)}} \times \mathbb{R}^{n_y^{(k)}} \times \mathbb{R}^{n_p} \times \mathbb{R} \longrightarrow \mathbb{R}^{n_x^{(k)}} \times \mathbb{R}^{n_y^{(k)}},$$

3. A set of transitions from one mode to another (possibly the same mode), $J^{(k)}$, where each transition is characterized by:
   (a) Transition conditions $L_j^{(k)}(\dot{x}^{(k)}, x^{(k)}, y^{(k)}, p, t)$, $j \in J^{(k)}$, defining the transition times and
   (b) Transition functions $T_j^{(k)}(\dot{x}^{(k+1)}, x^{(k+1)}, y^{(k+1)}, \dot{x}^{(k)}, x^{(k)}, y^{(k)}, p, t) = 0$, $j \in J^{(k)}$, a system of equations that maps the final values of the variables in the current mode to the initial values in the next mode. (Initial conditions are a special case of these transition functions.) The number of transition functions is determined by the dynamic degrees of freedom of the DAE in the new mode.

Notice that the state variables are partitioned based on whether or not the corresponding time derivative appears explicitly in the DAE. It is assumed that the transition conditions, $L_j^{(k)}(\dot{x}^{(k)}, x^{(k)}, y^{(k)}, p, t)$, are logical expressions composed of one or more real-valued relational expressions involving relational operators $<$, $\leq$, $>$, or $\geq$. For example, the logical expression $(x_1 \geq 5) \vee (x_2 > 2 \wedge x_2 \leq 5)$ contains the relational expressions $x_1 \geq 5$, $x_2 > 2$, and $x_2 \leq 5$. Putting these relational expressions into the form

$$g \geq (>)\, 0 \tag{29}$$

defines *discontinuity functions.* In a code these state conditions are typically represented as IF statements. The discontinuity functions associated with the state condition represented by the following Fortran IF statement:

```
IF ( LEVEL > 10.0 .AND. DELTAP > 0.0 ) THEN
    .
    .
    .
END IF
```

are the expressions:

```
LEVEL - 10.0
DELTAP.
```

The logical proposition of the transition condition may switch value when one or more discontinuity functions cross zero. A switching event is defined by the earliest time at which one of the transition conditions becomes true. Many modern state event location algorithms identify the events by monitoring the discontinuity functions associated with state conditions for these zero crossings.

By monitoring discontinuity functions for zero crossings, state events may be identified. The various algorithms available for state event location differ in how this monitoring is performed, ranging from simply examining the logical value of transition conditions at mesh points during the numerical integration, to rigorously identifying the earliest (and thus correct) zero crossing of the discontinuity functions at each time step throughout the entire integration [46]. Once the state event is identified, the precise event time is computed. At this point the integration is stopped, the DAE is reinitialized in the new discrete mode using the transition functions and the integration is continued. Obviously, in order to perform properly the hybrid numerical integration, the user must provide a substantial amount of additional information, including the state conditions, the discontinuity functions, and the transition functions. If the hybrid DAE is available in symbolic form then this information may be readily identified, extracted, and utilized within a proper state event location algorithm. Recently, this capability has been extended to hybrid DAEs represented as complex codes written in computer languages such as Fortran [57]. This is important even within an equation-oriented environment where some of the model equations are computed by calls to external code (e.g., physical property libraries).

3.2.5 PARAMETRIC SENSITIVITY ANALYSIS

The previous section describes some algorithms and solvers used to integrate numerically systems of ODEs and DAEs. Often the next step after numerical integration is parametric sensitivity analysis. This calculation is performed to examine the effects of infinitesimal perturbations in the parameters of interest on the state variable trajectories. These parameters may be constants within the model or initial conditions. Parametric sensitivities are often useful themselves, for example for experimental design and process sensitivity studies. Sensitivities are also important within the context of other calculations, including parameter estimation and control parameterization approaches for dynamic optimization.

Given a DAE containing n_p parameters, p,

$$f(\dot{z}, z, t; p) = 0, \tag{30}$$

the forward sensitivities may be obtained through numerical integration of the following augmented system of $n(n_p + 1)$ DAEs:

$$\begin{aligned} f(\dot{z}, z, t; p) &= 0 \\ \frac{\partial f}{\partial \dot{z}}\frac{\partial \dot{z}}{\partial p_1} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial p_1} + \frac{\partial f}{\partial p_1} &= 0 \\ &\vdots \\ \frac{\partial f}{\partial \dot{z}}\frac{\partial \dot{z}}{\partial p_{n_p}} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial p_{n_p}} + \frac{\partial f}{\partial p_{n_p}} &= 0, \end{aligned}$$

where n is the dimension of the DAE. Efficient algorithms are available for computing the forward sensitivities [38, 27]. These algorithms exploit structure in the augmented DAE. How the sensitivity residuals (the additional equations appended to the original DAE) are evaluated can have a significant impact on the performance of the calculation. When the number of parameters is small compared to the number of state variables, then using the *seed matrix* option of AD [10] is quite efficient. In this mode, the product of the Jacobian and a vector (in this case, the sensitivity vector) is computed efficiently at a cost bounded above by three times the cost of a DAE residual evaluation. If the number of parameters is large and the Jacobian matrix is sparse, then it is often beneficial to evaluate the sparse Jacobian then perform sparse Jacobian-vector products to compute the sensitivity residuals. The sensitivity residuals may also be computed via finite differences, however, the lower accuracy can often severely impact the quality of the results.

The cost and memory requirements for computing the forward sensitivities scales with the number of parameters of interest. However, a common situation is where the number of parameters is extremely large and the user is only interested in the derivatives of a few outputs. This is often the case when the sensitivities are required within control parameterization strategies for dynamic optimization. In these types of situations, a better alternative to computing the desired derivatives is by integrating the adjoint (or reverse) sensitivity system. Consider the following nonlinear ODE and initial condition:

$$\dot{x} = g(x, t; p) \tag{31}$$

$$x(t_0, t_0; p) = x_0(p) \tag{32}$$

Suppose we are interested in the derivative of the scalar-valued function $\phi(x(t, t_0), p)$ with respect to p at time t_f, where the dimension of p is very large. This sensitivity may be computed by the formula:

$$\begin{aligned}
\frac{\partial \phi}{\partial p}(x(t_f, t_0; p), p) &= \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial p}(t_f, t_0; p) + \frac{\partial \phi}{\partial p} \\
&= \lambda(t_0, t_f; p)^T \frac{\partial x_0}{\partial p} + \frac{\partial \phi}{\partial p} \\
&\quad + \int_{t_0}^{t_f} \lambda(\tau, t_f; p)^T \frac{\partial g}{\partial p}(\tau; p) d\tau,
\end{aligned} \tag{33}$$

where the vector of adjoints, λ, is obtained by integrating numerically the following system backwards from t_f to t_0:

$$\dot{\lambda} = -\left(\frac{\partial f}{\partial x}\right)^T \lambda \tag{34}$$

$$\lambda(t_f, t_f; p) = \left(\frac{\partial g}{\partial x}\right)^T. \tag{35}$$

The advantage of the adjoint approach for derivatives is that the cost of the backwards integration of the adjoints does not scale with the number of parameters. However, this is offset by the fact that both a forward and backward integration are required to obtain the desired derivatives. Also, adjoints only give information about derivatives at one point in time for a single dependent function. The intermediate values of the λ's have no interpretation as derivative information, and a new adjoint system has to be solved for each dependent function. These adjoint sensitivity results have been recently extended to DAEs [17, 16].

3.2.5.1 Hybrid Discrete/Continuous Parametric Sensitivity Analysis

As mentioned above, discontinuities embedded within the model often cause adverse effects in the numerical integration, including inefficient integration and sometimes failures. The situation is far worse when performing parametric sensitivity analysis of a model that contains discontinuities. Because the sensitivities will often jump at the discontinuities, if the numerical integration is not stopped and the jump computed explicitly, the computed sensitivity trajectories will generally be incorrect [28]. Furthermore, if the calculation does not fail, the user is given no indication about the correctness of the computed sensitivities [57].

As with proper hybrid system numerical integration, performing properly the hybrid sensitivity analysis requires a substantial amount of additional information from the user. The remainder of this section uses the hybrid system formalism described in Section 4.1. The sensitivity of the transition time with respect to parameter p_i is given by

$$\begin{aligned}
&\frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial \dot{x}^{(k)}}\left(\dot{s}_{x,i}^{(k)} + \ddot{x}^{(k)}\frac{\partial t}{\partial p_i}\right) + \frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial x^{(k)}}\left(s_{x,i}^{(k)} + \dot{x}^{(k)}\frac{\partial t}{\partial p_i}\right) + \\
&\frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial y^{(k)}}\left(s_{y,i}^{(k)} + \dot{y}^{(k)}\frac{\partial t}{\partial p_i}\right) + \frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial p_i} + \frac{\partial \tilde{g}_{k+1}^{(k)}}{\partial t}\frac{\partial t}{\partial p_i} = 0, \qquad (36)
\end{aligned}$$

where $\tilde{g}_{k+1}^{(k)}$ is the discontinuity function that triggers the event. The equation above is solved for $\partial t/\partial p_i$. This quantity indicates how the event time is changes with small perturbations in the parameter p_i. The required elements of $\dot{x}$, $\dot{y}$, and $\ddot{x}$ in the equation above can be computed from the first and higher order derivatives of the DAE. The remainder of this section assumes the following condition holds:

$$\text{rank}\left(\frac{\partial f^{(k)}}{\partial \dot{x}^{(k)}}\frac{\partial f^{(k)}}{\partial y^{(k)}}\right) = n_x^{(k)} + n_y^{(k)} \qquad (37)$$

for all k. This condition is sufficient for the DAEs on each of the discrete modes to be index 1. The jump in sensitivities can then be computed by solving the following linear system:

$$\begin{aligned}
&\begin{bmatrix} \frac{\partial T_{k+1}^{(k)}}{\partial \dot{x}^{(k+1)}} & \frac{\partial T_{k+1}^{(k)}}{\partial x^{(k+1)}} & \frac{\partial T_{k+1}^{(k)}}{\partial y^{(k+1)}} \\ \frac{\partial f^{(k+1)}}{\partial \dot{x}^{(k+1)}} & \frac{\partial f^{(k+1)}}{\partial x^{(k+1)}} & \frac{\partial f^{(k+1)}}{\partial y^{(k+1)}} \end{bmatrix} \begin{bmatrix} \frac{\partial \dot{x}^{(k+1)}}{\partial p_i} \\ \frac{\partial x^{(k+1)}}{\partial p_i} \\ \frac{\partial y^{(k+1)}}{\partial p_i} \end{bmatrix} = \qquad (38) \\
-&\begin{bmatrix} \frac{\partial T_{k+1}^{(k)}}{\partial \dot{x}^{(k+1)}} & \frac{\partial T_{k+1}^{(k)}}{\partial x^{(k+1)}} & \frac{\partial T_{k+1}^{(k)}}{\partial y^{(k+1)}} \\ \frac{\partial f^{(k+1)}}{\partial \dot{x}^{(k+1)}} & \frac{\partial f^{(k+1)}}{\partial x^{(k+1)}} & \frac{\partial f^{(k+1)}}{\partial y^{(k+1)}} \end{bmatrix} \begin{bmatrix} \frac{\partial \dot{x}^{(k+1)}}{\partial t} \\ \frac{\partial x^{(k+1)}}{\partial t} \\ \frac{\partial y^{(k+1)}}{\partial t} \end{bmatrix} \frac{\partial t}{\partial p_i}
\end{aligned}$$

$$- \begin{bmatrix} \frac{\partial T_{k+1}^{(k)}}{\partial \dot{x}^{(k)}} & \frac{\partial T_{k+1}^{(k)}}{\partial x^{(k)}} & \frac{\partial T_{k+1}^{(k)}}{\partial y^{(k)}} & \frac{\partial T_{k+1}^{(k)}}{\partial p_i} & \frac{\partial T_{k+1}^{(k)}}{\partial t} \\ 0 & 0 & 0 & \frac{\partial f^{(k+1)}}{\partial p_i} & \frac{\partial f^{(k+1)}}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial \dot{x}^{(k)}}{\partial p_i} + \frac{\partial \dot{x}^{(k)}}{\partial t} \frac{\partial t}{\partial p_i} \\ \frac{\partial x^{(k)}}{\partial p_i} + \frac{\partial x^{(k)}}{\partial t} \frac{\partial t}{\partial p_i} \\ \frac{\partial y^{(k)}}{\partial p_i} + \frac{\partial y^{(k)}}{\partial t} \frac{\partial t}{\partial p_i} \\ I \\ \frac{\partial t}{\partial p_i} \end{bmatrix} .$$

Performing the hybrid sensitivity analysis properly requires not only the transition conditions, transition functions, and discontinuity functions, but also the derivatives of these expressions with respect to the state variables, time derivatives, parameters, and time. This burden is mitigated if the model is available symbolically. Recently, it has been shown that this additional information can be generated from the Fortran code of a hybrid system [57]. An adjoint formulation for hybrid dynamic systems is described in [51].

3.2.6 OPTIMIZATION ALGORITHMS

The numerical algorithms described previously are used to examine the behavior of a process under a fixed set of operating conditions or examine the effect of infinitesimal perturbations on the response. In this capacity, the model is used to understand and predict the quantitative behavior of the process. Often, however, an engineer desires to improve the operation or design of a process with respect to some criterion. Optimization algorithms may be employed to adjust systematically the degrees of freedom in the problem to meet these objectives. As with simulation, the modeler may be interested in steady-state and dynamic optimization. The former case involves finding values for the time invariant parameters of a steady-state model to improve the objective. The latter case, where the embedded model is a dynamic system, not only includes adjustment of time invariant parameters but may also include the determination of optimal profiles for the time varying controls. The decision variables in the optimization problem may take continuous or discrete values. Discrete valued variables may be used to represent quantities such as the optimal number of trays in a distillation column or whether or not a a piece of equipment is required. The optimization problems may be classified by the decision variable types and the embedded process model. Table 1 contains a summary of different optimization problems commonly encountered.

Table 1: Classification of common optimization algorithms.

Name	Characteristics
LP	**Linear Program** Linear algebraic objective function and constraints. All variables are continuous valued.
MILP	**Mixed Integer Linear Program** Linear algebraic objective function and constraints. Both continuous and discrete valued variables.
NLP	**Nonlinear Program** Nonlinear algebraic objective function and constraints. All variables are continuous valued.
MINLP	**Mixed Integer Nonlinear Program** Nonlinear algebraic objective function and constraints. Both continuous and discrete valued variables.
DO	**Dynamic Optimization** Nonlinear algebraic objective function with possible integral terms. Nonlinear differential and algebraic constraints. All variables are continuous valued.
MIDO	**Mixed Integer Dynamic Optimization** Nonlinear algebraic objective function with possible integral terms. Nonlinear differential and algebraic constraints. Both continuous and discrete valued variables.

Optimization has a number of applications within CAPE. For example, during process design, the problem may be formulated and solved as an MINLP. The objective function in this case may be to minimize operating costs and capital expenditure for new equipment while satisfying production rates and purity. During batch process design, dynamic optimization may be employed to determine optimal control profiles to maximize selectivity or minimize waste. In addition, steady-state and dynamic optimization algorithms may be used for parameter estimation. A complete discussion of the algorithms required to solve all of the types of optimization problems described in Table 1 is beyond the scope of this chapter. The remainder of this section describes several approaches for solving common optimization problems that arise in practice, NLP and dynamic optimization.

3.2.6.1 Steady-state Optimization

This section describes two algorithms for solving the NLPs that arise during steady-state optimization. Efficient algorithms for solving NLPs are also important due to the fact that NLPs often arise as subproblems of other optimization algorithms, e.g., dynamic optimization and MINLP problems. Consider the following NLP:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0 \\ & x^L \leq x \leq x^U, \end{aligned} \tag{39}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the objective function, $g : \mathbb{R}^n \longrightarrow \mathbb{R}^{n_i}$ are the inequality constraints, $h : \mathbb{R}^n \longrightarrow \mathbb{R}^{n_e}$ are the equality constraints, $x \in \mathbb{R}^n$ are the variables, and $x^L, x^U \in \mathbb{R}^n$ are the variable lower and upper bounds, respectively. In process optimization, f is typically an economic objective function, h includes the model equations (such as mass and energy balances), and g represents design constraints such as product purity or waste production. One popular algorithm for solving (39) is successive (or sequential) quadratic programming (SQP). In this approach, a sequence of search directions are generated by solving quadratic program (QP) subproblems. A QP is an optimization problem where the objective function is quadratic and the constraints are linear. At iteration k, the following QP is solved for a search direction d^k:

$$\begin{aligned} \min_{d^k} \quad & f(x^k) + \nabla f(x^k)d^k + \frac{1}{2}(d^k)^T B^k d^k \\ \text{s.t.} \quad & g(x^k) + \nabla g(x^k)d^k \leq 0 \\ & h(x^k) + \nabla h(x^k)d^k = 0 \\ & x^L \leq x^k + d^k \leq x^U. \end{aligned} \tag{40}$$

The next approximation to the solution is simply

$$x^{k+1} = x^k + d^k. \tag{41}$$

The quadratic objective function in (40) is more than simply a second-order Taylor series approximation of $f(x)$ at point k. The matrix B^k approximates the Hessian of the Lagrange function:

$$L(x, u, \nu) = f(x) + u^T h(x) + \nu^T g(x). \tag{42}$$

Using an approximation of $\nabla_{xx} L(x^k, u^k, \nu^k)$, rather than $\nabla^2 f(x^k)$, includes information about the curvature of the constraints in the objective function of (40). In order for the d^k generated in (40) to be descent directions and thus converge to a local minimizer of (39), the matrix B^k must be positive definite. This may be achieved by using a positive definite approximation for B^k, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [30]. Not only does the positive definite update ensure that descent directions will be generated, it also avoids the need to compute second-order derivative information. However, the disadvantage is that the convergence is superlinear rather than quadratic and the update is dense.

As with Newton's method for solving systems of nonlinear equations, convergence for SQP is only guaranteed if the initial guess is sufficiently close to the solution. The algorithm can be made globally convergent through the use of a merit function [6]. A common example is the l_1 merit function:

$$\phi(x) = f(x) + \mu \left[\sum_i \max\{0, g_i(x)\} + \sum_j |h_j(x)| \right], \tag{43}$$

where μ is selected such that the computed d^k is a descent direction of ϕ. At each major iteration of the SQP algorithm, the new approximation to the solution is now

$$x^{k+1} = x^k + \lambda^k d^k. \tag{44}$$

Choosing λ^k such that $\phi(x^k + \lambda^k d^k)$ is minimized, ensures global convergence of the SQP method.

The fact that the BFGS update used in (40) is dense limits the size of problems that can be solved with the SQP algorithm described above. However, many problems in practice may be large but have few degrees of freedom. That is, the difference between the number of variables and number of active constraints is relatively small. This observation may be exploited in range and null space decomposition strategies for SQP [30, 59, 60]. In this reduced-space SQP approach, the QP subproblem is solved in the null space of the active constraints (which has dimension equal to the number of degrees of freedom). Linear algebra is used to compute search directions in the range space. Since sparse linear algebra techniques may be employed, the size of problems that may be handled is substantially larger.

Another popular algorithm for solving NLPs is that implemented in the Minos package [41]. When using Minos, variables that appear linearly in the model are distinguished from variables that appear in nonlinear expressions. In this case, the NLP (39) is rearranged to:

$$\begin{array}{ll} \min\limits_{z} & f^0(x_n) + c^T x_l \\ \text{s.t.} & f(x_n) + A x_l = b_1 \\ & B x_n + C x_l = b_2 \\ & x_n^L \leq x_n \leq x_n^U \\ & x_l^L \leq x_l \leq x_l^U, \end{array} \tag{45}$$

where x_l denote the subset of variables that appear linearly in the objective function and constraints and x_n are all other variables. The objective function and constraints have also been partitioned into linear and nonlinear terms. Notice that the inequality constraints have been converted to equalities by introducing *slack variables.* At the k-th *major* iteration, the following problem is solved:

$$\begin{array}{ll} \min\limits_{z} & f^0(x_n) + c^T x_l - \lambda_k^T(f(x_n) - \hat{f}(x_n; x_n^k)) + \\ & \frac{1}{2}\rho(f(x_n) - \hat{f}(x_n; x_n^k))^T(f(x_n) - \hat{f}(x_n; x_n^k)) \\ \text{s.t.} & \begin{bmatrix} \nabla f(x_n^k) & A \\ B & C \end{bmatrix} \begin{bmatrix} x_n \\ x_l \end{bmatrix} = \begin{bmatrix} b_1 + \nabla f(x_n^k) - f(x_n^k) \\ b_2 \end{bmatrix} \\ & x_n^L \leq x_n \leq x_n^U \\ & x_l^L \leq x_l \leq x_l^U, \end{array} \tag{46}$$

where $\hat{f}(x_n; x_n^k) \equiv f(x_n^k) + \nabla f(x_n^k)(x_n - x_n^k)$ denotes the linearized constraints at the current point x_n^k. The scalar $\rho > 0$ is chosen to improve convergence when the initial point is far from the solution and the problem is highly nonlinear. The value of ρ is reduced to zero when near the solution. This linearly-constrained subproblem is typically solved through an unconstrained optimization procedure in the space of the linearized active constraints (the *minor* iterations).

A full optimization in the space of active constraints is performed at each major iteration in Minos. Consequently, Minos typically requires more function evaluations than SQP and is better suited to process models which are mostly linear. SQP tends to perform better than Minos when the constraints are highly nonlinear. In addition, since fewer function evaluations are typically required with SQP, this approach is often preferable when the function evaluations are expensive.

3.2.6.2 Dynamic Optimization

The section above described two common approaches for solving NLPs, optimization problems where the underlying model is algebraic. Dynamic optimization, where the underlying model is a system of differential and algebraic equations, involves computing control trajectories that optimize some criteria. Consider the following dynamic optimization problem:

$$\begin{aligned}
\min_{u,p} \quad & J = \psi(z(t_f), t_f; p) + \int_{t_0}^{t_f} L(z, u, t; p)dt \qquad (47)\\
\text{s.t.} \quad & f(\dot{z}, z, u, t; p) = 0\\
& h(z, u, t; p) = 0\\
& g(z, u, t; p) \leq 0\\
& \phi_0(\dot{z}(t_0), z(t_0), u(t_0), t_0; p) = 0\\
& \phi_f(\dot{z}(t_f), z(t_f), u(t_f), t_f; p) = 0
\end{aligned}$$

where J is a scalar performance measure, containing a scalar function and integral term to be minimized. The constraint f is the embedded dynamic system, h are equality path constraints, g are the inequality path constraints, and ϕ_0 and ϕ_f are constraints at the initial and final time, respectively. Note the presence of control profiles, $u(t)$, as decision variables. These are elements of function spaces.

One approach for solving dynamic optimization problems is to parameterize both the control and state variable trajectories, $u(t)$ and $z(t)$, using a finite set of basis functions, converting the dynamic problem into an NLP. This approach is called the *collocation* or *simultaneous* method. The resulting NLP may be solved using an appropriate algorithm, such as those described above. The decision variables are now the coefficients used in the discretization, the size of the subintervals in the discretization (often the time domain is split into a series of subintervals, and separate basis functions are used on each subinterval), and the original time invariant parameters in the model. The advantage of this approach is that path constraints (i.e., constraints that must be satisfied at points along the trajectory) are readily incorporated into the formulation. On the other hand, it is only guaranteed that the path constraints are satisfied at the mesh points of the discretization. The disadvantage is that very large optimization problems are produced from reasonably sized problems. Further, the number of degrees of freedom is quite large. This reduces the effectiveness of the range and null space strategies for SQP.

Alternatively, only the control profiles can be parameterized. The dynamic optimization problem is then converted into an NLP where the objective function and constraints are obtained by integrating the embedded dynamic system, and gradient and Jacobian values are obtained via the chain rule and a parametric sensitivity analysis or adjoint analysis. In this *control parameterization* approach, the decision variables are the coefficients of polynomials used to discretize the controls, the size of the subintervals in the discretization, and the time invariant parameters originally in the model. Although initial and end-point constraints are readily incorporated into this method, complications arise if there are path constraints. Equality path constraints may be handled in a number of ways. They may be included in the objective function as a heavily weighted integral term, relying on the optimizer to minimize constraint violations [15]. Similarly, they may be converted to end-point constraints by replacing them with integrals of the constraint violation. Also, equality constraints may be appended to the DAE model, f. This will require some of the controls to be treated as unknowns determined by the numerical integration. However, these additional constraints may result in a high index DAE. An approach for handling this case is described in [26]. Inequality path constraints may be handled in a manner similar to the integral violation methods for equality path constraints. Often, the inequality path constraints are also enforced as regular inequalities at the time subdomain boundaries.

3.2.7 USING SOLVERS

The previous sections describe several numerical algorithms used to solve a variety of problems commonly encountered during process modeling activities. This section briefly highlights some of the ways these solvers are actually used within an application. Specifically, the form the solvers are made available to the user is discussed. Three approaches are described, procedure libraries, object-oriented libraries, and component libraries. These three library representations closely match the evolution of programming practices within the software development community.

3.2.7.1 Procedure Libraries

Traditional scientific computing has been implemented by calling numerical routines in a well-defined order. These numerical routines are typically made available in the form of procedure libraries, containing synchronous, local-address-space functions written in a procedural language such as C or Fortran. A very large number of procedure libraries are available, many in the public domain. A brief list includes BLAS [35], Linpack [22], Eispack [53], Lapack [3], Odepack [34], and the Harwell Subroutine Library (HSL) [37].

The Basic Linear Algebra Subroutine (BLAS) library is a collection of Fortran subroutines and functions for performing basic linear algebra tasks, including computation of norms, inner products, and other basic vector operations. The computations performed by the BLAS routines are typically key micro-kernels within other calculations and the performance of these routines can often dramatically effect the performance of the overall calculation. Consequently, BLAS libraries are available for specific computers, compiled for optimal performance. The Linpack library, developed in the 1970s and early 1980s, is a collection of Fortran subroutines for performing higher level linear algebra calculations, but rely on the BLAS routines for vector operations. Some numerical algorithms provided in the Linpack library include dense and banded matrix LU factorization and backsubstitution, condition number estimation, and computation of the determinant of a dense or banded matrix. The Eispack library is similar to the Linpack library, but provides a collection of subroutines for eigenvalue and eigenvector computation. The Lapack library also provides a set of Fortran subroutines for performing linear algebra computations. Many of the subroutines contained in the Linpack and Eispack libraries are superseded by those in the Lapack library. An advantage of the Lapack library is the use of Level-3 BLAS routines. Linpack and Eispack exploit Level-1 BLAS routines for performing optimized vector operations. However, with the advent of more sophisticated computer architectures (e.g., vector processors) much of the computational overheads associated with algorithms employing Level-1 BLAS routines involve the wasteful moving of data from one memory location to another. Level-3 BLAS are available which eliminate much of this cost by tailoring the algorithms for specific computer architectures. Another Fortran library readily available is Odepack. Odepack provides a set of Fortran subroutines for the numerical integration of stiff and nonstiff ODEs. The subroutines in Odepack exploit both the BLAS routines and linear algebra routines described previously. Another subroutine library is the Harwell Subroutine Library (HSL). The HSL provides a large collection of Fortran subroutines for performing tasks from sparse linear algebra to nonlinear optimization. The numerical algorithms employed in all of these libraries illustrate a common trait in numerical library design: more complex numerical algorithms are built on top of optimized lower level computational kernels. This set of libraries is by no means exhaustive, but does illustrate the availability of a number of procedural libraries available to the engineer for performing routine tasks within an overall numerical application.

The advantage of the procedural library is that quite efficient code may be produced, without any of the additional overheads associated with the other approaches described later. In addition, the use of procedure libraries is familiar to many computational scientists trained in the use of procedural programming techniques and not familiar with modern programming paradigms. Consequently, there is essentially no learning curve that must be overcome when using most procedure libraries.

There are a number of disadvantages to using solvers in the form of a procedure library. In typical procedure libraries, each solver has a specific interface (i.e., argument list and return value). This fact makes it somewhat difficult to swap between different solvers within an application (e.g., switching from an explicit to an implicit ODE solver). This problem is somewhat mitigated if a variety of solvers are provided from a given library and interface conventions are adopted. In addition, the varying calling standards between different languages (even different compilers for the same language) make mixed-language programs difficult to create, debug, and port to other computers. Tight coupling and high cohesion between subprograms often associated with procedural programs makes the evolution and maintenance of software difficult. Even minor changes in one section of an overall application (e.g., swapping between two different classes of solvers) can have rippling effects that propagate throughout the entire application. As software systems become more complex, they involve the contributions of multidisciplinary teams of experts. This problem is exacerbated by the use of more sophisticated computer architectures (e.g., parallel computers). The characteristics of procedural programming techniques make it very difficult to manage complexity of large and evolving software applications. Nevertheless, the performance possible with optimized procedure libraries will make this approach necessary for some time, particularly for key computationally intensive kernels.

3.2.7.2 Object-oriented Libraries

As stated above, as scientific computations become more complex, they typically involve the contributions of a multidisciplinary team of experts. The increasing complexity of the application is coupled with performing the computations on more sophisticated computer architectures. Procedural programming languages provide little language support for managing this complexity. Consequently, new programming paradigms have evolved in the computer science community. Specifically, there has been a move from procedural based languages, such as C, Fortran, and Pascal to object-based and object-oriented languages, such as Ada, Object Pascal, C++, and Smalltalk. In object-oriented design, the program is not thought of as the application of a set of algorithms performed in a well-defined order, but rather the interaction of objects. These objects, which encapsulate data and algorithms applied to the data, enable a higher level of abstraction in order to manage complexity. A growing number of object libraries are available, including Blitz++, Diffpack, ISIS++, POOMA, Overture, OPlus, and Lapack++. Several of these libraries are designed for applications on parallel computers.

As stated above, the use of objects rather than algorithms enables a higher level of abstraction, reducing the complexity of the overall application. Properly designed, these objects hide implementation details (referred to as *data hiding* or *encapsulation*). The engineer is able to focus on what transformation the object performs on the data rather than the specifics of how this transformation is performed. By hiding the implementation details properly, modifications can be made internal to the object with minimal impact to the overall application. This capability makes it much easier to reuse, extend, maintain, and evolve an application, particularly when several programmers are involved.

A price is paid for this flexibility. It is well known that code exploiting many of the features of an object-oriented language is less efficient than what is possible with code written in a procedural language. Second, well-designed objects require more than simply the use of an object-oriented language. They require the user to change the way they think about programs and how they are initially designed and implemented. Since most computational scientists are not trained in the use of object-oriented programming techniques, there is a substantial learning curve that must be overcome before object-oriented libraries may be designed and used effectively.

3.2.7.3 Component Libraries

Arguably, for large-scale, complex applications, object-oriented programming techniques have made a substantial impact. The decomposition of programs into loosely coupled objects, with proper data hiding, dramatically facilitates the collaboration of teams of programmers and the evolution and maintenance of the software application. However, complications still arise, particularly when using commercial or other third-party codes. Furthermore, the advent of heterogeneous computing environments, where different portions of a program may be running on entirely different computers, possibly with different operating systems, pushes the limit on what is possible with object-oriented programs. A common scenario is a physical property server running on one computer, the calculation engine running on another, and a visualization package running on a third. Component-oriented programming approaches address many of these issues. According to [54], "software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system." Components enable an even higher level of abstraction than objects. Proper component design involves well-documented contracts explicitly stating what information the component will receive and what transformation on the data will be performed. More complete contracts indicate computation complexity associated with the transformation. With a well-defined contract, the user of a component need not focus on the operations being performed, but simply the results of the operations. Several component frameworks are available, including CORBA [42], the Microsoft COM family [11], Sun's JavaBeans [7]. As stated above, heterogeneous computing is addressed within many component frameworks.

In contrast to procedure and object-oriented libraries, where different programming techniques are employed for the construction of the elements of the library (i.e., the use of procedure-based languages versus object-oriented languages), component libraries are based on frameworks which describe how the individual components are interfaced and linked to each other. By adhering to these standards, several different components maybe readily combined into an overall application, regardless of the origin of the components.

Several initiatives based on component frameworks have emerged for scientific computing. Three of these are the DOE Common Component Architecture Forum, the ALICE Project of the MCS Division of Argonne National Laboratory, and CAPE-OPEN, described in this book.

Similar to objects, additional overheads accompany the flexibility possible with components, particularly communication between components. In addition, many issues addressed by components are often not necessary for scientific computing. Components have been used to a greater extent within the business community where security is a serious concern. For many scientific applications, this is not a major requirement and the overheads are largely unnecessary.

Despite these disadvantages, components will play an important role in the design and implementation of large-scale scientific computing applications. Areas that will benefit from the use of components are higher level coordination tasks, monitoring, steering of the calculation, coordination between different aspects of an overall calculation, visualization, communication between different platforms.

The availability of these different programming techniques require the designer and user of a solver library to pay close attention to the structure of the overall library and individual solvers within the library. The three approaches described in this section are complementary: components frameworks for high level tasks, objects-oriented design of complicated components, and procedures for low level, computationally intensive kernels.

3.2.7.4 Automatic Code Generation

The first part of this section describes several library formats through which numerical solvers are made available to the user. With these libraries, the solvers may be readily linked into the overall application. However, often in practice, much of the time and effort expended when performing numerical calculations is spent providing additional symbolic information required by the solver. As mentioned several times in this chapter, the user must often provide analytical derivatives, sparsity patterns, and other information required by the solver. This additional information is provided in a number of ways, determined by the actual solver being used and the type of library containing the solver. For example, procedure libraries typically require the user to provide model equation residuals, derivative evaluations, etc, in the form of external procedures. Whereas, when using component libraries, the natural choice is models in the form of components which export all of the necessary information. Regardless of how this information is provided, the underlying computed quantities are the same. The remainder of this section describes automated code generation techniques that may be used to provide this additional symbolic information with minimal user intervention.

The use of automated code generation techniques to construct derivative information from computer programs has been around for some time. This effort has largely taken place within the automatic differentiation (AD) community where a number of algorithms and codes have been developed [10, 31, 50, 29]. These AD techniques operate on source code of computer programs and provide code for computing analytical derivatives by either generating new source code that may be compiled and linked to provide the derivative values or by using the operator overloading features of many languages (e.g., C++ and Fortran 90) to make the compiler generate the additional instructions within the object code for computing the derivatives. Recently, these code generation techniques have been extended to automatically construct a much broader class of information [56]. For example, given the code for evaluating the model equations, new code can be automatically constructed for determining analytical derivatives, sparsity patterns, discontinuity information, interval extensions, and others. In general, most or all of the additional information required by the solvers can be generated automatically from the original model code. These automatic code generation techniques typically apply to procedural code. However, the procedural code can the be embedded in any format required by the solver being used.

3.2.8 CONCLUSIONS

This chapter highlights several numerical algorithms used to solve common problems that arise during process modeling, including solution of systems of nonlinear equations, numerical integration and parametric sensitivity analysis of hybrid discrete/continuous systems, and optimization. The actual algorithm selected for a particular application depends both on the nature of the problem being solved and the availability of additional information concerning the model equations. If the model equations are available in symbolic form then much of this information is be readily available, removing many of these constraints. Further, recent developments in automatic code transformation techniques offers these same benefits when the model equations are available as code in a programming language such as C or Fortran.

3.2.9 REFERENCES

[1] E. L. ALLGOWER AND K. GEORG, *Numerical Continuation Methods: An Introduction*, Springer–Verlag, Berlin, 1990.

[2] K. ALLOULA AND J.-P. BELAUD, *Applying automatic differentiation to computer aided process engineering software.* presented at AD 2000, Nice, France, June 2000.

[3] E. ANDERSON, Z. BAI, C. BISCHOF, S. BLACKFORD, J. DEMMEL, J. J. DONGARRA, J. D. CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORENSEN, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, 1999.

[4] U. M. ASCHER AND L. R. PETZOLD, *Computer Methods for Oridinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, 1998.

[5] A. BACK, J. GUCKENHEIMER, AND M. MEYERS, *A dynamical simulation facility for hybrid systems*, in Hybrid Systems, R. L. Grossman, A. Nerode,̇ A. P. Ravn, and H. Rischel, eds., vol. 736 of Lecture Notes in Computer Science, New York, 1993, Springer-Verlag.

[6] M. S. BAZARAA, H. D. SHERALI, AND C. M. SHETTY, *Nonlinear Programming: Theory and Applications*, John Wiley and Sons, Inc., New York, 1993.

[7] C. J. BERG, *Advanced Java 2 Development for Enterprise Applications*, Prentice-Hall, Englewood Cliffs, N.J., 2000.

[8] D. BERTSIMAS AND J. N. TSITSIKLIS, *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA, 1997.

[9] M. Berzins, P. M. Dew, and R. M. Furzeland, *Developing software for time-dependent problems using the method of lines and differential algebraic integrators*, Appl. Numer. Math., 5 (1989), pp. 375–397.

[10] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland, *ADIFOR – Generating derivative codes from Fortran programs*, Scientific Programming, 1 (1992), pp. 11–29.

[11] D. Box, *Essential COM*, Addison–Wesley, Menlo Park, CA, 1998.

[12] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial Value Problems in Differential–Algebraic Equations*, SIAM, Philadelphia, PA, 1996.

[13] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, *Using Krylov methods in the solution of large-scale differential-algebraic systems*, SIAM J. Sci. Comput., 15 (1994), pp. 1467–1488.

[14] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577–593.

[15] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control*, Hemisphere, Washington, 1975.

[16] Y. Cao, S. Li, and L. Petzold, *Adjoint sensitivity analysis for differential-algebraic equations: Part II, numerical solution.* submitted, SIAM Journal on Scientific Computing, 2000.

[17] Y. Cao, S. Li, L. Petzold, and R. Serban, *Adjoint sensitivity analysis for differential-algebraic equations: Part I, the adjoint DAE system.* submitted, SIAM Journal on Scientific Computing, 2000.

[18] M. B. Carver, *Efficient integration over discontinuities in ordinary differential equation simulations*, Mathematics and Computers in Simulation, XX (1978), pp. 190–196.

[19] F. E. Cellier, *Combined Continuous/Discrete System Simulation by Use of the Digital Computer: Techniques and Tools*, PhD thesis, Swiss Federal Institute of Technology, Zurich, 1979.

[20] T. F. COLEMAN AND J. J. MORÉ, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM Journal on Numerical Analysis, 20 (1983), pp. 187–209.

[21] A. R. CURTIS, M. J. D. POWELL, AND J. K. REID, *On the estimation of sparse Jacobian matrices*, J. Inst. Maths. Applics., 13 (1974), pp. 117–119.

[22] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER, AND G. W. STEWART, *LINPACK Users' Guide*, SIAM, Philadelphia, PA, 1979.

[23] I. S. DUFF, J. NOCEDAL, AND J. K. REID, *The use of linear programming for the solution of sparse sets of nonlinear equations*, SIAM Journal on Scientific and Statistical Computing, 8 (1987), pp. 99–108.

[24] I. S. DUFF AND J. K. REID, *MA48, a FORTRAN code for direct solution of sparse unsymmetric linear systems of equations*, Technical Report RAL–93–072, Rutherford Appleton Laboratory, Oxon, UK, 1993.

[25] A. L. DULMAGE AND N. S. MENDELSOHN, *Two algorithms for bipartite graphs*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 183–194.

[26] W. F. FEEHERY AND P. I. BARTON, *Dynamic optimization with equality path constraints*, Industrial and Engineering Chemistry Research, 38 (1999), pp. 2350–2363.

[27] W. F. FEEHERY, J. E. TOLSMA, AND P. I. BARTON, *Efficient sensitivity analysis of large–scale differential–algebraic systems*, Applied Numerical Mathematics, 25 (1997), pp. 41–54.

[28] S. GALÁN, W. F. FEEHERY, AND P. I. BARTON, *Parametric sensitivity functions for hybrid discrete/continuous systems*, Applied Numerical Mathematics, 31 (1999), pp. 17–47.

[29] R. GIERING AND T. KAMINSKI, *Recipes for adjoint construction*, ACM Transactions on Mathematical Software, 24 (1998), pp. 437–474.

[30] P. E. GILL, W. MURRAY, AND M. H. WRIGHT, *Practical Optimization*, Academic Press, New York, 1981.

[31] A. GRIEWANK, *Evaluating Derivatives: Principles and techniques of algorithmic differentiation*, SIAM, Philadelphia, PA, 2000.

[32] E. HAIRER AND G. WANNER, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer, New York, 1991.

[33] J. L. HAY AND A. W. J. GRIFFIN, *Simulation of discontinuous dynamical systems*, in Proceedings of the 9th IMACS Conference on Simulation of Systems, 1979, L. Dekker, G. Savastano, and G. C. Vansteenkiste, eds., North-Holland, 1980.

[34] A. C. HINDMARSH, *ODEPACK, a systematized collection of ODE solvers*, in Scientific Computing, R. S. S. et al., ed., Amsterdam, 1983, North Holland, pp. 55–64.

[35] C. L. LAWSON, R. J. HANSON, D. KINCAID, AND F. T. KROGH, *Basic linear algebra subprograms for Fortran usage*, ACM Transactions on Mathematical Software, 5 (1979), pp. 308–323.

[36] K. LEVENBERG, *A method for the solution of certain nonlinear problems in least squares*, Quart. Appl. Math., 2 (1944), pp. 164–168.

[37] H. S. LIBRARY, *Specifications*, Tech. Rep. Release 11, AEA and SERC, 1993.

[38] T. MALY AND L. R. PETZOLD, *Numerical methods and software for sensitivity analysis of differential-algebraic systems*, Applied Numerical Mathematics, 20 (1996), pp. 57–79.

[39] D. W. MARQUARDT, *An algorithm for least squares estimation of nonlinear parameters*, J. Soc. Indust. Appl. Math., (1963), pp. 431–441.

[40] R. E. MOORE, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[41] B. A. MURTAGH, *On the simultaneous solution and optimization of large-scale engineering systems*, Computers and Chemical Engineering, 6 (1981), pp. 1–5.

[42] OMG, *The Common Object Request Broker: Architecture and specifications*, Tech. Rep. Release 2.0 July 1995, Update July 1996, Object Management Group, 1997. Formal document 97-02-25, (http://www.omg.org).

[43] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, Inc., New York, 1970.

[44] C. C. PANTELIDES, *The consistent initialization of differential-algebraic systems*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 213–231.

[45] ——, *Symbolic and Numerical Techniques for the Solution of Large Systems of Nonlinear Algebraic Equations*, PhD thesis, University of London, London, U.K., May 1988.

[46] T. PARK AND P. I. BARTON, *State event location in differential algegraic models*, ACM Transactions on Modelling and Computer Simulation, 6 (1996), pp. 137–165.

[47] A. POTHEN AND C. J. FAN, *Computing the block triangular form of a sparse matrix*, ACM Transactions on Mathematical Software, 16 (1990), pp. 303–324.

[48] M. J. D. POWELL, *A hybrid method for non-linear equations*, in Numerical Methods for Non-linear Algebraic Equations, P. Rabinowitz, ed., New York, 1970, Gordon and Breach, pp. 87–114.

[49] G. REISZIG, W. S. MARTINSON, AND P. I. BARTON, *Differential-algebraic equations of index 1 may have an arbitrarily high structural index*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1987–1990.

[50] N. ROSTAING, S. DALMAS, AND A. GALLIGO, *Automatic differentiation in Odyssee*, Tellus, 45A (1993), pp. 558–568.

[51] A. I. RUBAN, *Sensitivity coefficients for discontinuous dynamic systems*, Journal of Computer and Systems Science International, 36 (1997), pp. 536–542.

[52] L. K. SCHUBERT, *Modification of a qn method for nonlinear equations with a sparse jacobian*, Math. Comp., 24 (1970), pp. 27–30.

[53] B. T. SMITH, J. M. BOYLE, J. J. DONGARRA, B. S. GARBOW, Y. IKEBE, V. C. KLEMA, AND C. B. MOLER, *Matrix eigensystem routines - EISPACK guide*, Lecture Notes in Computer Science, Berlin, 1976, Springer-Verlag.

[54] C. SZYPERSKI, *Component Software: Beyond Object-Oriented Programming*, ACM Press, New York, NY, 1998.

[55] J. E. TOLSMA AND P. I. BARTON, *On computational differentiation*, Computers and Chemical Engineering, 22 (1998), pp. 475–490.

[56] ——, *DAEPACK: An open modeling environment for legacy models*, Industrial and Engineering Chemistry Research, 39 (2000), pp. 1826–1839.

[57] ——, *Hidden discontinuities and parametric sensitivity calculations.* submitted, SIAM Journal on Scientific Computing, 2000.

[58] ——, *Process simulation and analysis with heterogeneous models.* in preparation, 2001.

[59] S. VASANTHARAJAN AND L. T. BIEGLER, *Large-scale decomposition for successive quadratic programming*, Computers and Chemical Engineering, 12 (1988), pp. 1087–1101.

[60] S. VASANTHARAJAN, J. VISWANATHAN, AND L. T. BIEGLER, *Reduced successive quadratic programming implementation for large-scale opimization problems with smaller degrees of freedom*, Computers and Chemical Engineering, 14 (1990), pp. 907–915.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 3.3: Simulation, Design & Analysis

D. Hocking, J. M. Nougués, J. C. Rodríguez & S. Sama

3.3.1 Simulation

Simulation is today a well-known subject. It is nevertheless useful to get back to the basics and describe briefly what is understood as simulation.

Himmelblau and Bischoff define simulation as '*The study of a system or its parts by manipulation of its mathematical representation or its physical model*'[1]. Encarta provides another definition: '*The imitation of a physical process or object by a program that causes a computer to respond mathematically to data and changing conditions as though it were the process or object itself*[2].

The first definition outlines the relationship between simulation and modelling. While modelling is the construction of a (more or less simplified) mathematical representation of a system, simulation consists in the use of such model to study the system. For all but the most trivial models, simulation will require a non-trivial *solving* of the model. In spite of this distinction, modelling and simulation are increasingly been used as synonyms, on the basis that the modelling activity is almost always a pre-requisite for simulation.

There is, therefore, a strong relationship between the modelling and the simulation activities. In fact, in order to simulate a system, a model is needed. Moreover, the same system may have several models (possibly, with different levels of fidelity) that can be used for similar simulation purposes (leading to answers of varying degrees of quality).

The latter definition of simulation is interesting in the sense that it is implicitly restricted to computer simulation, leaving all other types of simulation outside of its scope. This is another example of how the computer revolution is influencing the meaning of words. The solution of any mathematical model that represents a physical process is a form of simulation, regardless of the number of equations or the method of solution. The generally accepted interpretation of simulation is the solution of a model of a system with a computer.

Simulation has a key role in the system design and analysis, especially in those systems where their intrinsic complexity makes analytical study impractical or infeasible. From this perspective, *simulation is a tool for the system design and analysis activities*. In these two contexts, simulation is used in different ways:

1. For the design process, where the objective is to develop a system that satisfies a specified set of requirements, simulation is used as an aid for the design through the prediction of the system behaviour.
2. For analysis purposes, the real system already exists (at least its representation –the model) and the objective is to use its model in order to analyse how that system behaves under a given set of conditions.

In both of the above mentioned activities, simulation aims at replacing with advantage experimentation. The cost-effectiveness of simulation is a function of the following factors:

1. The cost of performing an experiment, that is, interrogating the real physical system to measure its behaviour.
2. The quality of the information obtained in the above mentioned experiment.
3. The cost of building the mathematical model of the system.
4. The cost of simulating the system using its mathematical model to calculate its behaviour.
5. The quality of the information obtained from the simulation.

Progress in a number of dimensions is steadily pushing forward the advantages of simulation over experimentation:

1. Hardware. Moore's law states that computer speed approximately doubles every 18 months, which equates to three orders of magnitude increase in the last ten years. Simulation scale and speed have followed this trend. In addition, the cost per MHz of processor speed has decreased enormously. This has increased the application of simulation by making it more accessible and cost-effective and it is now often considered a real-time activity.

2. Software. Software Engineering has followed the evolution of the hardware, becoming more advanced and efficient. This in turn has driven a software revolution that has made the process of making the hardware perform useful work much more straightforward.

3. Engineering. Our understanding of the underlying principles and phenomena in physical processes continually increases. The accuracy and applicability of the corresponding simulations also increases.

4. Numerical methods. The scale of mathematical simulation problems solvable on a desktop terminal has increased from a few tens of equations thirty years ago to a few million today. The enormous increase in scale has forced the development of more efficient, stable and fast algorithms, such as representation and solution of large sparse matrix problems.

3.3.1.1 General approaches to simulation

Simulation tools have broadly followed one of the following approaches:

- Some of them are simulation-specific programming languages, such as Simnon from SSPA Maritime Consulting AB (Goteborg, Sweden) or Advanced Continuous Simulation Language (ACSL) from MGA software (Concord, Mass.).

- Other tools are graphical tools where the user builds and connects blocks through the inputs and outputs of each block. The type of information flowing through these connections is not pre- defined but is defined in each case. Some examples of this type are Simulink from Mathworks or VsSim from Visual Solutions.

- Other tools are domain specific simulators such as HYSYS from Hyprotech[3], Aspen plus from Aspen Technology[4] or ProII from Simulation Sciences[5], in the field of process simulation. These type of tools is specifically tailored for a certain domain: process simulation. These specificity leads to ease of use, albeit at the expense of a certain loss of generality.

The field of process simulation is to a large extent dominated by this latter type of software tools. Since the early works in the computer application to chemical process by L. Lapidus[6] (1962), E.M. Rosen[7] (1962), Rvicz and Norman[8] (1964) and other authors, over the past thirty years a significant advance was made in these area and several software packages have been developed. HYSYS[3], HYSIM[3], ASPEN[9], PROII[5], SpeedUp[10], DIVA[11], gPROMS[12], ABACUSS[13] are just some examples of products designed to free engineers from the burden of having to deal with numerical algorithms and allow them to focus instead in the model formulation and application.

3.3.1.2 Chemical Process Simulation

In the field of CAPE, the systems under consideration are physical and chemical processes, ranging from simple systems (*e.g.* vapour-liquid equilibrium calculations) to complete dynamic plant simulations suitable for training operating staff.

Steady-state simulation

Steady-state is defined as a mathematical condition whereby the properties of a system *at each point* are constant with time. This condition very rarely happens in real life. Why is then steady-state simulation so popular? The steady-state assumption removes time derivatives by setting them to zero. This provides a simpler mathematical model that is easier to understand and more adaptable to custom solution techniques.

It is worth mentioning at this point that it is often said that steady state is the state to which a system converges when t→∞, but this is incorrect. Some systems do not converge to any steady state and may only be described by their dynamic behaviour.

To illustrate a non-steady-state system, consider a pipeline in severe slugging conditions. The system exhibits changing behaviour (liquid flow rate/vapour flow rate) at each point in time, as shown in Figure 1.

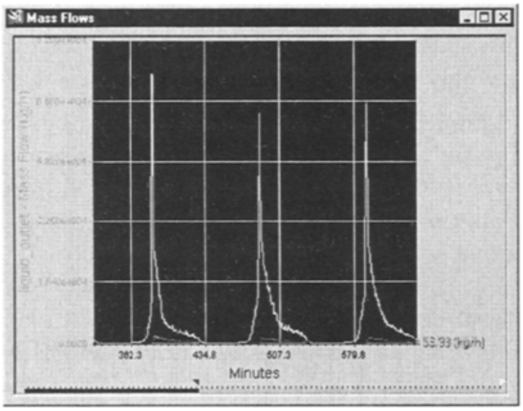


Figure 1. Evolution of gas and liquid mass flows at the exit of a pipeline showing severe slugging behaviour

Dynamic simulation

Dynamic simulation describes the time-dependent behaviour of a system. Dynamic simulation is not as widely applied as steady-state simulation. The two main reasons are the engineering complexity and the much greater computational load. As computing power increases and simulation software interfaces improve, dynamic simulation is rapidly becoming an effective tool. Some useful fields of application include:

- Dynamic studies. Steady-state simulation will provide the basic operating conditions for the plant. Dynamic simulation can provide further information on how to transition the plant between modes or hold it at a particular set of conditions.
- Control system design and validation studies. The dynamic behaviour of a process is dependent on the control scheme. Different control strategies will result in different behaviours. Dynamic simulation is required to determine which scheme is optimal and provides the advantage that the investigation is off-line. Dynamic models have a central role in model predictive control (MPC) applications. In this technology usually a black box identified models is used in the controller. The modelling and identification is the most time-consuming task during the implementation of a MPC in the industry[14]. The dynamic simulation of complete flowsheet can help in more effective deployment of the MPC applications, as this type of control usually involve plant wide control problems and/or strong interacting systems. MPC employs a simulation model of the plant to predict how the plant will behave in time with manipulation of the control variables. A MPC solves an optimisation problem at each control cycle, attempting to minimize the deviation from the setpoint(s) by manipulating the control variables. The MPC may predict several steps into the future to refine the optimisation. MPC permits a shift from feedback control that is reactive, to predictive control that is proactive.
- OTS (Operator Training Systems). OTS brings the concept of a flight simulator into the process plant. The goal is to train the plant operators in the use of the real plant by using a simulation model. The simulation model must be very detailed to replicate the real plant exactly.

Optimisation

Optimisation determines the values of a set of independent variables (*e.g.* temperatures and pressures) that minimize a real-valued objective function that is usually based on cost. Other optimisation applications are described below:

- Mixed-integer optimisation. In this technique, the independent variables are a combination of real-values (e.g. the reflux ratio) and integer-values (*e.g.* feed tray). The optimisation determines the best integer and real values for the problem, for example the feed tray location for the lowest reflux ratio. Heat exchanger network synthesis is another mixed-integer problem.
- Dynamic optimisation. This is similar to steady-state optimisation except that the objective function is also time-dependent in some way. Dynamic optimisation is computationally demanding. The MPC application described above is a form of dynamic optimisation. The objective function is

the weighted sum of the setpoint deviations. The assumption is that the setpoints are themselves optimal.

- Real-Time Optimisation (RTO). This is the direct application of a steady-state optimisation to a plant control system. On a suitable time cycle (every few minutes or so), the steady-state simulation is updated with appropriate plant conditions and optimised. The optimal control values are then fed to the plant control system and the process continues. This is effectively a dynamic optimisation with a steady-state model. To be effective, the optimisation time cycle must be considerably smaller than the time constants of the plant being controlled.
- Data reconciliation and parameter estimation. Data reconciliation uses statistical techniques and a simulation model to identify errors in plant measurements. Parameter estimation uses reconciled data and the simulation model to determine accurate equipment parameters such as heat transfer coefficients (fouled/not-fouled *etc*). The equipment parameters may be used for dynamic control studies, optimisations, fault identification and determination of maintenance cycles.

3.3.1.3 Simulation and model fidelity

It was mentioned earlier that the size of simulation problem we are able to solve increases with computing power. This has a significant impact on the level of detail in the simulation model. As more detail is incorporated, the computational load increases. The extreme example is CFD (Computational Fluid Dynamics), where the number of equations can be in the millions.

Traditional simulation models are based on lumped first-principles analyses, employing mass and energy balances across pieces of equipment with a thermodynamic characterisation, usually based on equilibrium. Basic equipment information and geometry is incorporated if required. Recently, neural net applications have been embedded into simulation packages to enable parameterisation of complex, computationally-expensive models for high-speed execution. These models have the same fidelity as first-principles models, but they trade off the first-principles extrapolation capability for speed.

At the other end of the spectrum, a high-fidelity CFD model requires detailed geometry, material properties and flow paths to be truly effective. Anomalies such as hot spots, stagnation points and high-shear conditions may be determined with CFD modelling. The associated computational cost is high.

Longer-term, a component-based architecture is the only way to provide a variety of model fidelities in simulation. A generic framework that permits a calculation granularity much finer than a macro unit operation is required. A particular vendor might provide an exceptional heat-transfer correlation, but a mechanism to deliver that calculation is required to make it globally beneficial.

3.3.2 DESIGN

A design problem exists when the desired end result is known, but the means or process leading to the end is not. Simulation is usually an integral part of the design process, in fact it is difficult to imagine a design process that does not incorporate simulation at some point. Traditionally the design work-process progresses linearly through conceptual, process, detailed and costing. Recent advances in CAPE software promote the concept of moving through the work process as requirements dictate rather than restricting the workflow to a linear progression.

3.3.2.1 Conceptual design

This is the first step in the design process, where the major process units required to achieve the end result are selected, *e.g.* reaction, separation, compression etc. Basic thermodynamic information is required to determine approximate operating conditions. Several alternatives may be screened at this point to determine candidates to move forward into process design. Conceptual design is a steady-state simulation process.

3.3.2.2 Process design

Process design occurs after the conceptual design. The major process units are decomposed into smaller groups of units with more equipment and detail. Reasonably precise operating conditions are determined. This is predominantly a steady-state simulation process with some optimisation. However, there are great advantages to performing dynamic simulations at this stage to design the basic control scheme for the plant, as rework is minimised and control-specific issues such as time constants may be identified early on.

3.3.2.3 Detailed Design

Detailed design completes the equipment specifications and 3-D layout for the plant. Complete PFDs, PIDs and construction diagrams are developed here. Simulation at this stage can form the basis for operator training, plant commissioning and online applications.

3.3.2.4 Cost Estimation

Cost estimation aims at producing an estimate for all the costs involved in a plant construction and operation. Obviously, the cost estimation can be performed with different levels of fidelity, as a function of the availability of information.

3.3.2.5 Impact of the Web in Design Activities

A detailed analysis of the impact of the web on the above described activities is beyond the scope of this article. Nevertheless, this section will mention a number of changes that the web may be introducing in the way these activities are carried out today.

As a general principle, the Web has greatly reduced the transaction times between the different stages in the value chain of most activities. In the field of CAPE, the Web has extended the reach of the C in the word CAPE: computers are not anymore restricted to one single stand-alone computer or to an intranet network, but they can now automate processes across companies. The impact of these automation of processes is very relevant in a number of design activities. The viability of these improvements is heavily dependent on the availability of robust standards that enable CAPE software tools to work across the internet (and therefore across companies).

Modelling complex processes

There are many process units such as cat crackers, reformers, and ethylene furnaces that require complex models. These models are often available commercially, but are generally not integrated into an overall simulation package. Someone wishing to incorporate such a process unit into his design must either develop his own model or integrate a commercial model into the simulator, either alternative will take significant time and effort.

The availability of standards such as CAPE-OPEN can alleviate such problems. The individual model vendors can provide their equipment models with an appropriate interface. The process engineer can then download the models he needs to complete his design. This concept extends beyond equipment models. Models for proprietary processes or materials could also be downloaded.

Equipment selection

The availability of standards also makes possible the very appealing scenario described hereinafter.

The process engineer works in the process design, which consists of a number of *ideal* building blocks (unit operations). At that stage, aspects like the need of some ancillary pieces of equipment (*e.g.* pumps to lift fluids to the top of a column) are not considered. At the detailed design activity, this ancillary equipment is incorporated into the process design. Next comes the equipment selection, which consists of replacing the abstract pieces of equipment by real models available in the marketplace.

The equipment selection, once the detailed design is available, is a task that usually involves contacting a number of equipment providers. Conjunction of the web and standards such as CAPE-OPEN make it possible, at least in theory, that the process of equipment selection be automated through the use of equipment selection agents.

Equipment selection agents would take care of replacing a flowsheet composed of models of abstract unit operations by one (or more) flowsheets composed of models of real pieces of equipment (that is, pieces of equipment available in the marketplace). The agent would communicate the requirements of the equipment to the equipment provider server. The process would end up (hopefully) with a number of alternatives fulfilling the requirements set up by the detailed process design activity. Those alternatives would differ in aspects such as the capital expenditure vs. operating expenditure tradeoff, safety, reliability, *etc.*

The usefulness of the standards becomes apparent when considering that equipment providers may decide to make available detailed simulation models of their pieces of equipment. The equipment selection agent would then be able to download a number of such simulation models, all of them meeting the requirements defined by the detailed design abstract model. A generic (abstract) unit operation is then instantiated in a number of (alternative) models of real unit operations provided by the equipment manufacturer. Standards like CAPE-OPEN ensure that the abstract simulation model can be seamlessly replaced by one of a set of alternative concrete simulation models.

The replacement of an abstract unit operation simulation model by a real, concrete one enables the process designer to rerun the simulation model with a more accurate description of the plant and thus identify effects that didn't show up with the abstract model.

While the practical implementation of the above described scenario is not available yet, it is thought to happen in the near future.

As a consequence of the equipment selection process, cost estimation can become cost calculation, as simulations are based on models of physical equipment with a defined purchase price.

3.3.3 ANALYSIS

Once one or more plant designs are available, simulation allows the engineer to perform a number of analysis activities. These analysis activities share the goal of *testing* the behaviour of the virtual plant for a number of parameters:

- Controllability. It is well-known that a good process design from a steady-state perspective is not necessarily controllable.
- Start-up and shutdown operations. A dynamic model of the plant is required for the analysis of start-up and shutdown operations.
- SHE studies. CAPE tools are extremely useful for performing Safety, Health and Environmental studies, such as:
    - <u>HAZOP/HAZAN</u>. A standard process design activity is a HAZOP/HAZAN studies on the plant equipment. A CAPE tool is very useful in simulating the equipment behaviour under the conditions determined by the HAZOP/HAZAN studies.
    - <u>Emission studies</u>. In some cases, a number of tools are used co-operatively in order to analyse, for example, the evolution of a toxic cloud caused by a leak in the plant. Typically this problem is analysed by using a combination of process simulator and CFD tool, such as in Figure 2 below.

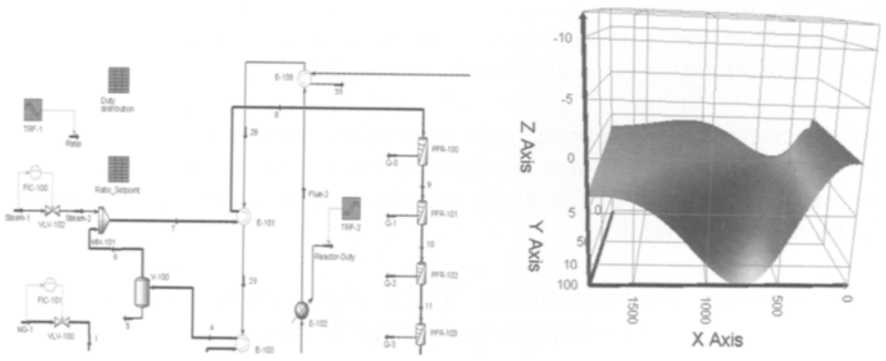


Figure 2. A dynamic model of a steam reforming plant demonstrates the behaviour of the plant during an upset. The feed and feed preheat system are modelled using first principles models for all the equipment. The reactor is implemented with a 1-dimensional pseudo homogeneous model of the reactor tube and uses an approximated model for the furnace side. It takes into account the radiation heat transfer, the furnace wall thermal lag and the combustion of the fuel gas. The control system reproduces the actual plant control system. The plot

shows the impact of a sudden 10% increase in feed rate on the reactor temperature profile. The X-axis represents time, the Y-axis represents the length of the reactor and the Z-axis shows the deviation of temperature from the nominal conditions.

3.3.4 SYSTEM FRAMEWORKS

3.3.4.1 Business requirements

The scenarios described above require realistic mathematical models and software tools that make constructing and using the models a cost-effective process. Modern system frameworks aim at achieving such cost-effectiveness, almost without exception, by being designed allowing for reuse. Reuse needs to be enabled in two fundamental, complementing areas:

- Model reuse. Usually companies will have invested important resources in building the model of the system under study. It is expected that several simulation tools will be able to use the same model, even if they will make different use of it.

- Simulation tool reuse. Similarly, companies may have invested important resources in the development of a special-purpose simulation tool. It is also a reasonable expectation that such special-purpose tool can be re-used in the context of a variety of more general-purpose simulation tools. As an example, a specialised reactor model could be used inside a conceptual design tool or inside an optimisation tool.

The above high-level reuse requirements become more specific depending on the following factors.

- Model reuse throughout engineering disciplines. The tools need to allow each engineering discipline to work co-operatively, ideally to *add knowledge* into a common model. Examples of such engineering disciplines are:

  - Thermodynamics
  - Process design
  - Costing
  - Equipment selection
  - Control engineering

- Activity sector. Process licensors, engineering, procurement and construction (EPC) companies and operating companies pose completely different requirements on the tools and their models.

- A process licensor may be interested in using modelling tools both internally and as an additional, differentiating item in its product portfolio whereby the process model would be provided to their customers. For this latter application a restricted version of the model may be required, so that the end user does not have access to information that is considered as confidential by the process licensor. In addition, process licensors will usually have their own special-purpose in-house simulation software and will require them to be plugged in the simulation tool.
- An EPC company may be particularly concerned about workgroup capabilities in the modelling tools they use. Among other aspects, this may pose versioning, auditing trail, etc, requirements on the tools. EPCs will often have as well in-hose software that is expected to interact efficiently with the main simulation tool.
- An operating company may make a variety of uses depending on the department. Operating units in such companies are usually resource constrained and may value robustness and ease of use of the simulation tool over flexibility and complex features. These latter capabilities may, on the contrary, be the preferred ones by central engineering departments in order to allow them assessing innovative, non-standard, design alternatives.

- Industrial sector. While each industrial sector has different needs, it is impractical to build specific tools from scratch that will address just one sector's needs. Modern system framework architectures may support the creation of specific purpose tools built on top of general purpose ones by providing them with a number of additional features. The following is a list of industry sectors briefly mentioning their specific needs.

    - Upstream Oil & Gas. This sector is particularly demanding for dynamic simulation and for the simulation of complex hydraulic systems such as pipeline networks.
    - Refinery. The refinery sector requires capabilities for the characterisation of different crude oil types and also modelling capabilities for a number of specific reactor types (hydrodesulphurization, fluid catalytic cracker,...). As above, the basic simulator's architecture may address these challenges by allowing for plugging-in specific modules to address these needs.
    - Petrochemicals. The petrochemicals sector has specific requirements for modelling some types of petrochemical reactors (*e.g.* ethylene crackers). These requirements may be handled similarly.
    - Chemicals. The chemicals sector has a very wide variety of requirements, among them, and just to mention a few, it requires electrolyte modelling capabilities and similarly batch processes modelling capabilities. These requirements are not trivial to fulfil:

issues such as electrolyte simulation capabilities may require in some cases serious re-engineering of the simulator.

- Stages in the life cycle. Different stages in the process life cycle pose different requirements on the model and its simulation tool. Modern system frameworks share the common goal of allowing seamless porting of the model from one stage to the following one:

  - Conceptual design.
  - Front-line engineering.
  - Detailed engineering.
  - Equipment selection and costing.
  - Operability analysis.
  - Operator training.
  - Performance monitoring, maintenance
  - Revamp studies, optimisation

- Rigor. The required level of model fidelity is not uniform in all the activities, but depends to a large extent on a number of factors. Some operational problems, in particular, may require the use of complex mathematical techniques such as computational fluid dynamics, while in other applications (like the preliminary screening of distillation sequences) a much lower model fidelity is perfectly acceptable. The architecture of the modern system frameworks needs to be able to accommodate different levels of model rigor and to allow the smooth transition between these different rigor levels.

The above mentioned requirements are reflected in the following dimensions that simulation system frameworks must address satisfactorily:

1. Fidelity. Fidelity is the amount of equipment and process detail in a model. A qualitative assessment of a separation process requires relatively little detail; an operator trainer requires a higher level of detail; the detailed analysis of a mixing process requires yet additional level of detail.
2. Scale. The scale of a model is a significant factor. A small-scale model might only model one section or unit of a plant. A large-scale model might cover an entire plant or many plants. While it is clear the impact of the scale in the computational requirements, often the scale of the problem also impacts radically the suitability of the whole simulation methodology.
3. Performance. The performance is a function of the scale and fidelity of the model. The software must support the engineering activity in the time horizon that makes the activity useful. It is interesting to note here that the whole concept of *time horizon that makes the activity useful* is in some cases an absolute one but in some other cases a relative one. Quite

logically, an operator training application will be required to execute at least at real-time speed, which constitutes the perfect example for an absolute performance requirement. On the other side, in many other applications, it is the market forces of both competition and supply of better and faster tools that continuously raises the expectations and requirements of the engineering community, what provides a *relative* performance level. The qualitative separation assessment mentioned above is a small-scale, low-fidelity model that is today *expected* to solve quickly (in the order of seconds). Similarly, an on-line optimisation usually must be solved in the order of minutes in order to be useful, but for the (off-line) analysis of a mixing process, longer computation times (of the order of hours, or even days in some cases) may be acceptable. As can be inferred, the evolution in the computing power is continuously pushing the expectations of the engineering community towards shorter response times and/or more complex applications.

4. Interface, communication and usability. The software tools used for modelling applications now operate within a complex corporate environment of software systems. Modern tools are expected to communicate with complementary applications and provide extension capabilities. In addition, software tools are expected to be straightforward to use and support the engineer as they work.

3.3.4.2 Basic technical approaches

The most important task in process simulation is the solving of the large set of equations that represents the flowsheet. For all the existing solving methods the initial point is the block diagram of the model. Taking into account the approach used to solve the set of equations, all the current process simulators fall into the following categories.

- Sequential/Modular approach. In this approach each unit operation is represented by a set of equations grouped into a block (or module) and the whole flowsheet is solved one a module-by-module basis (in sequential way). Aspen, Process and FLOWTRAN are examples of the application of this approach.
- Simultaneous/Modular or Two-Tier Approach. The basic idea of this approach was first developed by Rosen[7] and is based on a simplified, linearised set of equations for each unit operation, allowing the solution of interconnected unit operations simultaneously.
- Simultaneous Solution or Equation Oriented approach. The main idea of this approach is to collect all the equations and solve them as a large system of non-linear algebraic equations. QUASALIN and SpeedUp are examples of the application of this approach.
- Non sequential modular approach. This approach combines the sequential modular approach with bi-directional information flow and degree of freedom

monitoring techniques, allowing the interactive construction and solution of the flowsheets[16].

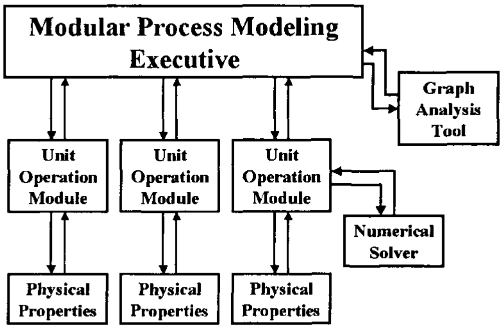


Figure 3. Architecture of a sequential/modular simulator[17]

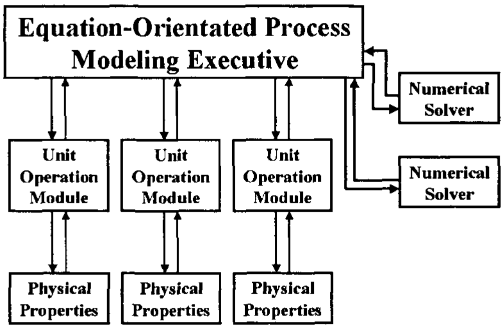


Figure 4. Architecture of an equation oriented simulator[17]

Traditional software architectures relied heavily on the use of monolithic software applications that tried to address all requirements from within a single-source application. Modern software architectures have become focussed on delivering functionality through the combination of software components from different sources.

Many software tools provide some of the capabilities mentioned above but not all of them. There are two broad categories of CAPE software. The first one delivers narrow, well-defined functionality, such as the computation of physical properties, the simulation of a particular unit operation, or the numerical solution of certain types of mathematical problems. These may be described as Process Modelling Components (PMC). The second category of tools provides an environment that supports the construction of a process model from first principles and/or libraries of existing models. The user may then perform a variety of different tasks, such as process simulation or optimisation, using this single model of the process. This type of tool is a Process Modelling Environment

(PME) and is usually composed of many smaller PMCs working together co-operatively[17].

Enabling technologies such as COM or CORBA define a low-level communication mechanism that is software-independent. Good software design principles, systems analysis and standards such as CAPE-OPEN define domain-specific, granular interfaces for software tools. The combination of these makes it easier to develop portable PMCs that are reusable within different PMEs and provide a portal for third-party development of PMCs.

The interoperability between PMEs and PMCs is made possible by a set of enabling standards. Even after due simplification, at least two levels of standards need to be distinguished:

1. The binary layer, *e.g.* CORBA, COM.
2. The domain layer, *e.g.* CAPE-OPEN, pDXI.

While the first, lower-level layer, is a general purpose one required for enabling component-based software (and is therefore used by literally thousands of software products), the second layer, built on top of the first one, provides the domain-specific standards that make it possible meaningful communication between software components.

The first layer provides the fundamental means of communication between the software components. A physical analogy would be the telephone, which facilitates communication between people. Such communication is useless if the participants do not speak the same language, however. This is the purpose of the second layer, which provides the *semantics* so that pieces of software can exchange information meaningfully.

The meaning of the above may be clarified with the following example. COM defines programmatic standards so that a certain *property* residing in a server software component can be accessed from a client component[18]. CAPE-OPEN (or, for the same purpose, pDXI) may state that *Fugacity* is one such property that a certain component needs to be able to recognise.

3.3.4.3 The problem of the degree of standardization

The domain-specific standards described above need to be carefully designed with two apparently opposed criteria in consideration:

- They need to be as easily applicable to each specific existing software product as possible.
- They must have as wide applicability as possible.

The first criterion calls for simplicity while the second one calls for complexity. Using a mathematical analogy, while on one hand a *least common multiple* of the existing software is ideal to suit meet the first of the criteria, on the other hand, a *greatest common divisor* approach is better in order to meet the second criterion.

For example, some thermodynamic components do support and expose for access by their clients the property Fugacity while some others do not. A naïve interpretation of the first criterion would recommend that the domain-specific standard does not mention such property. Similarly, a naïve interpretation of the second criterion would indeed recommend Fugacity is a recognised property by all the standard-compliant thermodynamic components.

The solution to the above apparent incompatibility resides in the standard taking a pragmatic approach:

- Recognising the fact that some thermodynamic components may not expose some properties such as *Fugacity* (be it because they do not use such properties at all, be it because the authors have decided not to make them available to the outside world), the standard may state that some properties do not necessarily need to be supported by a certain standard-compliant component.
- Recognising the fact that some pairs of client and server components may benefit from using such properties, the standard defines some mechanisms for *handshaking*, that is, mechanisms so that such *Fugacity-aware* client and server can recognise such *Fugacity-awareness* in their counterparts and make use of the Fugacity property. Among other things, this requires:
    - Uniqueness. *Fugacity* needs to be a universally recognised unique tag name for the property meant. That is, there shouldn't be interfering varieties of the same property name such as *fugacity* (lower case initial letter).
    - Fault tolerance. A component that does not support Fugacity need to fail gracefully when asked for such property by a client. The meaning of failing gracefully needs to be specified by the standard in order for the above mentioned handshaking to be possible.
    - Extensibility. The standard needs to recognise the fact that the wealth of human knowledge is an ever-increasing one, and thus it needs to define the mechanisms so that the standard can be evolved and extended in order, for instance, to accommodate new properties (such as *FugacityCoefficient*) that may come up after the standard was initially established.

Fortunately, the above problems are common place in today's software and are not restricted by any means to the world of process simulation. An excellent example of a similar situation can be found in the rise of XML versus html[17,19,20]. While html was planned as an all-inclusive standard, and thus required a number of updates in order to adapt such standard to new findings or requirements, XML was designed as intrinsically extensible. XML's de-coupling of structure (the DTD file) and contents (the XML file) allows for self-documentation of the *lexicon* being used.

The techniques outlined above tackle the issue by requiring a minimum set of services in order for a component to become standard-compliant, but also allowing the component to exceed such minimum set and also allowing its client to make use of such extra capabilities. Following with the mathematical analogy, the standard *requires* the least common multiple but allows the use of the greatest common divisor if two interacting components so *decide*.

3.3.4.4 The Problem of Granularity

The concept of granularity is one that is gaining relevance as component-based software architectures become popular. In essence granularity consists in the level of aggregation exposed by a component to other external collaborating pieces of software. A thermodynamic calculation component may be designed in order to allow a programmatic client full access to detailed configuration features such as binary interaction parameters. This would constitute an example of fine granularity. Alternatively, the same component may be designed to restrict the interaction with external programmatic clients to the use of just a few methods, explicitly preventing such client from accessing information considered as private such as the mentioned binary parameters. This would be a case of coarse granularity.

The level of granularity of a software component is the result of a number of factors:

- Existing technology on which the component is built upon. Some architectures naturally allow for finer levels of granularity, while monolithic architectures usually allow only for coarse granularity levels.
- Intended interaction with clients. Even if the architecture allowed for fine granularity, it may be a conscious decision to implement a component in a coarse granularity fashion in order to *steer* the interaction between client and component through a relatively small number of well-controlled methods.
- Confidentiality issues. Coarse granularity may be the preferred option when the developers of the component aim at providing their clients with some useful functionality and yet want to keep confidentiality on some aspects that a finer granularity choice would effectively disclose.

In practice, apparently minor differences in the levels of granularity may lead to very different *use cases* supported by the components involved. Following the example of the thermodynamic component above mentioned, while the fine granularity one may support entirely programmatic *configuration* by a client component using the standard, the coarse granularity one may only support programmatic *execution*. While this may seem as a minor difference, it has several practical implications:

- In the fine granularity case a thermodynamic package may be created from scratch and fully configured (*i.e.* compound slate selection, single component parameters, binary parameters,...) and used programmatically using the standard. In the coarse granularity case, though, either human or non-standard interaction (*i.e.* through proprietary means) will be required for the same purpose. When this is considered in the context of support for automating tasks (*e.g.* a situation where an executive program is dynamically reconfiguring such thermodynamic component), the difference in capabilities' consequences become clear beyond the apparently minor differences.
- As a consequence, some applications are just not supported by a coarse granularity component. Optimisation of the binary parameters in order to meet some experimental results, for instance, would only be possible with the fine granularity component.

The flexibility introduced by a fine granularity architecture brings some additional problems, though. In particular, as mentioned above, the developers of a component may consciously decide to provide only a coarse level of granularity in order to ensure adequate use of the component. Finer granularity may provide the client of the component with so many use options that would make the component, in practice, very difficult to use or maintain.

Following the example above, a fine granularity component providing access to the binary interaction parameters has a number of requirements that the coarse granularity one does not have. In particular, and to start with, the level of documentation required by the fine granularity one is much more detailed, not just regarding the description of how to use the interfaces in order to access the mentioned parameters, but also in order to describe *exactly* what equation of state the binary parameters are regressed for. Even the smallest differences between the form of the equation of state actually *implemented* by the component and the one *assumed* by the client may lead to completely wrong results.

The situation is even more complex when the fine granularity component allows not just for access but also for mix-and-match replacement of some of its built-in sub-components by external ones. The same example of the thermodynamic component is also suitable to illustrate this situation. A fine granularity implementation may allow the replacement of the individual properties calculation engines (methods, subroutines,...) by external ones. But in the field of thermodynamics it is well-known that some properties are theoretically related, and replacing the calculation of one of them without adequate replacement of also the other one may lead to serious inconsistency, as both quantities are strongly coupled (heat capacity, enthalpy and entropy provide one of the best examples).

3.3.4.5 Example 1: Hyprotech

Hyprotech's engineering framework is focused on the *model-centric approach*. In this paradigm, the different engineering activities use different views of a common model. Moreover, different engineering activities (*e.g.* conceptual design, dynamic simulation,...) use different software tools around the same basic model. These software tools work co-operatively on that common model, for example:

- A conceptual design tool is used to select a process topology from a number of design alternatives.
- A steady-state simulator is used to tune the basic process model to the desired process conditions and to assess the flexibility of the process.
- An optimiser is used to find the best operating conditions.
- A dynamic simulator is used to perform controllability studies on the process model.
- Links to DCS's and to real-time databases are used for applications such as Operator Training Systems (OTS) and Performance Monitoring.

Hyprotech's products support the above mentioned business requirements through a variety of techniques:

- Model reuse throughout engineering disciplines. This business requirement is directly addressed by the model-centric approach. Each engineering discipline adds knowledge to a common model, which is therefore being refined with more detail.
- Activity sector. The product can be adapted, departing from a common base product, by extension with special-purpose modules. A process licensor may add to the basic simulation its own proprietary model for a certain reactor, for instance. The architecture allows for such reactor to hide confidential information to the final user.
- Industrial sector. Similarly as above, the basic means for addressing the variety of needs throughout different sectors is by allowing the basic process simulator to be customised for each sector by adding relevant features. In this context, for instance, a specific product for refineries, called HYSYS Refinery, is nothing more than the customisation of the basic HYSYS with a number of refinery reactor models developed in co-operation with a domain expert partner.
- Stages in the life cycle. Once more, a set of optional capabilities, engineered as optional components compatible with the basic simulator, addresses the needs of the different stages in the lifecycle. It is worth mentioning that in

some cases, such as performance monitoring, the whole simulator acts as a component itself embedded in an executive system that uses simulation in co-operation with a real-time process database.
- Rigor. The requirement of scaleable rigor requires a finer granularity level than the above other requirements. In practice, scaleable rigor means that a certain model, at the unit operation level, can be replaced by a higher or lower fidelity one, usually *keeping the rest of the flowsheet unchanged.* In practice this requires the unit operation model to have been engineered for such underlying variety of rigor levels. This is achieved in HYSYS by decoupling the common aspects that all unit operations need to support, no matter how rigorous the model is (*e.g.* interaction with the rest of the flowsheet) from the model internals.

The model-centric approach is possible only if the different tools can work co-operatively by interacting with the model from different perspectives. The Hyprotech architecture delivers this functionality by building process models from interconnected granular pieces that interact with each other.

Some of the key pieces involved in the above describe scenario for Hyprotech's products are:

- Common persistence layer. A common layer is used in order to grant access to the model information to the different clients. Such layer allows clients to use persistence features without explicit knowledge of the low-level implementation details.
- Thermodynamic engine. Hyprotech's thermodynamic engine is designed to provide common services to all the applications so requiring, both internal applications and also external ones. It has been chosen to provide fine granularity in order to allow users to have full configuration capabilities, including overriding some of the built-in methods. XML technology is used in order to define the run-time configuration of the engine, that is, what methods should be used for the calculation of each property.
- Hydraulics engine. Hydraulic effects are present in most unit operations. A component providing its clients with such hydraulic steady-state and dynamic simulation capabilities is available.
- Mathematical component. A component based on, but not restricted to, the Harwell Subroutine Library, is available for clients requiring mathematical services.
- Optimiser. A component providing optimisation and data reconciliation capabilities (which, in turn, makes use of the mathematical component), is available.

Apart from these components supporting the above mentioned requirements, the different products are engineered not just by using existing components, but also so that they themselves become components. From this perspective, for instance, HYSYS, which is engineered using the above mentioned components, is in itself another component that can be used inside other types of applications such as performance monitoring.

3.3.4.6 Example 2: CAPE-OPEN

In this section, CAPE-OPEN is used to illustrate a process modelling standard that helps alleviate the problems mentioned above[21].

The process leading to the design of such a standard is also described. Since it is not the aim of this work to describe in detail the CAPE-OPEN project, only two examples will be considered here. The first one concerns Unit Operation blocks and the second one deals with a thermodynamic server, here termed as *Property Package.*

The key point of *component-ware* process modelling is *abstracting* the behaviour of the different *packages* participating in the system. This implies identifying the behaviour that is common to a class of packages (*e.g.* the class of unit operations) and their relationships with the other packages (*e.g.* a unit operation is a *client* for a thermodynamic *server*).

This abstraction exercise will allow partitioning a monolithic simulation tool into smaller entities with similar behaviour. This, in the end, can be used to replace these entities with others of the same class to achieve customisation.

The abstraction begins by simply putting on paper the list of requirements (*i.e.* the functionality) that a class of components needs to fulfil. An exhaustive list of requirements will allow creating a "class diagram" of the component (*i.e.* a picture of the functionality of the component, in terms of actions that the component is able to perform).

A further simplification of the "class diagram" will discard all internal actions of the component. This will be considered as model details, and represents the functionality that clients of the component do not need to be aware of. The result of this simplification is called "interface diagram".

An interface diagram reflects what happens in the boundaries between modelling components. An interface diagram is the final result of the abstraction process and 1) gives a clear indication of the common functionality of a class of simulation components and 2) establishes the rules of how the other components have to communicate with the component exposing the interface.

Once the information to be exchanged in the boundaries of the various components has been defined in terms of the component interfaces, this can be expressed in terms of software artifacts that will be used by programmers to create an standard simulation component or to re-engineer and existing model. In either case, the final goal is to achieve "plug-and-play" of these components to create a composite modelling environment respecting a set of agreed standards.

Example 2.1 - Unit Operations

The CAPE-OPEN abstraction process for steady state sequential/modular unit operations resulted in a design similar to the one represented in Figure 5, which illustrates how a FORTRAN reactor model is re-engineered to follow the CAPE-OPEN standard. There is a main *entity* here called *Unit Operation* that contains the engineering code (*i.e.* encapsulated as FORTRAN code). Unit Operation exposes a set of CAPE-OPEN interfaces that are common to all unit operation components (interfaces are represented by lollypops).

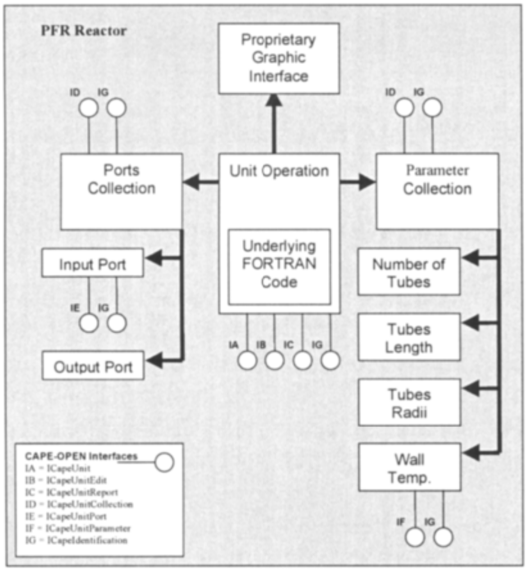


Figure 5. Structure of a unit operation component created according to the CAPE-OPEN standard

Unit Operation uses other smaller aggregate *entities*; such as port and parameter collections, ports and parameters. Ports implement the connectivity, while parameters are the way Unit Operations use to expose their variables (in the reactor example, these are number of reactor tubes, length of the tubes, wall temperature and tubes radii).

Figure 6 shows how this simulation component can be used within a commercial simulation package that is aware of the CAPE-OPEN standard. The PME and the PMC are independent entities working co-operatively in a customised environment. Both can have their own configuration or reporting mechanisms (*e.g.* graphical interfaces). The information that has to be transmitted between them in order to carry out the simulation is defined by the standard interfaces.

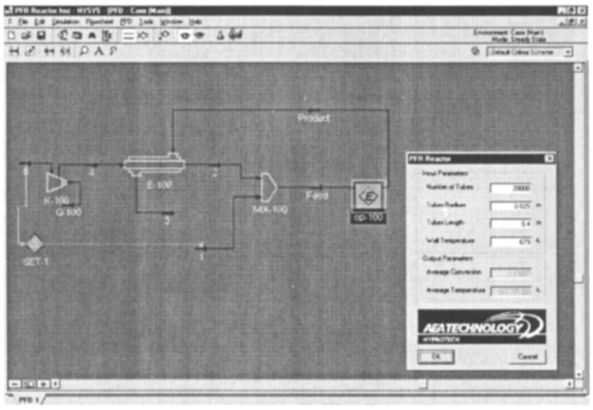


Figure 6. Using a CAPE-OPEN component within a commercial simulation package.

3.3.4.7 The Promise of Innovative Software Architectures

Initiatives such as the CAPE-OPEN standards are designed in such a way that parallel developments in software technology can be easily leveraged. As a consequence, internet-based technologies are opening up several important opportunities, some of which have already been outlined in the previous sections in this paper. In this section some of the anticipated short-term improvements will be outlined from the technical perspective.

The implications of the so-called *internet revolution* in all types of activities has today been recognised by most companies. Software tools that simplify the development of internet-aware software applications are becoming common place. In particular, Microsoft is endorsing the .NET framework as the preferred architecture for web-based applications[22]. The promise of .NET is similar to the promise that operating systems such as Windows made in the past: providing a unified framework for the development of applications. While in the case of Windows, the developers were suddenly freed from the need of designing from scratch user interface tools, in the case of .NET the promise is that developers will have a framework of reference so that they will not need to solve web-related architectural issues by themselves, but they will just have to follow the relatively easy-to-follow rules of such .NET architecture.

Clearly, from the development perspective, what these novel architectures will bring is replacement technologies for the binary layer of standards mentioned above. Standards such as COM or CORBA are to evolve into new standards that specifically provide support for web-based tools, allowing for the next step in component software, namely, true local/remote transparency.

Web-based local/remote transparency will allow different software components to be executed from different computers connected via web. The implications of these new possibilities have been briefly outlined in this paper. Everything will be a web service, including PMCs and PMEs.

A description of one sample scenario may help get an accurate idea of the implications.

As mentioned in this paper, the world of simulation has evolved into PMEs and PMCs. The combination and inter-operability between them is providing value beyond the possibilities of one PME or PMC in isolation. The current standards (e.g. COM/CORBA in co-operation with the CAPE-OPEN ones) allow for PMCs and PMEs to be *developed separately* and then *installed on the same computer and used in co-operation.*

In the forthcoming architectures, the step of installation is completely skipped, as in order for a PME to use a set of PMCs, physical installation will be irrelevant: the PME will be able to access the co-operating PMCs through the web in a seamless manner, as if the mentioned PMCs reside on the same computer.

Moreover, a scenario where the PMCs suitable for a certain application are decided at simulation time is conceivable. Such applications could be, for example, equipment selection as outlined elsewhere in this document.

Obviously the practical implementation of this vision will not be free from difficulties (*e.g.* software licensing, confidentiality issues, *etc.*), but there is little doubt in the potential of these new architectural frameworks.

3.3.5 ACKNOWLEDGEMENTS

Figure 3 & 4 have been extracted from the reference **Error! Bookmark not defined.**. The authors wish to thank its authors and the editor. Also, this paper uses information obtained from the Global CAPE-OPEN project.

3.3.6 REFERENCES

1. Himmelblau D. M. and K. B. Bischoff “Process Analysis and Simulation” J Willey and Sons, New York, (1968).

2. Microsoft Press® Computer and Internet Dictionary © & 1997, 1998 Microsoft Corporation. All rights reserved. Portions, The Microsoft Press® Computer Dictionary, 3rd Edition, Copyright © 1998 by Microsoft Press. All rights reserved.
3. http://www.hyprotech.com
4. http://www.aspentech.com
5. http://www.simsci.com
6. Lapidus L., "Digital Computation for Chemical Engineers" McGraw Hill, New York, (1962).
7. Rosen E.M., "A Machine Computation Method for Performing Material Balances", *Chem. Eng. Prog.* 69 October (1962).
8. Ravicz A. E., R. L. Norman, "Heat and Mass Balancing on a Digital Computer", Chem. Eng. Progr. 60,9, p70 (1964).
9. Gallier, P.W., L.B. Evans, H. I. Britt, J. F. Boston, and P. K. Gupta, "ASPEN: Advanced Capabilities for Modeling and Simulation of Industrial Processes" Computer Applications to Chemical Engineering, ACS Symposium Series N^{o} 124 edited by R.G. Squires and G.V. Reklaitis, 293, (1980).
10. Perkins, J. D. and R.W.H. Sargent, "SPEEDUP: A Computer Program for Steady State and Dynamic Simulation and Design of Chemical Processes", *AIChE Symp. Ser.*, 78 (1982).
11. Marquardt W., P. Holl, D. Butz and E.D. Gilles, " DIVA A flowsheet Oriented Dynamic Process Simulator", *Chem Eng Technol.* **10**, 64 (1987).
12. Barton P.L. and C. Pantelides " Modeling of Combined Discrete/Continuous Processes" *AIChEJ.* **40**, 966 (1994).
13. Allgor, R. , M. Becerra, L. Evans and P. L. Barton, "Optimal Batch Process Development", *Comput. Chem. Eng.*, **20** 885 (1996).
14. Zhu, Y.C. and T. Backx (1993). "Identification of Multivariable Industrial Processes: for Simulation, Diagnosis and Control", Springer-Verlag, London.
15. Evans L. B. , "Advances in Process Flowsheeting Systems", Foundations of Computer-Aided Chemical Process Design, AIChE Meeting , Henniker , New Hampshire July 6-11, II pp 425-469 (1980).
16. Morris, C.G., A. Vysniauskas, W.D. Sim, "An Iterative Approach to Process Simulation", *Chem Eng. Prog.* (1985).
17. Braunschweig, B. L., Pantelides, C.C., Britt, H.I., Sama, S., "Process Modeling: The Promise of Open Software Architectures", *Chem. Eng. Prog.*, **96** (9), pp. 65-76 (Sept. 2000)
18. Sama, S., Cebollero, S., Rodríguez, J.C., "The CAPE-OPEN Standard: Towards the Future in Process Simulation", contribution to Chemputers 9, Budapest (May 99).
19. http://www.w3.org/xml/
20. Walsh, N., "A technical introduction to XML", http://www.xml.com/xml/pub
21. http://www.global-cape-open.org/. CAPE-OPEN and Global CAPE-OPEN are funded by the European Community under the Industrial and Materials Technologies Programme (Brite-EuRam III), under contracts BRPR-CT96-0293 and BPR-CT98-9005. In addition, Global CAPE-OPEN follows the

Intelligent Manufacturing Systems initiative promoting collaboration between six international regions.

22. http://www.microsoft.com/net. See, in particular, http://www.microsoft.com/net/define_net.asp and http://www.microsoft.com/business/articles/net/netvision.asp

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 3.4 : Data Reconciliation Framework

N. Arora, L. T. Biegler & G. Heyen

Data reconciliation and parameter estimation are important components of model fitting, validation, and real time optimization in chemical industries. In its most general form, data reconciliation is a minimization of measurement errors subject to satisfying the constraints of the process model. Parameter estimation is the step after data reconciliation in which the reconciled values of the process variables are used to set values for the model parameters. The most commonly used formulation of both problems is to minimize the sum of squares of the measurement corrections subject to model constraints and bounds. This formulation is based on the assumption that measurements have normally distributed random errors, in which case least squares is the maximum likelihood estimator. However, the data reconciliation problem is compounded when gross errors or biases are present in the data, as these can lead to incorrect estimates and severely biased reconciliation of the other measurements. This paper discusses optimization strategies that deal with data reconciliation and gross error detection. Included in this study are two case studies, a comprehensive steady state example and a small dynamic example. Both illustrate these strategies and issues related to this task.

3.4.1 INTRODUCTION

Efficient and safe plant operation can only be achieved by monitoring key process variables, which contribute to the process economy (*e.g.* yield of an operation) or are linked to equipment quality (fouling in a heat exchanger, activity of a catalyst), safety limits (departure from detonation limit) or environmental considerations (amount of pollutant rejected). Measurements of these variables are needed to monitor process conditions and also to ensure that operating conditions remain within acceptable quality and safety ranges. These measurements are never error free and some reconciliation of the data is a necessary condition for estimating the condition of the plant.

Recent progress in automatic data collection and archiving has raised an awareness of estimation, at least for modern, well-instrumented plants. Operators are now faced with a lot of data, but they have little means to extract and fully exploit the relevant information it contains. Furthermore, most performance parameters are often estimated and not directly measured. As a result, random errors on measurements also propagate in the estimation of performance parameters.

Data reconciliation, also called *validation*, allows state estimation and measurement correction problems to be addressed in a global way. The aim of validation is to remove any error from available measurements, and to yield consistent and complete estimates of all the process state variables as well as unmeasured process parameters. Data reconciliation is based on measurement *redundancy*. This concept is not limited to replicate measurements of the same variable by separate sensors; it includes the concept of spatial redundancy, where a single variable can be estimated in several independent ways, from separate sets of measurements. Moreover, the plant structure yields additional information, which is exploited for correct measurements. Variables describing the state of a process must be reconciled to consistency constraints representing basic laws of nature, such as mass and energy balances and equilibrium constraints. Data reconciliation uses information redundancy and conservation laws to correct measurements and convert them into accurate and reliable knowledge. As a result, the reconciled values exhibit a lower variance compared to original raw measurements; this allows process operation closer to limits (when this results in improved economy).

Current developments in the field aim at combining online data acquisition with data reconciliation. Reconciled data are displayed in control rooms in parallel with raw measurements. Departure between reconciled and measured data can trigger alarms and analysis of time variation of those corrections can draw attention to drifting sensors that need recalibration. Data reconciliation can also be viewed as a *virtual instrument* where key process variables are estimated from variables that are directly measured on-line. Finally, current commercial software aims at easing the development of data reconciliation models as follows: use of libraries of predefined unit operations, automatic generation of equations for typical measurement types, analysis of redundancy and observability, analysis of error distribution of reconciled values, interfaces to on-line data collection systems and archival databases, and the development of specific graphical user interfaces.

The benefits derived from data reconciliation in chemical processes are numerous. They include improvement of measurement layout, fewer routine analyses, reduced frequency of sensor calibration (only faulty sensors need to be calibrated), removal of systematic measurement errors, systematic improvement of process data, a clear picture of plant operating condition, and reduced measurement noise for key variables. Moreover, monitoring through data reconciliation leads to early detection of sensor deviation and equipment performance degradation, actual plant balances for accounting and performance follow-up, safe operation closer to the process limits and improved quality and performance at the process level.

The next section presents the mathematical structure of the data reconciliation problem along with various statistical choices for objective functions. A comprehensive case study that illustrates the use of this structure is provided in Section 3.4.3. The case study also motivates the treatment of gross errors and Section 3.4.4 provides a discussion of this treatment through mathematical programming formulations that involve the Akaike Information Criterion and robust statistics. A small case study on dynamic data reconciliation and gross error detection is provided in Section 3.4.5. Section 3.4.6 then briefly describes the tasks, software components and their interactions for data reconciliation. Finally, Section 3.4.7 summarizes the paper and outlines areas for future work.

3.4.2 MATHEMATICAL STRUCTURE OF DATA RECONCILIATION PROBLEMS

The data reconciliation problem is formulated from data collected at sampling times *i*. If we assume these data sets to be independent of each other, then the data reconciliation and parameter estimation problem can be stated as the following nonlinear programming (NLP) problem:

$$\begin{aligned} &\text{Min } \Sigma_i\, F_i(x^M_i, x_i) \\ &\text{s.t. } h(x_i, u_i, p) = 0 \quad \text{for all i} \\ &x^L \le x_i \le x^U \\ &u^L \le u_i \le u^U \\ &p^L \le p \le p^U \end{aligned} \tag{1}$$

where $F_i(x^M_i, x_i)$ is some objective function that depends on a difference between the measurements and their reconciled values, x^M is the set of measurement data of the corresponding variable x, p is the set of parameters, u is the set of unmeasured variables, h is the set of model equations and the subscript i refers to the ith measurement set. In (1), we assume that all variables are identified with a particular data set and the problem is an *errors in variables measured* (EVM) problem. On the other hand, one can also have multiple measurements of each variable, such as in problems with *moving horizons* [16,22]. The problem is then formulated more generally as:

$$\begin{aligned} &\text{Min } F(x^M, x) \\ &\text{s.t. } h(x, u, p) = 0 \\ &x^L \le x \le x^U \\ &u^L \le u \le u^U \\ &p^L \le p \le p^U \end{aligned} \tag{2}$$

where the symbols mean the same as in (1) but now x and u are concatenated vectors. In this case the reconciled values of variables would lie somewhere in between their successive measurements. This leads to a smaller variance of the reconciled variables and also reduces their sensitivity to any gross error detection tests. Finally, if the model comes from a discretized differential algebraic equation (DAE) system, then time is generally incorporated into the constraints and the objective function. The constraints, discretized successively in time, can be written as in (1) but the equations do not decouple easily. This coupled structure can be a significant issue if the dynamic data reconciliation problem is itself large.

Problems (1) and (2) are usually formed with objective functions derived from maximum likelihood [3]. Here a number of specializations can be made for data reconciliation. In particular, if we assume data snapshots *i* are independent and all data have errors from similar sources, we can simplify the error structure. An objective function in (1) derived from a log-likelihood estimator with a known covariance matrix leads to:

$$F_i(x^M_i, x_i) = \tfrac{1}{2}\,(x^M_i - x_i)^T\, V^{-1}\,(x^M_i - x_i) \tag{3}$$

where V is the covariance matrix, assumed to be the same for all data sets. In addition, if we assume that the elements of each data vector are independent of each other, then (3) simplifies to

$$F_i(x^M_i, x_i) = \tfrac{1}{2}\,\Sigma_j\,((x^M_{ij} - x_{ij})/\sigma_j)^2 = \tfrac{1}{2}\,\Sigma_j\,(\varepsilon_{ij})^2 \tag{4}$$

where σ_j^2 is the variance of element j and ε_{ij} is the studentized residual. However, outliers in the data can strongly bias the reconciled data if least squares forms (3,4) are used. Instead to reduce the effect of these biases we also consider functions based on *M-estimators* (related to maximum likelihood) for the objective function in (1) or (2). Here $\Sigma_i F_i(x^M_i, x_i) = \Sigma_{ij} \rho_{ij}$ is defined by an overall M-estimator and ρ_{ij} is the estimator associated with the each data element. Two particular M-estimators include the *fair function*, a Huber estimator [15], and the *redescending estimator* proposed by Hampel [13]. These are defined as:

Least squares estimator: $\rho^L_{ij} = \frac{1}{2}\varepsilon_{ij}^2$ (5)

Fair function: $\rho^F_{ij} = C^2 \left[|\varepsilon_{ij}|/C - \log\left[1 + |\varepsilon_{ij}|/C\right]\right]$ (6)

Three part redescending estimator:

$$\rho^H_{ij} = \begin{cases} \frac{1}{2}\varepsilon_{ij}^2 & 0 \le |\varepsilon_{ij}| \le a \\ a|\varepsilon_{ij}| - a^2/2 & a < |\varepsilon_{ij}| \le b \\ ab - a^2/2 + \frac{1}{2}(c-b)a[1 - \{(c - |\varepsilon_{ij}|)/(c-b)\}^2], & b < \varepsilon_{ij}| \le c \\ ab + \frac{1}{2}(c-b-a)a & c < |\varepsilon_{ij}| \end{cases} \tag{7}$$

Here least squares is the only non-robust estimator. For the fair function, C is the tuning constant, which has a direct relation to the efficiency of this function. Also, a, b, and c (with c $\geq$ b + 2a) are the tuning constants for the redescending estimator. Figure 1 shows these M-estimators as a function of the studentized residual. Note that the fair function increases only linearly for large residuals and this gives it some robustness compared to the least squares estimator. The redescending estimator becomes constant for large observations; thus large residuals have no influence on the reconciled data. Both the fair function and the redescending estimator approximate the least squares function for small residuals and these estimators have high efficiency for data derived from Gaussian distributions.

Once formulated, problems (1) or (2) can be solved with a number of efficient approaches, as follows:

- Problems (1) or (2) can be solved with any NLP solver. Often SQP is the method of choice as it requires the fewest function evaluations.
- In the absence of variable bounds, Newton's method could be applied directly to the KKT conditions of (1) and (2). Normally this requires first and second derivatives of the objective function and the model equations. With a quadratic objective function and linear constraints, the KKT system is just a set of linear equations. Normally one would not expect variable bounds to be active.
- If the model fits the data well (and gross errors can be eliminated or ignored through M-estimators), the Lagrange multipliers for the equality constraints vanish. As a result, Newton-like behavior is still maintained without requiring second derivatives from the model equations.

More detail on these simplifications can be found in [27, 2, 3].

Finally, before solving the NLP problem, some variable classification and pre-analysis is needed to identify unobservable variables and parameters, and non-redundant measurements. Stanley and Mah [26] and later Crowe [9] proposed observability and redundancy tests for steady state data reconciliation. Albuquerque and Biegler [2] extended these to dynamic systems and applied a sparse LU decomposition rather than a QR factorization. Measured variables can be classified as *redundant* (if the measurement is absent or detected as a gross error, the variable can still be estimated from the model) or *nonredundant.* Likewise, unmeasured variables are classified as *observable* (estimated uniquely from the model) or *unobservable.* The reconciliation algorithm will correct only redundant variables. If some variables are not observable, the program will either request additional measurements (and possibly suggest a feasible set) or solve a smaller sub-problem involving only observable variables. The preliminary analysis should also detect *overspecified variables* (particularly those set to constants) and *trivial redundancy*, where the measured variable does not depend at all upon its measured value but is inferred directly from the model. Finally, it should also identify model equations that do not influence the reconciliation, but are merely used to calculate some unmeasured variables. Such preliminary tests are extremely important, especially when the data reconciliation runs as an automated process. In particular, if some measurements are eliminated as gross errors due to sensor failure, non-redundant measurements can lead to unobservable values and non-unique solutions, rendering the estimates and fitted values useless. As a result, these cases need to be detected in advance through variable classification. Moreover, under these conditions, the NLP may be hard to converge and stronger globalization strategies, such as trust regions [8] need to be applied.

3.4.3 STEADY STATE DATA RECONCILIATION CASE STUDY

There are few data reconciliation case studies of significant size in the literature. The one proposed here is provided so that it can be used to benchmark different solution strategies, and to illustrate what can be achieved by data reconciliation tools. Actual plant data are not used here because they are difficult to obtain and actual processes are much more complex, with details that require more space than available. Instead the process information used here was generated from a steady state simulation model with random noise with zero mean and normal distribution added to the measurement variables. we consider a simplified flowsheet of an ammonia synthesis loop (Figure 2) modeled with the BELSIM VALI III software [5]; this is a typical state of the art commercial application.

As seen in Figure 2, synthesis gas is available as stream 1 at 30 bar pressure. It is brought to synthesis pressure (195 bar) using a 3-stage compressor with intermediate cooling. Unreacted synthesis gas (stream 15) is recycled with the feed of the last compression stage. Compressed reactants are brought to reaction temperature in a product-to-feed heat exchanger E-103. A waste heat boiler E-104 recovers part of the reaction heat to raise steam. To recover ammonia, the reactor effluent is further cooled in the chilling train E-105 (with water) and E-106 (ammonia vaporizer). Condensed ammonia is separated in B-101. The liquid is flashed to moderate pressure in B-102, while a purge (stream 16) allows the inert components (Ar and CH_4) to leave the loop. Physical properties and phase equilibrium calculations were estimated using Soave-Redlich-Kwong model.

Table 1a : Measured and reconciled values for streams

Tag name	Measured	Accuracy	Validated	Accuracy	Penalty	Unit
1_MASSF	85.0	2.0%	82.4	0.8%	2.31	t/h
1_MFAR	0.970	0.05	0.978	0.025	0.02	%
1_MFC1	0.950	0.05	0.980	0.025	0.36	%
1_MFH2	74.70	1	73.66	0.082	1.08	%
1_MFN2	24.10	1	24.38	0.084	0.08	%
1_MFNH3	0.00	CST	0.00			%
1_P	29.9	2.0%	29.9	2.0%	0.00	bar
1_T	39.4	1.00	39.416	0.996	0.00	C
2_P	75.1	2.0%	75.1	2.0%	0.00	bar
2_T	165.0	2.00	164.93	1.98	0.00	C
3_T	39.1	1.00	39.128	0.997	0.00	C
4_P	183.6	2.0%	183.8	2.0%	0.00	bar
4_T	167.9	2.00	167.8	1.98	0.00	C
5_T	40.4	1.00	40.547	0.97	0.02	C
6_P	181.1	2.0%	181.0	1.0%	0.00	bar
6_T	5.9	1.00	5.5	0.35	0.17	C
7_P	188.3	2.0%	190.4	1.1%	0.30	bar
7_T	10.9	1.00	11.1	0.85	0.05	C
8_MFNH3	2.80	0.20	2.89	0.04	0.19	%
8_P	191.2	2.0%	189.9	1.2%	0.11	bar
8_T	329.3	2.00	327.8	1.59	0.56	C
9_MFNH3	14.60	0.20	14.61	0.124	0.00	%
9_P	195.6	2.0%	191.6	1.2%	1.02	bar
9_T	502.8	3.00	505	1.95	0.54	C
10_P	186.4	2.0%	191.3	1.2%	1.71	bar
10_T	408.4	3.00	409.5	1.82	0.14	C
11_P	192.4	2.0%	191.3	1.2%	0.09	bar
11_T	84.4	2.00	82.7	1.48	0.68	C
12_T	39.1	1.00	39.1	1.00	0.00	C
13_T	-5.8	1.00	-5.2	0.37	0.31	C
14_P	182.0	2.0%	181.0	1.0%	0.08	bar
14_T	-3.9	1.00	-5.45	0.36	2.41	C
15_MASSF	293.1	2.0%	294.7	1.5%	0.07	t/h
16_MASSF	18.40	3.0%	18.51	2.5%	0.04	t/h
16_MFAR	5.20	3.0%	5.03	2.0%	1.15	%
16_MFC1	4.80	3.0%	4.86	2.1%	0.19	%
16_MFH2	66.7	1.00	65.38	0.45	1.74	%
16_MFN2	22.3	1.00	20.87	0.42	2.05	%
16_MFNH3	3.90	3.0%	3.85	1.3%	0.15	%
16_P	181.9	2.0%	181.0	1.0%	0.06	bar
16_T	-6.9	1.00	-5.45	0.36	2.10	C
18_MASSF	0.80	5.0%	0.83	2.1%	0.56	t/h
18_MFAR	4.90	3.0%	5.01	2.0%	0.54	%
18_MFC1	9.40	3.0%	9.15	1.9%	0.76	%
18_MFH2	32.4	1.00	33.61	0.42	1.47	%
18_MFN2	11.60	3.0%	11.64	2.1%	0.01	%
18_MFNH3	40.4	1.00	40.59	0.52	0.04	%
18_P	10.1	2.0%	10.1	1.6%	0.05	bar
19_MASSF	62.7	1.0%	63.1	0.8%	0.36	t/h
CW1_P	3.0	2.0%	3.0	2.0%	0.00	bar
CW1_T	20.2	1.0	20.2	1.0	0.00	C
CW2_P	3.0	2.0%	3.0	2.0%	0.00	bar
CW2_T	20.7	1.0	20.7	1.0	0.00	C
CW3_P	2.9	2.0%	2.9	2.0%	0.00	bar
CW3_T	22.8	1.0	22.8	1.0	0.00	C
CWR1_T	30.8	1.0	30.8	1.0	0.00	C
CWR2_T	30.2	1.0	30.2	1.0	0.00	C
CWR3_T	31.4	1.0	31.4	1.0	0.00	C
AM1_P	2.0	2.0%	2.0	2.0%	0.00	bar
AM1_T	-19.1	1.00	-19.1	1.00	0.00	C
AMR1_T	-17.2	1.00	-17.2	1.00	0.00	C
BFW_MASSF	40.8	2.0%	40.3	1.5%	0.32	t/h
BFW_P	40.4	1.0%	40.3	1.0%	0.02	bar
BFW_T	24.3	1.00	24.3	1.00	0.00	C
STM_P	39.3	2.0%	39.5	1.6%	0.08	bar
STM_T	247.9	2.00	247.8	2.00	0.00	C
WCOMP POWER	21471	5.0%	21745	1.8%	0.07	kW

Table 1b : Measured and reconciled values for unit parameters

Tag name	Measured	Accuracy	Validated	Accuracy	Penalty	Unit
C-101EFFIC	0.75	0.15	0.756	0.03	0.00	-
C-102EFFIC	0.75	0.15	0.721	0.03	0.04	-
C-103EFFIC	0.75	0.15	0.728	0.13	0.02	-
B-102DP	170	100	170.9	1.74	0.00	bar
R-101DTEQ1	0.0	100	0.28	3.08	0.00	K
E-101A	500	CST	500			m2
E-101DP1	0.5	1.00	0.5	1.00	0.00	bar
E-101DP2	0.1	1.00	0.114	1.00	0.00	bar
E-101U	0.35	1.00	0.325	0.01	0.00	kW/m2/K
E-102A	500	CST	500			m2
E-102DP1	0.5	1.00	0.5	1.00	0.00	bar
E-102DP2	0.1	1.00	0.094	1.00	0.00	bar
E-102U	0.35	1.00	0.323	0.01	0.00	kW/m2/K
E-103A	3000	CST	3000			m2
E-103DP1	0.5	1.00	0.433	0.98	0.00	bar
E-103DP2	0.1	1.00	0.025	0.98	0.01	bar
E-103U	0.35	1.00	0.437	0.01	0.01	kW/m2/K
E-104A	200	CST	200			m2
E-104DP1	0.5	1.00	0.825	0.66	0.11	bar
E-104DP2	0.1	1.00	0.370	0.98	0.07	bar
E-104U	0.35	1.00	0.476	0.01	0.02	kW/m2/K
E-105A	2500	CST	2500			m2
E-105DP1	0.5	1.00	0.5	1.00	0.00	bar
E-105DP2	0.1	1.00	0.110	1.00	0.00	bar
E-105U	0.35	1.00	0.194	0.02	0.02	kW/m2/K
E-106A	1500	CST	1500			m2
E-106DP1	0.5	1.00	0.5	1.00	0.00	bar
E-106DP2	0.1	1.00	0.110	1.00	0.00	bar

The problem is formulated by specifying two models: the process model and the measurement model. Measurements, x^M, are identified by a tag name and data for each measurement includes the measured value, the expected standard deviation, σ (either a constant value or a percentage of the measurement), and a measurement type. Lower and upper bounds may also be provided for all measured variables. Further, nonstandard measurements can be defined by adding the corresponding linking equations to the process model. Process parameters, p, such as heat transfer coefficients or compressor efficiencies also fit easily in such a framework. They can be bounded appropriately to enforce some physical constraints or assigned a constant value. This is the case for some equipment parameters (*e.g.* area of a heat exchanger) or variables whose value must remain fixed (*e.g.* nitrogen content in atmospheric air, or absence of reaction product in the feed). Measured values in this study, their standard deviations and validated results are displayed in Tables 1a and 1b.

For the case examined here, the objective function is the weighted sum of squares of measurement corrections and the program generates 123 model equations. These include mass and energy balances, pressure drop equations, vapor-liquid equilibrium constraints, and definitions for isentropic compressor efficiency and heat transfer coefficients and linking equations to relate the measured quantities (*e.g.* mole fractions or mass flowrate) to the state variables (*e.g.* partial molar flowrates). The reconciliation problem involves 89 measurements and 84 unmeasured state variables, 10 variables set as constants (e.g. area for all heat exchangers), and thus 40 redundancies. Solution of this NLP can be obtained in a few seconds on a personal computer. Two different solution algorithms have been tested and perform equally well on the proposed problem. The first one solves the KKT conditions of (2) as a set of algebraic equations using a dogleg method. In the absence of bound constraints, this method is the fastest for large well-behaved problems. As an alternative, the NLP was also solved with a large scale SQP solver, which is somewhat slower, more robust, and can handle bounds on the variables.

At the solution the weighted least squares objective function has a value of 24.5035 and the measurements appear to be acceptable; none is corrected by more than twice the assumed standard deviation, as shown in the results (Tables 1a and 1b). After convergence is attained, a global Chi-square test is performed (with a value of 54.58) which confirms that measurement corrections stay within the acceptance range of the assumed error distribution. The equality constraints are satisfied to within a tolerance of 10^{-6} and no variable lies on a bound, although eleven are near their bounds. Table 2b also shows that the available measurement set allows process parameters to be identified with acceptable accuracy. For instance, the isentropic efficiency of compressor C101 was guessed at 0.75 with a standard deviation of 0.15. The new estimate is 0.756 with a standard deviation of 0.03.

In addition, a *sensitivity analysis* provides measured and reconciled values of each measurement as well as its specified standard deviation and *a posteriori* variance of the reconciled value. These are reported in relative (relative accuracy) and absolute (absolute accuracy) terms below. Also, state variables depending on a given measurement are listed, along with the weight factor (contribution) indicating the contribution of the measurement variance to the variance of the reconciled value. For instance, for the efficiency of C101 compressor, the sensitivity report indicates:

Measurement		Tag Name	Value	Absolute accuracy	Relative accuracy	Unit
EFFIC	U C-101	Reconciled	0.75573	0.28988E-01	3.84%	-
		C-101EFFIC	0.75000	0.15000	20.00%	-

Variable		Tag Name	Contribution	d/d meas	d/d sigma	Unit
P	S 1	1P	36.75%	-0.29387E-01	0.72%	bar
P	S 2	2P	36.31%	0.11630E-01	1.36%	bar
T	S 2	2T	14.99%	-0.56122E-02	1.17%	C
T	S 1	1T	7.26%	0.78132E-02	0.35%	C
EFFIC	U C-101	C-101EFFIC	3.73%	0.37347E-01	80.67%	-

Note that the uncertainty on the C-101 efficiency can be decreased mainly by improving the accuracy of pressure measurements for streams 1 and 2, and, to a lower extent, by improving temperature measurements. The initial guess of efficiency has little impact on the final value: the derivative of the result with respect to the guess is 0.037. This sensitivity analysis detects the importance of all measurements on the identification of process states and parameters. Some measurements might appear to have little effect on the result, and might thus be discarded from analysis. For instance, adding a measurement of ammonia concentration in stream 19 is almost useless; the process is designed so that stream 19 is almost pure ammonia and its purity is fixed mainly by the flash pressure and temperature.

The data reconciliation program can also identify the influence of measurements on reconciled state variable values. As an example, consider the reactor productivity, where one might expect the reactant flow rate and ammonia fractions in the reactor inlet and outlet streams to be the most influential measurements. Instead, the sensitivity report below shows that the major source of uncertainty is due to the flowrate of the pure ammonia product (stream 19). Process feed (stream 1) also contributes to a lower extent, as well as the boiler feed water flowrate, indicating that the energy balance of the synthesis loop also provides information about the reaction rate.

Variable		Tag Name	Value	Absolute accuracy	Relative accuracy	Unit
EXTENT1	U R-101	Computed	1896.2	15.161	0.80%	kmol/h

Measurement		Tag Name	Contribution	d/d meas	d/d sigma	Unit
MASSF	R 19	19MASSF	66.42%	19.707	18.30%	t/h
MASSF	R 1	1MASSF	9.88%	2.8031	59.59%	t/h
MASSF	R BFW	BFWMASSF	4.92%	4.1220	25.42%	t/h
MASSF	S 15	15MASSF	4.66%	0.55857	24.65%	t/h
T	S 11	11T	2.75%	1.2560	26.07%	C
MASSF	R 18	18MASSF	1.73%	49.897	57.38%	t/h
MFN2	R 18	18MFN2	1.34%	-5.0349	30.02%	%

These examples show that some measurements can have a very high impact on the validated variables and on their variance. These measurements should be carried out with special caution, and it may prove wise to duplicate the sensors. More examples of such analysis are discussed in Heyen *et al* [14].

Finally, we examine how data reconciliation can be used as a fault detection framework. Here we generate another data set that assumes an internal leak in the heat exchanger E-103. A fraction (10%) of stream 7 is directly mixed with stream 11 and part of the flow bypasses the reactor section and cycles in the compressor and the cooling section. This is obviously a waste of energy. Solving the data reconciliation problem leads to an objective function of 42.8857 and a Chi-square of 55.76. Compared to the previous case, the objective function is definitely larger, although the Chi-square test is still satisfied. On the other hand, one measurement (flow rate of stream 15) is corrected by more than 3 standard deviations, and is flagged as suspect. Other measurements in the reaction loop (streams 6, 9, 16, 18, 19) are also corrected by more than one standard deviation.

Here, one may suspect that something is wrong, but there is no direct evidence about the cause. Instead, we modify the flowsheet and the reconciliation model to assume that an unknown fraction of stream 7 bypasses E-103 and is transferred directly to stream 11. Rerunning the data reconciliation problem with a very small initial guess (10^{-4}) for the bypass fraction, leads to an objective function value of 35.721 and a Chi-square value of 55.76. The objective function is now reduced, and the largest variable correction is less than 2 σ. The bypass fraction is estimated as being approximately 0.06, with a standard deviation of 0.021.

Variable		Tag Name	Value	Absolute accuracy	Relative accuracy	Penalty	Unit
FRAC2	U S-1	Reconciled	0.60335E-01	0.21536E-01	35.69%		-
		S-1FRAC2	0.10000E-03	0.50000	******%	0.01	-

Measurement		Tag Name	Contribution	d/d meas	Rel.Gain	Penalty	Unit
MASSF	S 15	15MASSF	34.90%	0.12808E-02	16.94%	2.92	t/h
MFNH3	R 9	9MFNH3	18.91%	0.46826E-01	20.48%	0.10	%
T	S 11	11T	14.22%	-0.40610E-02	18.61%	3.95	C
MASSF	R BFW	BFWMASSF	6.00%	-0.67275E-02	20.10%	2.03	t/h
T	S 8	8T	5.06%	-0.24211E-02	16.45%	0.19	C
...							

As shown above, the sensitivity report for this parameter indicates that better estimation of the bypass fraction requires improved accuracy (or additional measurements) of the flow rate for stream 15 and, to a lesser extent, the ammonia concentration in stream 9 and the temperature of stream 11. As a final check, we consider the bypass model with our original (leak-proof) data. Here we obtain the same objective function value (24.5035) as for the base case and the estimated leak fraction is correctly estimated as being zero.

3.4.4. OPTIMIZATION STRATEGIES FOR GROSS ERROR DETECTION

In the previous section the comprehensive case study indicates the usefulness of data reconciliation for obtaining accurate states, assessing the sensitivity of measurements and their uncertainties on estimated parameters, and in providing a tool for fault detection. In this section we focus in more detail on efficient strategies for handling gross errors in data reconciliation.

As seen in Section 3.4.2, the least squares objective function can be severely biased leading to incorrect reconciliation and estimation. As indicated in the case study a common procedure identifies the measurements that suffer from gross errors and eliminates them in a sequential procedure. Early papers on the subject describe tests based on Chi-square statistics as the criteria for identifying outliers [11]. In addition, Crowe *et al.* [12] used matrix projection to reconcile process flows. They devised a Chi-square test based on the inverse of the reduced Hessian. Madron [18, 19] proposed a Chi-square test based on the squared studentized residuals following a non-central Chi-square distribution. He also provided methods for gross error detection and the concept of measurement credibility. Kao *et al.* [17] proposed a Chi-square test for gross error detection in serially correlated process data. They also compared this with three other tests for outlier detection.

Serth and Heenan [24] devised a combinatorial strategy called the *screened combinatorial method* which uses standard normal deviates calculated for each measured stream; they reduce the size of the combinatorial problem by identifying the smallest complete subset of biased measurements. Narasimhan and Mah [20, 21] tested their *generalized likelihood ratio* method against the results of Serth and Heenan [24] and reported that their method was able to identify a wide variety of gross errors. This is also a combinatorial strategy that uses the normal distribution for outlier identification. Crowe [10] devised a maximum power test by which he tested constraint residuals for normality and identified suspect balances. More recently, Rollins *et al.* [23] derived a linear combination technique which is a combinatorial technique based on the chi-square test that identifies equivalent gross errors. Bagajewicz and Jiang [4] performed gross error detection on models characterized by differential equations by fitting polynomials to the differential variables and solving the resulting linear system. They used concepts from graph theory for their combinatorial strategy. However, all of the above methods were derived only for linearly constrained problems and are multi-step processes that use a sequential scheme for reconciliation and gross error detection.

On the other hand, the estimators derived from robust statistics can be used as objective functions in data reconciliation problems. These estimators put less weight on large residuals corresponding to outliers. As shown in Section 3.4.2, this leads to less biased parameter estimates and reconciled values. Tjoa [27] used the *contaminated normal distribution* to derive the objective function and used this to simultaneously identify outliers. Albuquerque and Biegler [2] used the *fair function* (6), to reduce the effects of gross errors. Finally, Arora and Biegler [3] used the redescending M-estimator (7). Here a key advantage for outlier detection is simultaneous data reconciliation and gross error detection, and an elimination of the combinatorial procedure.

The presence of so many different methods for data reconciliation raises the question of whether there is a common statistical framework in which both combinatorial and robust methods can be interpreted. Here we employ the *Akaike Information Criterion* (AIC) to provide a general framework for data reconciliation. Yamamura *et al.* [28] applied the AIC to identify biased measurements in a least squares framework for gross error detection. Due to the combinatorial nature of the problem attempted, they suggested a *branch and bound* method to solve the problem. This approach can be automated by mixed integer programming techniques (see *e.g.* [25,3]) However, this can be computationally expensive, as it requires a discrete decision for each measurement and problems become even more difficult when the model is nonlinear. Nevertheless, in the remainder of this section we will explore the AIC framework for simultaneous gross error detection and data reconciliation. In particular, we will derive a mixed integer approach as well as a robust approach where the constants in the M-estimator are tuned using AIC.

3.4.4.1 Data Reconciliation with the Akaike Information Criterion

Data reconciliation and gross error detection can be addressed as a model discrimination and parameter estimation problem, where multiple models correspond to the partitioning of random and gross errors. If more than one of these models can be fitted to the data under consideration, it becomes necessary to identify which model to use. To this end, one is interested in obtaining the most likely model (which gross errors are identified) and its parameters. Since maximum likelihood estimators are asymptotically efficient under certain conditions [1], the likelihood function is a very sensitive criterion of deviation of model parameters from their true values. For data reconciliation, the Akaike Information Criterion (AIC) can be written as:

$$\mathrm{AIC} = \Sigma_{ij}\, \varepsilon_{ij}^2 + 2\,\dim(q) \tag{8}$$

where ε_{ij} is the studentized measurement error obtained after reconciliation, and dim(q) is the number of independently adjusted model parameters given by: $\dim(q) = \dim(p) + n_{out}$, where dim(p) is the number of model parameters and n_{out} is the number of outliers. Here variables with outlying measurements are treated as parameters because their reconciled values are determined only from measurements without gross errors.

3.4.4.2 Mixed Integer Approaches for Data Reconciliation

The AIC estimate includes both continuous and discrete variables and in the context of data reconciliation, each potential gross error is represented by a binary (0-1) variable. The resulting problem can be translated into a mixed integer non linear program (MINLP) with binary variables identifying faulty sensors as follows:

$$\begin{gathered}\mathrm{Min}\ \Sigma_{ij}\, (\varepsilon_{ij} + \mu_{ij})^2 + 2\,\Sigma_{ij}\, y_{ij} \\ \text{s.t. } h(x, p) = 0,\ \varepsilon_{ij} = (x^M_{ij} - x_{ij})/\sigma_j \\ L\, y_{ij} \le |\mu_{ij}| \le U\, y_{ij} \\ y_{ij}\, \varepsilon\ [0, 1],\ x \ge 0\end{gathered} \tag{9}$$

where y_{ij} is a binary variable denoting existence of bias in element j of data vector i, , μ_{ij} is the magnitude of bias in this variable, and L and U are lower and upper bounds on the bias variable. If the model equations are linear, they can be replaced by $Ax = 0$, where A is the coefficient matrix of linear balance constraints. Due to the presence of the absolute value operator in the bound constraints of (9), this problem is reformulated by adding additional binary and nonnegative continuous variables. In addition, mixed integer linear programming (MILP) approaches were derived [25, 3] where the quadratic term in (9) is replaced by a linear objective function related to AIC. Here, the advantage of an MILP is that it eliminates the nonlinear programming subproblem associated with MINLPs that have only quadratic objectives. Both MINLP and MILP formulations were compared [3] on a steam metering example solved in a moving horizon framework (3).

However, mixed integer formulations can be expensive to solve on-line, particularly for (EVM) problems (2). As an alternative, one can also consider the AIC in the context of robust M-estimators. Here the tuning parameters of the M-estimators (6, 7) can be chosen for the NLP formulation to reduce the AIC. Although this criterion is not minimized directly as in (9), the robust approach has the advantage of solving only continuous variable optimization problems. In the next section we present a small case study, with a dynamic data reconciliation model, that illustrates the use of M-estimators with AIC tuning.

5. DYNAMIC DATA RECONCILIATION CASE STUDY

In this section we compare data reconciliation strategies derived from M-estimators on a small dynamic process problem. This problem has been taken from Albuquerque [2] and deals with two stirred tanks connected by a valve; liquid flows into the first tank, from the first tank into the second tank, and out of the second tank. The three flows, F_0, F_1, and F_2, and the heights L_1 and L_2 are measured at 12 second intervals. The areas of the tanks, A_1 and A_2 are estimated as unknown parameters. The process is shown in Figure 3 and the system is described by the following index-2 DAE system (10):

$$\begin{aligned} A_1\, dL_1/dt &= F_0 - F_1 \\ A_2\, dL_2/dt &= F_1 - F_2 \\ F_2 - A_2\,(2g\,L_2)^{1/2} &= 0 \\ L_1 - L_2 &= 0 \end{aligned} \tag{10}$$

Measurement data is simulated as in [2]. All of the flowrates and levels are measured; they have independent Gaussian distributions, with a variance of 0.01. In addition, a large number of gross errors are introduced into the system by assuming that sensors for F_2 and L_1 fail at their lower bounds at the second time instant and remain there. The DAE system is discretized by an *implicit Euler scheme* to form the set of model equations in (1); measurements are taken at each interval of discretization. Thus this problem is an *EVM* problem with coupling in successive constraints for each set of measurements. Also, because of the large amount of data and gross errors, the problem cannot be considered with the MINLP formulation in (9).

All measurements except F_0 are redundant and the parameters $1/A_1$ and $1/A_2$ are observable and are bounded between 0.25 and 0.55. Here the resulting data reconciliation problem is solved with the least squares estimator, fair functions (6) at 95%, 80%, and 70% asymptotic efficiencies (see [3]), and the redescending estimator (7) with an AIC tuning. All cases were solved with the reduced Hessian Successive Quadratic Programming (rSQP) algorithm [6]. Results for estimation of $1/A_1$ and $1/A_2$ have been summarized in Table 2. Note that the least squares estimator performs very poorly because the estimates of both parameters go to their lower and upper bounds respectively. This behavior is expected because the least squares estimator is heavily biased by the gross errors; relaxing the parameter bounds would lead to even worse estimates. The fair function at various tuning levels performs somewhat better but still the estimate of $1/A_1$ converges to the lower bound; the estimate of $1/A_2$ becomes better as the fair function becomes more robust. In contrast, the redescending estimators perform much better. For tuning parameters given in the experiments M_2 to M_6, the estimates of $1/A_1$ and $1/A_2$ are very close to their true values. For the redescending M-estimator, Table 2 also lists the number of outliers along with the AIC. From Table 2, the scaled AIC objective indicates that the best estimator lies close to M_5. Upon tuning the parameters in (7) to minimize the AIC, we find this to be indeed the case. Here the estimates of $1/A_1$ and $1/A_2$ are almost at their true values.

Figures 4 and 5 show the fitted values of L_1 and F_2 when estimation is performed by the least squares estimator, the fair function (efficiency=70%), and the tuned redescending estimator. In addition, we have also plotted the original noisy data (with a variance of 0.01) and the gross errors. Here the poor performance of least squares is clearly observed, as its fitted values have been grossly underestimated. The fair function underestimates L_1 but fits F_2 quite closely. This is due to the effect of $1/A_1$ at its lower bound. Finally, in both plots the tuned redescending estimator ignores the noise and fits L_1 and F_2 at very close to their true values.

Table 2: Dynamic Data Reconciliation Results

Experiments	$1/A_1$	$1/A_1$	n_{out}	AIC
True Values	0.500	0.500		
Least Squares	0.25*	0.55*		
Fair Function (Eff. = 95%)	0.25*	0.320		
Fair Function (Eff. = 80%)	0.25*	0.388		
Fair Function (Eff. = 70%)	0.25*	0.439		
M_1 (a=8, b=16, c=32)	0.308	0.391	48	24.855
M_2 (a=4, b=8, c=16)	0.454	0.485	49	36.647
M_3 (a=2, b=4, c=8)	0.484	0.498	95	1.409
M_4 (a=2, b=2, c=4)	0.485	0.499	95	1.078
M_5 (a=0.5, b=1, c=2)	0.498	0.499	103	1.048
M_6 (a=0.1, b=0.2, c=0.4)	0.492	0.495	163	1.303
Iterative Tuning (a=0.362, b=0.724, c=1.449)	0.502	0.499	105	0.974

3.4.6. SOFTWARE ARCHITECTURE

So far, we have considered the formulation, solution algorithms and analysis of data reconciliation problems for both steady state and dynamic processes. In this section we consider the software components and their interactions needed to perform these activities for data reconciliation. For these data reconciliation studies, several tasks need to be performed, each requiring access to one or several software components, as shown in Figure 6.

1. A process model has to be defined. This can be done either by using a graphical user interface to assemble process building block (heat exchangers, mixers, splitters, reactors, *etc*) as shown in Figure 2, or by explicitly defining all the constraints equations (mass and energy balance, constitutive equations, *etc*). This definition phase is similar to the set up phase of a simulation problem. Reference can be made to a library of custom models (*e.g.* unit operations, physical properties) and to a data bank of model parameters (*e.g.* pure component and mixture parameters).
2. The process description and its parameters must be stored in an appropriate data structure, usually a process database.
3. A measurement model has to be defined; this requires the declaration and description of all available measurements : a name, an identification (*e.g.* the tag name of the sensor in the DCS system), the default value, the error distribution (*e.g.* standard deviation if a Gaussian error distribution is assumed), possibly rules used for gross error detection (in the simplest case, lower and upper bounds for the measured value, outside which the measured value will be flagged as erroneous and discarded). When the measured value is not a state variable, link equations must be added to relate it to the process state variables. Access to a database of sensor properties can also provide some help.
4. The measurement model must be stored in an appropriate data structure that can be part of the process database or kept separate. Measurements can be obtained in real time from the DCS interface.
5. The first step in the solution of the data reconciliation problem is to detect and isolate gross errors, to verify the redundancy, and to classify variables. This problem analysis phase relies on the analysis of the incidence matrix of the equation system. A library of modules allowing a graph analysis and/or equation and variable ordering is needed. Feedback is provided to the user, in the form of advice on additional measurements to add in case the problem is underspecified.
6. The problem solution makes use of some solver: it can be a linear or non-linear equation solver, or an optimizer (typically large-scale SQP). In the solution phase, residuals of the energy balance equations and equilibrium constraints must be evaluated, which requires access to a physical property package. Feedback is provided to the user, in case of any convergence problem, or with a display of reconciled values through the GUI and the process database.
7. Reconciled values can also be transferred back to the DCS for archival, or for use in the supervisory control system or the on-line optimizer.
8. Sensitivity analysis, statistical inference and assessment of the accuracy of the reconciled values can be obtained by post processing the Jacobian matrix of the constraint equation system. This requires access to linear algebra and statistical packages. Feedback is provided to the user, in the form of advice on measurements to add or to improve in order to reduce the uncertainty on key process parameters.

Data reconciliation can be run under control of a user, who enters measurement values in the measurement database or modifies the model in order to carry out a performance study or to design a measurement system. It can also run under the control of a real time executive, to process routinely all measurements gathered by the DCS, and to provide reconciled values for the real time optimizer. This is why a typical data reconciliation software is composed of several applications: graphical user interface, database systems, expert systems and a run time solver.

3.4.7. CONCLUSIONS

Data reconciliation and parameter estimation is a widely used technology that has become an important component in model fitting, validation, and real time optimization for chemical processes. Most of the applications are for processes operated at steady state, and for which a model based on steady state conservation laws is adequate. However, careful attention must be paid to the optimization formulation. Here commonly used data reconciliation formulations that use least squares objective functions can lead to severely biased reconciliation of the state variables in the process. To deal with these issues, this paper discusses optimization strategies for data reconciliation and gross error detection. Moreover, their application is illustrated with a comprehensive steady state case study and a small dynamic example.

Despite advances in the development and application of data reconciliation strategies, there are a number of ongoing developments in several areas. First, most implementations of data reconciliation assume that measurement errors follow a Gaussian distribution with zero mean and known covariance (usually diagonal covariance is assumed); the implications of those assumptions and the benefit expected from more complex hypothesis still deserves some attention. Preliminary work along these lines includes the use of Bayesian and M-estimator formulations which allow for more general distributions as well as robustness to deviations in the assumed error distribution [27, 3].

Second, model based data reconciliation ignores any model uncertainty. However some constraint equations (e.g. energy balances) involve empirical relationships, such as the physical property models, that are not totally accurate; a data validation framework accounting for model uncertainty is still to be developed. Some interesting work along these lines relates to model discrimination. Although applied to off-line problems, Bayesian approaches can provide additional information on the suitability of competing process models. In addition, model discrimination algorithms, that detect failing sensors (in order to ignore them) and distinguish between process upsets and perturbed measurements, need to be improved.

Third, data reconciliation is useful for process monitoring, but it could be used also for the rational design of measurement systems. This includes questions on where to locate sensors, when to make them redundant, how to minimize measurement cost for a prescribed accuracy, or how to maximize accuracy for a given measurement cost.

Fourth, on-line data reconciliation based on steady state models involves the solution of a large NLP or a large set of nonlinear equations. Here a robust solution is usually obtained by using the previous solution as the initial guess for the next problem. However, this strategy can fail when the process structure or the measurement set changes (*e.g.*, shutdown or start up of a unit, failing sensor, measurement result available only after a long delay). Moreover, the presence of gross errors can lead to nonunique solutions and convergence failure. More efficient and robust nonlinear programming strategies are needed for these situations [8].

Finally, the on-line application of dynamic data reconciliation (during process transients, or for batch processes) is only in its infancy. A number of challenges include an increase in model complexity as well as the optimization of large-scale differential-algebraic systems. In addition, the choice of a suitable horizon for the observations that are taken into account in the analysis has to be a compromise between precision and computation time. One way to explore this trade-off is the application and refinement of wavelet approximations over time [7].

3.4.8 REFERENCES

[1] Hirotugu Akaike. A New Look at the Statistical Model Identification , IEEE Transactions on Automatic Control , AC-19(6):716, 1974.

[2] Joao S. Albuquerque and Lorenz T. Biegler. Data Reconciliation and Gross-Error Detection for Dynamic Systems. AIChE J. , 42(10):2841, 1996.

[3] Nikhil Arora and Lorenz T. Biegler, “Redescending Estimators for Data Reconciliation and Parameter Estimation,” Computers and Chemical Engineering, accepted for publication, 2001.

[4] Miguel J. Bagajewicz and Qiyou Jiang. Gross Error Modeling and Detection in Plant Linear Dynamic Reconciliation . Computers Chem. Engng., 22(12):1789, 1998.

[5] Belsim s.a. Vali III Users's Guide, version 9.05, Belsim, Rue G. Berotte 29a, B Saint-Georges-sur-Meuse (Belgium), 2001

[6] Lorenz T. Biegler, Jorge Nocedal, and Claudia Schmid. A Reduced Hessian Method for Large-Scale Constrained Optimization . SIAM J. Optimization , 5(2):314, 1995.

[7] T. Binder, L. Blank, W. Dahmen and W. Marquardt, "On the Regularization of Dynamic Data Reconciliation Problems," Journal of Process Control, to appear (2001)

[8] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal. An Interior Point Algorithm for Large Scale Nonlinear Programming . Technical report, Optimization Technology Center, Northwestern University, 1997.

[9] Cameron M. Crowe. Observability and Redundancy of Process Data for Steady State Reconciliation . Chemical Engineering Science , 44(12):2909, 1989.

[10] Cameron M. Crowe. Test of Maximum Power for Detection of Gross Errors in Process Constraints . AIChE J. , 35(5):869, 1989.

[11] C. M. Crowe, Data reconciliation - Progress and challenges, J. Proc. Cont., (6) 89-98, (1996)

[12] C.M. Crowe, Y.A. Garcia Campos, and A. Hrymak. Reconciliation of Process Flow Rates by Matrix Projection . AIChE J. , 29(6):881, 1983.

[13] Frank R. Hampel. The Influence Curve and its Role in Robust Estimation . Journal of the American Statistical Association , 69(346):383, Jun., 1974.

[14] G. Heyen, E. Maréchal, B. Kalitventzeff, Sensitivity calculations and variance analysis in plant measurement reconciliation, Computers and Chemical Engineering, vol. 20S, pp 539-544 (1996).

[15] Peter J. Huber, Robust Statistics, John Wiley and Sons, New York, 1981.

[16] S.S. Jang, B. Joseph, and H. Mukai, Comparison of two approaches to on-line parameter and state estimation of nonlinear systems, Ind. Eng. Chem. Process Des. Dev., 25:809, 1986.

[17] Chen-Shan Kao, Ajit C. Tamhane, and Richard S.H. Mah, Gross Error Detection in serially Correlated Process Data. 2. Dynamic Systems, Ind. Eng. Chem. Res. , 31:254, 1992

[18] F. Madron, A New Approach to the Identification of Gross Errors in Chemical Engineering Measurements, Chemical Engineering Science , 40(10):1855, 1985

[19] Madron, F., Process plant performance : measurement and data processing for optimization and retrofits, Ellis Horwood, London (1992)

[20] S. Narasimhan, C. Jordache, " Data Reconciliation & Gross Error Detection ", Gulf Publishing Company (2000)

[21] S. Narasimhan and R.S.H. Mah, Generalized Likelihood Ratio Method for Gross Error Identification, AIChE J. , 33(9):1514, 1987.

[22] Douglas G. Robertson, Jay H. Lee, and James B. Rawlings, A Moving Horizon-Based Approach for Least-Squares Estimation, AIChE J., 42(8):2209, 1996.

[23] Derrick K. Rollins, Yisun Cheng, and Sriram Devanathan, Intelligent Selection of Hypothesis Tests to Enhance Gross Error Identification, Computers Chem. Engng , 20(5):517, 1996.

[24] R.W. Serth and W.A. Heenan, Gross Error Detection and Data Reconciliation in Stream-Metering Systems, AIChE J. , 32(5):733, 1986.

[25] Tyler A. Soderstrom, David M. Himmelblau, and Thomas F. Edgar, A Mixed Integer Optimization Approach for Simultaneous Data Reconciliation and Identification of Measurement Bias, Control Engineering Practice, to appear, 2001

[26] G.M. Stanley and R.S.H. Mah, Observability and Redundancy in Process Data Estimation, Chem. Eng. Sci. , 36:259, 1981.

[27] I.B.F. Tjoa, Simultaneous Solution and Optimization Strategies for Data Analysis, PhD thesis, Carnegie Mellon University, 1991.

[28] K. Yamamura, M. Nakajima, and H. Matsuyama. Detection of gross errors in process data using mass and energy balances, International Chemical Engineering, 28(1):91, 1988

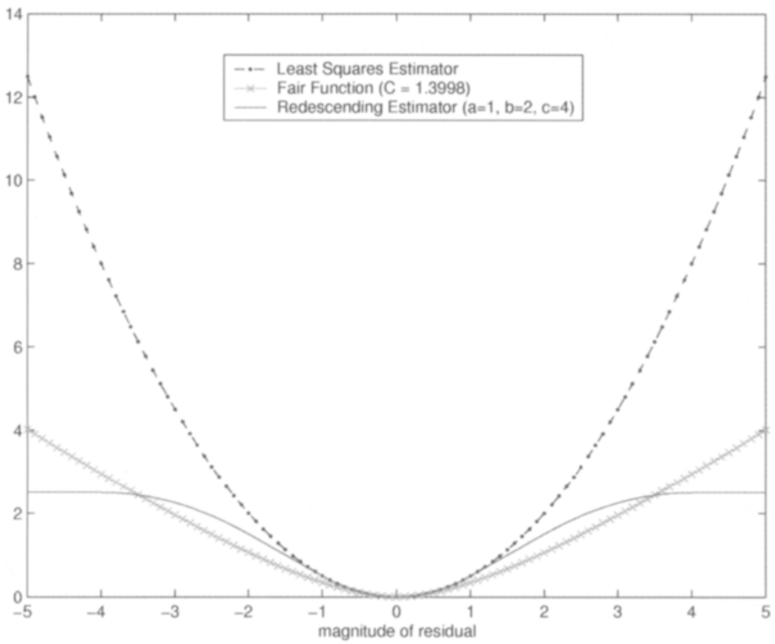
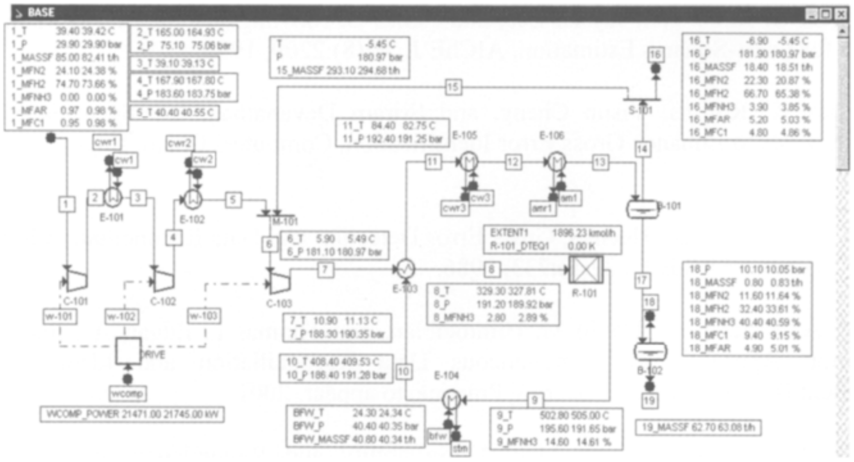


Figure 1: Comparison of M-estimators



```
1_T        39.40  39.42 C
1_P        29.90  29.90 bar
1_MASSF    85.00  82.41 t/h
1_MFN2     24.10  24.38 %
1_MFH2     74.70  73.66 %
1_MFNH3     0.00   0.00 %
1_MFAR      0.97   0.98 %
1_MFC1      0.95   0.98 %

2_T 165.00 164.93 C
2_P  75.10  75.06 bar

3_T 39.10 39.13 C

4_T 167.90 167.80 C
4_P 183.60 183.75 bar

5_T 40.40 40.55 C

T                    -5.45 C
P                   180.97 bar
15_MASSF 293.10 294.68 t/h

11_T   84.40  82.75 C
11_P 192.40 191.25 bar

16_T       -6.90  -5.45 C
16_P      181.90 180.97 bar
16_MASSF   18.40  18.51 t/h
16_MFN2    22.30  20.87 %
16_MFH2    66.70  65.38 %
16_MFNH3    3.90   3.85 %
16_MFAR     5.20   5.03 %
16_MFC1     4.80   4.86 %

6_T    5.90    5.49 C
6_P 181.10 180.97 bar

EXTENT1        1896.23 kmol/h
R-101_DTEQ1       0.00 K

8_T        329.30 327.81 C
8_P        191.20 189.92 bar
8_MFNH3      2.80   2.89 %

7_T   10.90  11.13 C
7_P 188.30 190.35 bar

18_P       10.10 10.05 bar
18_MASSF    0.80  0.83 t/h
18_MFN2    11.60 11.64 %
18_MFH2    32.40 33.61 %
18_MFNH3   40.40 40.59 %
18_MFC1     9.40  9.15 %
18_MFAR     4.90  5.01 %

10_T 408.40 409.53 C
10_P 186.40 191.28 bar

WCOMP_POWER 21471.00 21745.00 kW

BFW_T        24.30 24.34 C
BFW_P        40.40 40.35 bar
BFW_MASSF 40.80 40.34 t/h

9_T          502.80 505.00 C
9_P          195.60 191.65 bar
9_MFNH3       14.60  14.61 %

19_MASSF 62.70 63.08 t/h
```

Figure 2: Sample flowsheet for data reconciliation case study
A simplified ammonia synthesis loop

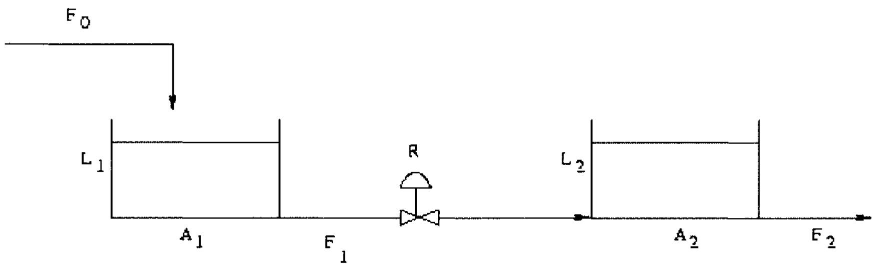


Figure 3: Dynamic Example – Connected Tanks

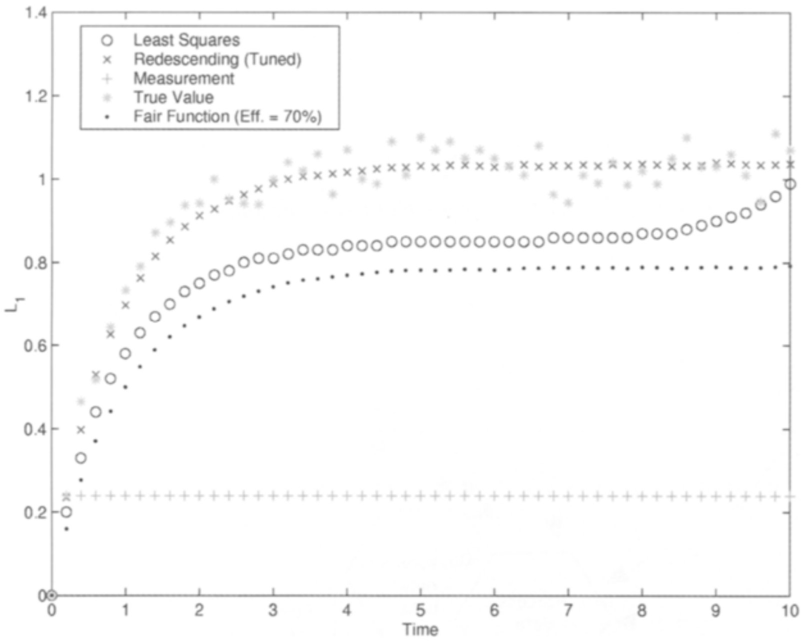


Figure 4: Fitted Values for L_1 in Tank Example

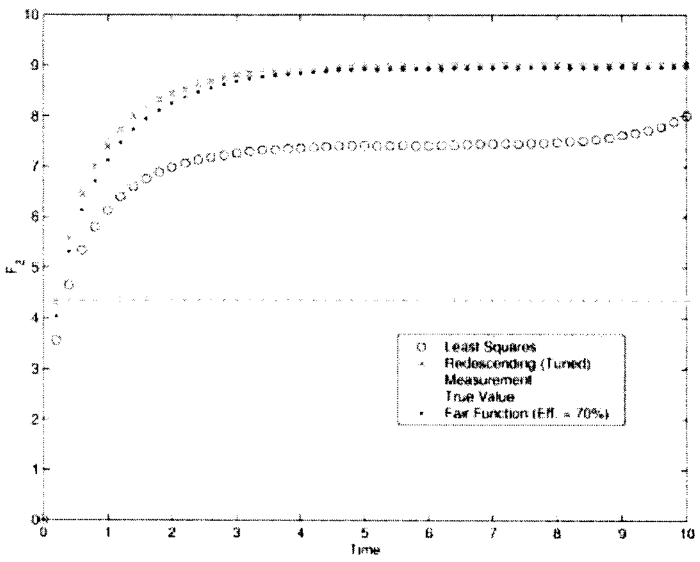


Figure 5: Fitted Values for F_2 in Tank Example

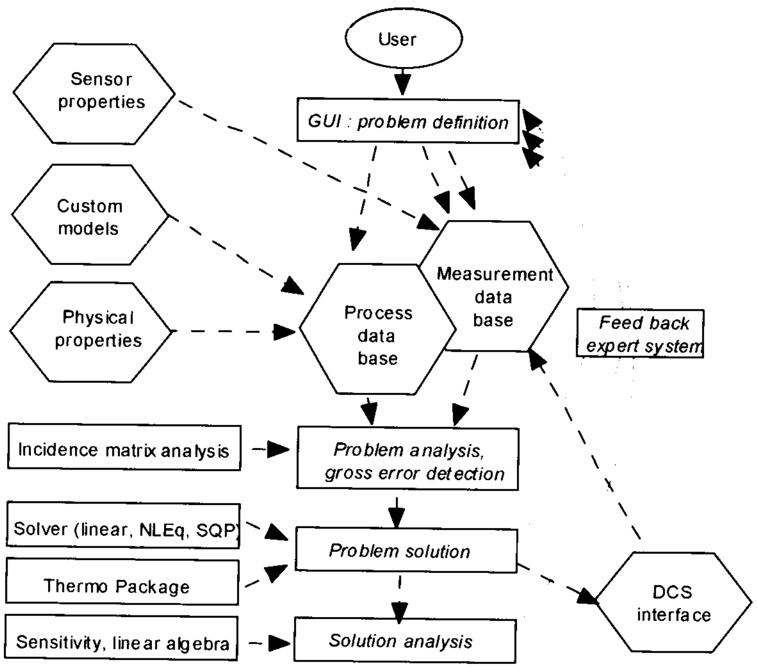


Figure 6 : Data flow in a typical data reconciliation package

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 3.5: Computer Tools for Discrete/Hybrid Production Systems

L. Puigjaner, M. Graells & G.V. Reklaitis

The software tools for batch processes may be classified in two broad classes. The first class involves modelling, simulation, and analysis of the physico-chemical processes that take place during batch operations. These tools can be associated with the process simulation methodology that constitutes the core of the first Cape Open project. The second class is designed to support the decision processes at the different managerial and operational levels of the plant operational hierarchy, which are the topics included in the scope of the follow-up, Global Cape Open Project. This class of tools may be further classified on the basis of the planning, scheduling, monitoring, and control tasks, which they support, specialized to the batch processing case. In this chapter we review a representative set of tools from these two classes that are available commercially. While there is much innovative research and development in both academic and industrial groups in this domain, we have excluded experimental software and proto-types from our discussion.

3.5.1 PROCESS SIMULATION & ANALYSIS

3.5.1.1 Continuous Process Simulation

Process simulation systems were initially conceived to allow modeling of continuous steady-state process flowsheets, as represented by classical petrochemical processes such as the hydrodealkylation of benzene or the production of ethylbenzene. The process flowsheet model generally consisted of models of the unit operations present in the flowsheet that were linked through the process streams which constituted the external input and output variable sets for these models. The solution of the flowsheet model was complicated by the presence of recycle streams, which created linkages between the unit models. To solve the resulting large scale coupled algebraic equation systems a natural decomposition strategy called the sequential modular approach was widely adopted. Under this approach unit models where executed in a serial fashion in the direction of the principal process streams with the recycle stream, or more precisely, tears streams serving as iteration variables. AspenPlus (Aspen Technology.), HYSIM-HYSYS

(Hyprotech/AEA) and PRO/II (Simulation Sciences) remain the most significant examples of this type of simulation technology. These tools gradually evolved to contain substantial libraries of unit operations modules, including some which involved models consisting of differential equations, as well as extensive physical property estimation capabilities and supporting properties constant databases. Because of the desirability of allowing the easy addition of new process specific unit operations models or modifications of existing models, the need soon arose to allowing linkage or insertion of user-added FORTRAN subroutines. In recent years, this in part served to stimulate the idea of COSE (Cape Open Simulation Executive). In due course, with advances in efficient methods for the solution of large-scale algebraic systems, these methods were adopted to solve the entire set of flowsheet model equations simultaneously, resulting in the so-called equation oriented flowsheet simulation architecture.

A natural next step for flowsheet simulation was to provide the capability to model the dynamics of the process. Again the solution of such dynamic simulation models was initially approached using sequential modular strategies but contemporary systems have emphasized simultaneous solution methods. Tools such as DYNSIM (Simulation Sciences), HYSYS (Hyprotech/AEA) and AspenDynamics (Aspen Technology) embodied these architectures. Ideally such tools offer a consistent set of steady state and dynamic models of the same unit operations, allowing the user the convenience of ready transition from steady state to dynamic simulation of a given process. The DynaPLUS system marketed by Aspen Technology is intended to achieve precisely that aim.

3.5.1.2 Batch Process Simulation Tools

Since batch operations are inherently dynamic, one would not expect steady state flowsheet methodology to be applicable to batch process simulation. On the other hand, one might well assume that dynamic simulation systems would be extensible to handle batch processes. While this is the case for individual operations, it is not when one seeks to model the entire network of batch operations typical in batch chemical processing. The complicating factors include the need to handle the discontinuities inherent in the start and stop of the tasks which comprise a batch process and the fact that with batch processes the description of the set of chemical–physical tasks which must be executed to manufacture a given product (the recipe) is distinct from the set of equipment which are used to perform these tasks. Since the equipment items are generally multipurpose, the definition of the flowsheet for a batch process requires a series of task to equipment assignment decisions which may be governed by equipment availability, the availability of resources such as feedstock's, catalysts, and hold tanks, product priorities, and other state dependent or even economic factors. Thus, in the batch case, the flowsheet is in effect defined dynamically as the recipe is executed. These additional numerical and decision aspects are not accommodated by conventional dynamic flowsheet simulation systems.

Of course, simulations of individual batch operations, such as batch reaction and batch distillation, can be modeled and solved within the framework of dynamic simulation methodology. These simulations are most commonly structured as stand-alone modules or programs, which either share the physical properties estimation features of an associated process simulation system or employ an independent physical properties package. BATCHFRAC and BatchCAD are examples of these types of commercial systems. Other examples of programs of this kind include Batch Colonne, Batch Réacteur (ProSim) and BDIST-SimOpt (Batch Process Technologies). Such systems may offer additional features such as operating profile optimization or parameter optimization to fit empirically observed operating profiles.

To facilitate the simulation of processes involving a mix of continuous and batch operations, as might arise in single product plants in which some of the operations must necessarily be of a batch nature, several of the process simulators do allow use of batch reactor and batch distillation modules, of the type noted above, within a continuous steady-state model of the process. As is the case with a plug flow reactor model, the integration of the batch operation model is carried out internal to the module and averaged output streams are computed for use by the downstream continuous unit operation. Conceptually, this can be viewed as following the batch operation with one or more implicit holding tanks, which at the termination of the batch effectively provide the averaged output stream or streams, which feed the succeeding continuous unit. This type of linkage of steady state and dynamic model types can readily be accommodated in the sequential modular architecture since all unit operations models are treated as closed procedures. It can also be handled within equation-oriented systems provided such system also accommodates closed procedures. Although the AspenTech and SinSci steady-state products thus do permit interfacing of batch and continuous operations in this fashion, the entire process model remains effectively steady state.

As noted earlier, the effective simulation of batch processes requires representation of the dynamics of the individual batch operations, the decision logic associated with the start and stop of operations, as well as the decisions associated with the assignment of equipment and other resources to specific operations as defined through the product recipe. Some conventional dynamic simulators (*e.g.*, HYSYS) do offer tools for programming the decision logic associated with a series of events to be executed at specific points during the execution of a simulation run. In this way it is possible to simulate certain classes of batch processes. However, this type of adaptation of the dynamic simulation executive only address part of the requirements for the simulation of typical multiproduct batch processes. The more advanced capabilities of a combined continuous-discrete simulation architecture are required to accomplish this in a general fashion. The BATCHES (Batch Process Technologies) simulation tool does accommodate the above mentioned batch process features and uses advances in combined discrete-continuous dynamic simulation methodology to follow the progress of the batch plant over time.

A BATCHES simulation model consists of three main building blocks: a recipe network, an equipment network and a set of processing directives. The equipment network defines the equipment specifications and the connectivity or transfer limitations between the equipment. The recipe for the manufacture of a product is modeled as a network of tasks and each task as a sequence of subtask. A task consists of all of the operations performed in a single item of equipment: a subtask consists of a model of one of these operations. Tasks have associated with them requirements of specific types of equipment and selection priorities. BATCHES provide a library of models of various types of operations (heating, cooling, decanting, batch reaction, etc.). Subtasks may have associated with them additional requirements for resources types and levels such as operator types and utilities as well as definition of conditions under which the subtask is to be terminated. These may be state dependent (a specific temperature or composition level is achieved) or directly specified (completion time or duration). Processing directives consist of information that drives the execution of the process over time. These include information such as the amounts and sequences in which the various products are made, the due date and amount for finished product delivery, or the amounts and frequency of raw materials deliveries or other resource releases.

BATCHES uses a dynamic solution strategy under which the dynamic models associated with all of the subtasks that are active in a given time period are solved simultaneously using a DAE solver. As the solver advances through time, the occurrence of subtask termination or start events is tested at each solver time step. As events are identified and occur, the set of active subtask models is reconfigured and the solution process continued. This computational approach is effectively a decomposition strategy as only the models of the subtasks active at a given point in time are actually included in the integration step executed. To accommodate stochastic parameters, BATCHES allows Monte Carlo sampling of simulation parameters from a library of distributions using established techniques from the discrete event simulation literature. It also provides linkages to physical properties estimation capabilities. More complex decision processes, such as solution of an assignment or scheduling model, can be accommodated by defining event conditions under which the simulation is interrupted, the information necessary to execute the decision model is assembled, the decision model solved, and the simulation of resulting actions transferred back to the simulation executive.

3.5.1.3 General Purpose Modeling Languages

In BATCHES non-standard subtask models as well as unusual decision or event logic can be accommodated using the vehicle of user-supplied models written in conventional programming languages. This is a limitation, which the tool shares with conventional steady state and dynamic simulators that do not provide a higher level modeling language. That limitation is mitigated by general-purpose process modeling and simulation software such as Speedup, gPROMS and Abbacus. gPROMS is a successor to Speedup and is in turn superceded in part by the academic package, Abbacus. gPROMS does accommodate the full range of model types, from purely batch to purely continuous. It allows model developers to write models of the most complex processes and their operating procedures – from the detailed mathematical equations for individual components, to the structure and operation of large complex systems composed of many such components – using a sophisticated natural language. The complexity of the processing of the resulting equations and the solution methodology are handled through system utilities and thus can largely be hidden from the user. gPROMS offers extensive facilities for linking to external software across a range of hardware and software platforms. It also provides advanced features such as dynamic optimization of continuous dynamic models thus allowing simultaneous optimization of the parameters of equipment and operating procedures. Of course, since it is a modeling system, it leaves to the user the definition and formulation of the models and particular decision processes of batch operations.

3.5.1.4 Batch Process Development and Information Management

An alternative approach to supporting the development and operation of batch processes is to offer a software package that provides the capabilities of organizing and managing recipe information together with a suite of tools for operating on that information, including a rudimentary recipe simulation capability. Linkage to more detailed tools such as stand-alone batch reactor and batch distillation packages can also be provided. This is the approach that has been used in two of the packages described in this section. The additional functionalities not provided by these tools, namely the generation of operating procedures and the execution of batch process hazards analysis, are available from IPS. We describe these developments in the third part of this section.

Batch Plus (Aspen Technologies) is a general-purpose system designed to model the complex, recipe-based processes found in the batch process industries. The key design concept is the representation of a batch process using the multilevel "process recipe" metaphor. Based on the recipe description of the process, software allows design, engineering, and scale-up to be performed. Among features provided is the AspenTech electronic Batch Record System (AeBRS) which is designed to address manufacturers' needs in the areas of work orders management, procedure management and documentation management.

The BaSYS system of software is designed to help batch processing companies improve communication between chemists and engineers, perform faster process scale-ups and more efficiently allocate existing equipment. BDK, the core of BaSYS, is an integrated batch process development and design environment that helps companies accelerate many aspects of batch process development, from route synthesis to implementation. BDK offers tools for enabling the rapid selection from among alternative synthesis route for manufacturing, improving waste processing and facility utilization, and providing a "knowledge warehouse" that documents the development process. Neither of these systems are intended to provide rigorous process simulation capability but could in principle be interfaced to a simulator, given that much of the recipe and other data required for a simulation is contained in the information base that these systems maintain.

Two critical tasks in the development cycle of a batch process are the synthesis of operating procedures given basic recipe information and the analysis of the operating procedure to identify, evaluate and mitigate potential operating hazards. These two functionalities are not provided by the systems described above. However, the generation and validation of operating procedures for new processes constitute time-consuming and error prone activities that lend themselves to computer support. Moreover, with the increasing complexity of operating procedures and governmental and social pressure to reduce safety and environmental incidents, there is strong incentive to conduct process hazards analysis and mitigation for all new and existing processes. Since considerable portions of this activity are also repetitive and very labor intensive, there is an opportunity for intelligent tools to support process hazards analysis. PHASuite, a system initially developed at Purdue University to support this task, is now available from IPS. PHASuite consists of two closely integrated components, iTOPs and Batch HAZOPexpert. The former component serves to synthesize the detailed sequence of instructions an operator in a chemical plant needs to follow in order to manage a process safely and optimally. BHE serves to systematically analyze the process in question to identify, assess, and mitigate the possible hazards that can occur.

PHASuite follows ISA S88 batch standards for modeling batch process information. In iTOPS, Graf chart-based concepts are used to represent the batch process and a hierarchical planning technique is employed together with information about the process materials, equipment, and chemistry to synthesize procedures at the phase level. BHE uses a logical separation of process information into specific and generic components, qualitative causal digraph models and a two-layered Petri-net based model of the process to systematically identify possible hazardous situations that can arise in a chemical process. The two PHASuite components are fully integrated, including the creation of the various representations of the process recipe and logic. The user inputs batch process information – the materials, the chemistry, and the equipment – through a top-level interface. PHASuite returns a complete batch record documents, including safety instructions and a list of potential hazards classified by severity. The systems has been tested extensively in pharmaceutical applications, resulting in documented substantial savings in engineer time, and considerable improvements in accuracy of the resulting operating procedures and completeness in hazards identification.

3.5.2 PROCESS PLANNING & SCHEDULING

3.5.2.1 Planning

Production planning for individual batch plants and planning for entire supply chains consisting of multiple interacting batch plants can in principle be performed using generic planning tools. Thus, generic linear programming (LP) and mixed integer programming (MILP) packages such as CPLEX (ILOG) can be used providing that the user is prepared to develop the case specific formulation and provide the appropriate data interfaces. Alternatively, if manufacturing recipe details are aggregated and thus the plant treated as a black box, then various capacity planning tools offered by ERP vendors can be applied. However, in batch manufacturing applications, the details of the batch operations often prove to be important because equipment and other limited resources are shared among the various products under consideration, thus production capacity is constrained. Unfortunately, at the present time there are no commercial tools which can accommodate this level of detail without explicit user developed models. Such a suite of tools is available for petroleum refinery planning and supplies chain applications in the form of Aspen Technology's PIMS and Ref-Sked packages, which are based on LP methodology.

3.5.2.2 Scheduling

As at the planning level, scheduling applications for batch operations can be developed using general purpose scheduling toolboxes such as Scheduler provided by ILOG. Aspen Technology's MIMI system also falls into this category. However, the use of these toolboxes to batch processing problems requires knowledge of the strengths and limitations of the individual tools and experience in scheduling application formulation. This is a level of expertise beyond that of plant engineers. However, several tools do exist that have been developed specifically for batch processing applications, two of which, gBBS and Virtecs, are described in this section.

Available through Process Systems Enterprise (PSE) Ltd, gBBS is the outcome of extensive research conducted at the Centre for Process Systems Engineering at Imperial College, London. It is a scheduling tool designed for multi-purpose process production, from purely batch to purely continuous. Process specific issues such as cleaning, recycling and intermediate materials that are also final products can all be treated. A complete gBSS application is composed of data such as product demands and inventory, product recipes, plant resources, staff and maintenance schedules and the current status of plant equipment. The user's data is checked for consistency and converted automatically into one of three MILP (mixed integer linear programming) formulations. Specialised MILP algorithms requiring little or no user intervention are used to find the solution which is guaranteed to be optimal if allowed to run to convergence and the results are then processed and presented in an engineering form. When solving especially large problems, gBSS breaks them down into several smaller ones and combines the results into a final solution. Recipes and processes are modelled using the State-Task Network representation and the modelling language is designed to express complex plants in a simple and flexible way. Established models can be accessed from a 3rd-party front-end such as MS Excel.

gBSS can be configured to solve three types of scheduling problem formulations: Short-term scheduling, Campaign planning, and Scheduling for Design. In short-term scheduling, the plant layout, processes and products are known and change infrequently. Other data may change with each run of gBSS - demand, deadlines, availability of equipment and staff. Transport costs and times can be included so that multi-site production and distribution can in principle be accommodated, at the cost of increased computational burden. The campaign planning form is appropriate for applications in which product demands are stable or may be forecast accurately thus allowing longer production horizons to be divided into a number of campaigns. The Scheduling for Design option is used to find the optimum plant resources given a fixed set of demands and their deadlines. gBSS can take account of both capital and running costs and find the ideal plant for a minimum initial cost or a minimum lifetime cost. It is also suitable for designing extensions to existing plants. This facility is available for plants operated in either campaign or short-term scheduling mode.

Advanced Process Combinatorics, Inc. (APC) has developed Virtecs for process scheduling and planning. Virtecs is based on Mixed Integer Linear Programming (MILP) technology and offers the ability to model processes at a high level of detail including constraints on limited material shelf life, process vessel storage, shared storage, labor, utilities, minimum/maximum inventory levels, complex bill of materials, piping connectivity, equipment downtime, multiple stages of production, parallel equipment, and process changeovers. The MILP based scheduling tool can provide solutions in fully automatic mode and a user can readily make process changes. APC has recently released Virtecs v5.0 that includes support for Internet based use and publication of schedules, developing schedules from previous schedules and efficient human override of automated scheduling capability so that the tool can be used anywhere from a fully manual to fully automatic mode. Previous versions of Virtecs have been successfully used in the pharmaceutical, specialty chemical, food and beverage, consumer products, lubricant, and retail industries. The MILP based solver behind Virtecs is highly customized and routinely solves scheduling problems involving hundreds or thousands of tasks in one or two minutes on a desktop personal computer. While MILP scheduling technology is in its infancy and will continue to grow in capability, the existing advantages include explanations of why demands cannot be met on time, lower consulting costs for installing and supporting tool use, ability to support engineering applications such as expansion studies or debottlenecking, online applications that can be readily extended by the user to support new products or process equipment changes.

Because of reliable and accurate automated solution capability, Virtecs can be used in a distributed fashion to integrate and mediate detailed coordination between multiple facilities. Because of the versatility afforded by their solution technology, APC has also applied Virtecs technology to warehouse management applications and integrates them with upstream production and downstream distribution activities. APC also uses their MILP solver underneath a tool that selects and schedules projects in research and development pipelines. The main components of the Virtecs tool are a natural modeling language for describing processes, reporting system, database for managing multiple scenarios, graphical user interface with readily extensible Gantt Chart, and a highly engineered MILP solver based on implicit formulation generation. Because of the extensibility of APC's MILP approach, their tools can be used in conjunction with simulation capability to analyze the impact of uncertainty on the performance of solutions and manage risk in high-level applications such as pricing studies, mergers and acquisitions, and supply chain design/operations. Because of close ties with Purdue University, APC has world-class research and development capability and has demonstrated significant functionality gains with each new product release.

3.5.3 SCHEDULING-CONTROL INTERFACE

Once a scheduling solution is developed using one of the tools described above, it must be implemented in the production environment and executed through the batch control system. At the level of process detail represented in scheduling applications, operational details such as valve opening and closing are normally not taken into account. Moreover, it is usually inefficient to rerun the scheduler with every modest process variation or delay that is encountered. Thus, there is a practical need for either manual or automatic conversion, if necessary, readjustment, and execution of the scheduling solution. The *SUPERBATCH* package, offered by PSE Inc and prototyped at Imperial College, offers the ability to work on-line, in conjunction with standard batch control systems, and to automatically update schedules as small delays and process variations occur. The *TotalPlant* Batch system offered by Honeywell provides the interface for the user to execute batches manually following a schedule created off-line and to manually readjust timing as needed, with all of the recipe and equipment details, communications with the control system, and handling of alarms and messages handled automatically. In this section we briefly describe the functionalies of these two systems. The reader should note that these functionalities are to varying degrees provided by other commercial batch automation systems.

Designed for embedding within manufacturing execution systems (MES) or linkage to a scheduling product, SUPERBATCH provides the capabilities for static off-line short-term schedule readjustment as well as for on line schedule correction, with changes broadcast once a minute to screens in departments throughout the factory. It uses a modeling language which conforms to ISA S88.01 to describe the plant, the materials, recipes and ancillary procedures (such as changeovers and cleaning suitable for hygienic industries) as well as the production batches themselves. For off-line scheduling applications, once embedded within a graphical user-interface to present the schedule, SUPERBATCH will find the earliest possible time for each batch, subject to the constraints of the model. Equipment allocations may be pre-defined or picked by the user from the feasible set which SUPERBATCH offers. SUPERBATCH also delivers the profiles of each plant item as needed to draw a graphical schedule. For on-line schedule adjustments, SUPERBATCH provides an on-line monitor, which executes the schedule, and a versatile interface, which accesses control systems (and simulators) usually across a network. Once a minute, the current status of the plant is read from the control system and the schedule is updated to match. In this fashion, delays and stoppages can be accommodated automatically. SUPERBATCH then initiates execution of any operations due by sending the appropriate commands and parameters to the control system. The system also issues alert messages, which provide advance warning of impending events requiring operator attention. SUPERBATCH is written in object-oriented C++ adhering to the ANSI draft standard, using rpc for it's networking, and is portable to a wide variety of environments and control systems.

The TotalPlant Batch systems provided by Honeywell is an open, object-oriented software application for modular batch automation. Developed around ISA S88.01 it provides batch recipe and equipment management, batch management and execution, an integrated operator interface, easy to use graphical configuration tools, and rudimentary batch simulation capability. The assignment of units can be done dynamically at run time or batch creation time. Implemented to run under Windows NT, it is designed to operate with Honeywell's TotalPlant and the PlantScape control systems. The major components of TotalPlant Batch and their functionalities are the following:

- *Batch View* allows the user to create batches, execute them, review batch related information and respond to alarms and messages.
- *Batch Server* monitors and controls execution of batch procedures and displays batch execution information
- *Batch Data Server* communicates and between the Batch Server and the phase logic sequences in the control system
- *Batch Recipe Editor* allows the user to specify recipe parameters and to graphically construct the recipe sequence using sequential function charts and tables
- *Batch Equipment Editor* allows the user to configure and maintain the physical plant model used by the other components

The system also provides customizable event archiving and batches report generation utilities.

3.5.4 CONCLUDING REMARKS

As should be evident, the suite of functionalities provided by existing software tools for batch process systems engineering is quite broad but as yet far from complete. While there is much exploratory research conducted within the academic community, commercial developments seem to be lagging. For instance, with the exception of BATCHES there is limited recognition in the existing software of the highly stochastic nature of the operation of most batch processes. Issues of robustness and risk are simply not addressed. Also missing is the capability to perform batch optimization through the optimal selection of recipe parameters such as conversions, batch times, separation fractions, a capability that is generally available with steady state flowsheet simulation systems. Systematic tools for batch monitoring and fault diagnosis are notably absent: a combination of trend analysis and predictive dynamic models would appear to be required.

Integration of the existing tools is as yet a distant dream. For instance, the seamless linkage of planning and scheduling tools or of scheduling and simulation tools is not available. Furthermore, at present there clearly are no standards that would facilitate the integration of these tools through suitable common date structures. The S88 standard for representing batch recipes and equipment could well serve as the heart of such a common data structure. Indeed a number of the tools have used the recipe structure and naming conventions promulgated under that standard in defining their input data structures. However, this structure must be further elaborated to encompass all of the information associated batch product and process design and batch plant operations.

The batch operations domain evidently offers great opportunities both for methodology research and for commercial software development.

3.5.5 CONTACT INFORMATION

The following list provides contact information for the products and tools cited in this chapter. The reader is invited to pursue the latest developments through these electronic sources.

Advanced Process Combinatorics, Inc.
Products: Virtecs
Information: www.combination.com

AEA Technology Engineering Software
Products: BaSYS, BDK, HYSIM, HYSYS
Information: www.hyprotech.com info@hyprotech.com

Aspen Technology, Inc.
Products: AeBRS, AspenDynamics, AspenPlus, BATCHFRAC, Batch Plus, DynaPlus, Mimi, PIMS, Re-Sked
Information: www.aspentech.com info@aspentech.com

BatchCAD Ltd
Products: Batch CAD
Information: sales@batchcad.com

Batch Process Technologies, Inc
Products: BATCHES, BDIST-SimOpt
Information: www.bptech.com sales@bptech.com

ChemEng Software and Services
Products: BATCHDIST
Information: ChemEng@BTinternet.com

Honeywell, Inc.
Products: TotalPlant Batch
Information: www.iac.honeywell.com

ILOG, Inc
Product: CPLEX, Solver, Scheduler
Information: www.ilog.com info@ilog.com

Integrated Process Solutions, Inc.
Products: BatchHAZOPexpert, iTOPS, PHASuite
Information: www.ipsol.com

Process Systems Enterprise. Ltd.
Products: gBBS, gPROMS, SUPERBATCH
Information: www.psenterprise.com

Simulation Sciences, Inc.
Products: DYNSIM, PRO/II. PRO/II Batch Module
Information: www.scimsci.com

Part IV: Making CAPE-Tools

Part IV shows what are the current tools (component software, objects, middleware, databases, XML...) and methods (data modelling, UML ...) used by the CAPE software industry to develop the new frameworks and tells about current standards projects and their status. Through examples such as CAPE-OPEN, PlantData XML, PI-STEP, the reader will get an insight on how these standards are being developed and implemented. This part is mostly written by software specialists and experienced CAPE tools providers.

Chapter 4.1, by Köller et al., gives an in-depth assessment of some of the technologies: middleware, COM and CORBA, XML, application integration. In addition, it introduces the UML notation and the Unified development process, which are now widely recognized as standard techniques for specifying, designing and developing object-oriented systems.

Chapters 4.2 and 4.3 present two approaches to developing standardized data models for the process industries. In 4.2, Teague develops his Plant Data XML standard for data exchange, which is based on the eXtensible Markup Language (XML); XML is a core technology of the World-Wide Web and of Microsoft's .NET architecture; it is the key to web services and to the future semantic web, and is used as well for supporting data communication between heterogeneous platforms; chapter 4.2 shows the use of XML for unit operation data, with an emphasis on heat exchangers; then, chapter 4.3 by Murris summarizes several years of efforts put into the development of the PI-STEP (STEP for the Process Industries) standard; these efforts led to the definition of the ISO-10303 Application Protocol 221, which uses the EXPRESS formalism. API 221 is interesting as it defines a data metamodel (a model of data models), which has been further developed into the STEPLib dictionary of process plant classes.

Finally, Chapter 4.4 by Belaud et al. presents the CAPE-OPEN (CO) standard, developed in two successive international projects, CAPE-OPEN *and Global CAPE-OPEN. As we wrote in our introductory chapter (part 1), CAPE-OPEN and Global CAPE-OPEN motivated us to prepare this book, and therefore the CO technologies and development process are evoked in several chapters; however, Chapter 4.5 is the only chapter totally devoted to CO; the chapter gives an overview of it and looks into parts of the standard, in order to give a basic understanding; the interested reader will find a lot of additional information on the CAPE-OPEN Laboratories Network website, www.colan.org.*

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 4.1: Methods & Tools for Software Architecture

J. Köller, J.-P. Belaud, M. Jarke, A. Kuckelberg, & T. Teague

4.1.1 INTRODUCTION

It is an obvious trend that software systems, especially the ones used in an enterprise environment, have become more and more complex in recent years. Applications have changed from simple stand-alone batch programs running on mainframes to complex, highly integrated, and distributed systems. The recent trend of building internet-enabled systems brings even more problems into the game. For example, large enterprise resource planning (ERP) systems such as SAP R3 can now be found in every major company. They consist of many functional modules that have to interact and must have the ability for integration with other external systems such as databases or e-commerce solutions.

Unfortunately, many of the existing systems that were not designed to run in such an environment and are still in use in many companies (often performing mission critical tasks) do not fit into that picture. They are closed, monolithic applications, which are inflexible and not easy (if at all) to integrate with other software systems. Additionally, maintaining these programs is very expensive especially in the case of non-modular systems that were implemented in 'ancient' languages such as FORTRAN, COBOL or even Assembler.

These problems apply to the CAPE domain as well. Many different tools are needed throughout the design and operation of a chemical plant. The systems employed for that task range from computer-aided design tools (CAD) over process simulators and optimisers to costing and ERP systems. But nowadays all of these systems are mostly stand-alone tools, which offer no direct integration among each other.

To change this situation for the CAPE domain and for software development and design in general new paradigms, systematic approaches and flexible technical solutions are needed. New paradigms (compared to the ones used for creating the existing monolithic systems) are mainly based on object-oriented and component-based modelling of a software system. These approaches facilitate the design and implementation of loosely coupled systems. These systems are rather flexible and easy to integrate with other pieces of software written following the same paradigm. More on the idea of object-oriented design is presented in the following sections. Additionally, methods for using these paradigms to develop well-designed software systems are shown. We also discuss several technical approaches and frameworks, which support the software developer in transforming a good and flexible design into running software, including application integration via standardised data exchange and middleware solutions.

If you a look at many large-scale up-to-date enterprise applications in industry, you can easily recognize that the employment of the frameworks and techniques presented here facilitates the development of reliable and flexible software systems. Additionally, most of the approaches shown in the following sections have been used for the development of the CAPE-OPEN standard (see Chapter 4.3) and the implementation of CAPE-OPEN compliant software. This is a good example of how state-of-the art software development techniques are used to create high quality software in the CAPE domain.

Of course, only a very brief introduction to all these issues can be given in this book. For a deeper understanding of the methods, frameworks and techniques, the resources listed in this chapter are recommended many of which are available online.

4.1.2 BASICS OF SOFTWARE ENGINEERING

This section gives a brief overview of some basic principles of software engineering. A short history of the development of programming and design principles is given to motivate why a change of the paradigms in software engineering has taken place. This will make clear why legacy software systems are causing so many problems and how they can be avoided. Among other issues the following subjects are addressed: A brief sketch of object-oriented development and design; concrete object-oriented specification methods are presented in sections 4.1.5.2 and 4.1.4.1, namely the Unified Modelling Language (UML) and the Unified Process (UP). Next, the ideas of component-based software and middleware are presented. Again, technical implementations such as COM and CORBA will be discussed later (Section 4.1.5.1). Because the migration of old software to new standards (*e.g.* transforming FORTRAN77 code to an object-oriented design) is an important issue, some introductory material on that is shown. This section concludes with some general thoughts on systems and data modelling as well as data representation. As an application, the universal data representation format XML is presented in Section 4.1.5.3.

4.1.2.1 Evolution of Programming Principles

The architecture of CAPE software follows the mainstream of software development and programming principles [21,34,35]. Therefore, CAPE software has changed as well. This is one reason for the difficulties arising when legacy CAPE software is to be integrated with other software systems. To understand the problems arising with the evolution of software development paradigms it is helpful to summarize this development in brief.

In general the evolution can be divided in five principles: linear procedural programming, structured procedural programming, event-driven programming, object-oriented programming and component-based architectures. The programming languages themselves can follow different programming paradigms such as imperative, functional or logic programming, but are not of deeper interest in our context because most problems in developing modern (CAPE) software architectures result from the changing programming principles, not from the paradigms.

- ***Linear procedural programming***: Initially, programs were written as monolithic command sequences. Even if basic program structuring techniques were available, the program was developed linearly in an imperative manner using programming languages like Assembler, COBOL, ALGOL60 and early versions of FORTRAN. Main control structures used within the programs are jumps (GOTO), jumps with conditions (IF-THEN-ELSE) and loops (FOR-NEXT/WHILE-DO/REPEAT-UNTIL) which were usually combined to one huge block of code and data. Therefore, such software systems were often comprehensible only by a small group of involved developers.
- ***Structured procedural programming***: The next wave of programming principles in the middle of the 1960's was strongly influenced by C.A.R. Hoare and E.W. Dijkstra. The programs were structured into functional units or sub-procedures. These procedures were identified and coded as independent units. The data was encapsulated, or hidden, from the calling procedure, and the programming languages got a theoretical foundation comprising syntactical specification of the language and compiler theory to qualify them for universal problem solving. This is likely to be the principle used by most current (legacy) CAPE systems because current CAPE software is often descended from this era. Popular languages were FORTRAN77, ALGOL68, PASCAL, C, PL/1, although languages with other programming paradigms like LISP or PROLOG were also used. The programs themselves remained monolithic procedural constructs, some with module concepts allowing an extension or replacement of functional program units.
- ***Event-driven programming***: Event-driven programming approaches were needed for computer-based process control applications, where a closed procedural approach is inadequate. In event-driven programs, the main program is a simple routine, which is essentially a timer loop, waiting for

events to happen. When events are detected, an interrupt-driven event-handling subroutine is called to handle the event, before returning control to the main program.

- ***Object-oriented programming***: Object-oriented programming, introduced in the 1980's, uses an event-driven approach of cooperating objects, rather than a monolithic, beginning-to-end procedural paradigm. This approach focuses on how to map real-world, user-controlled, event-driven scenarios, such as mouse clicks or keystrokes, into the program. Objects are identified as entities similar to those in the real world with properties (encapsulated data) and a behaviour (encapsulated procedural methods). This ability to hold both the procedure and the data in the object is conceptually very different from procedural approaches in which data are held in (potentially large) data structures separate from the procedural code. Programming languages such as C++, JAVA, and SMALLTALK provide language constructs to reflect the object-oriented view [24].
- ***Component-based systems***: Component-based systems can be thought of as pre-built collections of objects that exist independently of any single main application program, whereas in "just object-oriented" systems the objects are typically created and used in the context of a single specific application program. In component-based systems, applications can be built from pre–assembled components, which may be provided by multiple, independent supplier companies. To facilitate this, standardized frameworks to access the components programming interfaces are required. In component-based systems, some of these components are "servers" for other components who are "clients". Often the client components and the server components are on the same computer, but in general, they can be located on different computers communicating over a network. The only important thing for a client component is to know how to communicate with the server component. To achieve this implementation and location independence of components, a middleware layer is needed which covers the technical details of the communication and which provides a common and standardized interface to a components. Current examples of middleware solutions are CORBA, (D)COM or EJB but also distributed and client-server systems which use (system-specific) standardized communication protocols [24,25,37].

Considering the very different programming paradigms of legacy software, which uses structured procedural programming, and modern software systems, which use object-oriented and component-based architectures, we can see that it is not easy to evolve legacy software for use in modern software systems. We will come back to this issue in Section 4.1.2.3.

4.1.2.2 Middleware Principles

Component-based applications consist of several pieces of software, which are executed independently and may reside on the same host or on remote host over a network (such as intra-/extra-/internet). For such a component-based application to work, the components must be able to interoperate, *i.e.* to exchange data and issue method calls to each other via the component's defined interface. The technology that implements an infrastructure for component communication is called middleware. A middleware solution provides standardized mechanisms for the definition of component functionality and communication as well as additional services easing the use and implementation of component-based software. Currently, the most prominent object middleware solutions are (D)COM from Microsoft, CORBA from OMG and Enterprise Java Beans from SUN to all of which we will return in Section 4.1.5.1.

The middleware described so far is called object middleware, as it is an intermediary for components, which have interfaces following the object-oriented paradigm. There are two other kinds of middleware, which should be named but will not be discussed here: remote procedure calls (RPC) and message-oriented middleware (MOM). Further information about these approaches as well as a good general introduction to middleware principles can be found in [33]

Basically, a middleware framework consists of four parts: an interface definition language (IDL), an interoperability protocol, an object request broker (ORB), and additional supporting services. The IDL is a language, which describes all accessible data and methods of a component. It is very similar to a programming language but contains method headers only, no implementations. Most IDLs follow the object-oriented paradigm. The interoperability protocol defines how the components communicate which each other. It defines valid message types and formats and how their content should be interpreted. One of the most important aspects of the interoperability protocol is the definition how the data structures defined in terms of IDL (*e.g.* a string or integer) are translated into network messages and vice versa. This process is called marshalling (resp. unmarshalling).

Having defined the IDL and the low-level communication, an entity is needed for coordinating component communication. Direct point-to-point communication between components would be possible but is very inflexible in multi-component applications. Therefore, most middleware frameworks include an object request broker (ORB), which is responsible for directing messages to and from different components throughout the network. It is the central piece of software in every advanced middleware because it is the logical backbone for all component communication. Additional supporting services are defined in most middleware systems for making the system more efficient and easier to use. Examples of such services are naming or persistence services and might be implemented as components themselves.

Figure 1 illustrates the use of a middleware infrastructure for component communication. A client application uses some server functionality. The IDL is used to define the interface that a server implements, i.e. the set of services that clients may request. The IDL interface specifications are compiled into client and server stubs. The client application calls a client stub to request a service. The client stub interfaces to the runtime system of which the ORB is a part and which invokes server code that implements the requested service through the appropriate server stub. The transmission of service requests and responses between clients and servers is handled by the runtime system of the middleware platform. Thus, applications need not deal with concerns such as: location of clients and servers on the network, differences between hardware platforms and operating systems, and implementation languages (*e.g.* data formats and calling conventions).

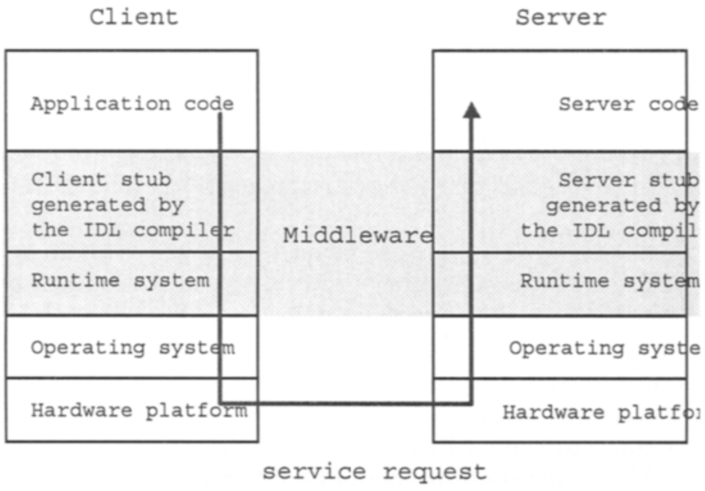


Figure 1 Middleware Approach

The example shows that client and the server can be located on different computers, based on different operating systems, and different hardware platforms (*e.g.* a Windows PC and a SUN Solaris machine). Additionally, different programming languages can be used on client and server side (*e.g.* C++ and Java). This integration aspect is one of the most important advantages of component and middleware technology. However, because most middleware frameworks follow the object-oriented approach, non-object-oriented languages (*e.g.* FORTRAN77) might not be suitable for use in a middleware environment or may require additional effort for integration. Furthermore, a hard problem is the integration of components that use different middleware platforms (*e.g.* one using COM, the other CORBA). We will come back to these issues in the next section. See also [6,12,33,37].

4.1.2.3 Migration of Legacy Software

In the last section we have presented the principles of component-based systems, which integrate components via middleware technology to form a complex distributed software system. Unfortunately, a lot of existing software was not designed using a component-based approach or was written before these concepts even existed. These systems are called legacy systems. In many C, COBOL or FORTRAN legacy systems, the implementation history may reach back several decades. Migrating these systems to a component-based environment is a non-trivial but nevertheless very useful task. By migrating these systems they are opened up to other applications and can be integrated with them. This exposes their functionality to the outside world, which would not be possible otherwise, and avoids wasting the money spent on designing and implementing those systems.

There are many approaches how to migrate a legacy system. None of them is really easy. Which way of migration is the best depends on how your legacy system is structured and what information about it still exists. The worst case for migration is a system that has no or only useless documentation and where all developers knowing the system are gone. Then the options for real migration are very limited. A complete re-implementation of the whole system may be the only choice. But things often look better. In most cases, source code is available along with some documentation. Depending on what language was used for the implementation there are several options. If a standard language such as COBOL, FORTRAN, C or C++ was used, there are several ways to move to a component-based system. We present two small scenarios in which the original code is completely preserved. Other solutions may include partial re-implementation and re-design of the system but will not be discussed here.

Migrating an object-oriented legacy system is rather easy. If the system was designed well it already encapsulates objects and data that can be directly re-used in a component-based environment such as COM or CORBA. The only thing that remains to be done is the connection to the middleware system. If the interfaces of the components directly correspond to the interfaces of the legacy objects, you need only a thin wrapper doing the translation from middleware calls to calls to the legacy objects. If the interfaces differ, the wrapper (in this case sometimes called mediator) must provide a translation between the component and the middleware interfaces.

If a non-object-oriented language such as FORTRAN77 was used, things are a little more complicated. In this case, a wrapper must accomplish two things. First, it must provide an object-oriented view to the non-object-oriented legacy code. This is a problem whose complexity strongly depends on the structure of the underlying system. For example, it may be necessary to extract code segments from monolithic code and restructure the data interface to fit an object-oriented approach. Because the wrapper usually will be in a different language than the legacy system, some kind of bridge between those languages is necessary. There are products able to solve this problem by cross-compilation or other bridging strategies. Second, the wrapper must provide a connection to the middleware system as in the first scenario. A case study of a migration of a FORTRAN77 thermodynamics package to the CAPE-OPEN standard can be found in the Migration Methodology Handbook from the CAPE-OPEN formal documentation set [11].

This can only be a short introduction into the wide area of software migration. But it is important to know that there are strategies for moving from legacy systems to component-based architectures without throwing away all the code that has been written and proven useful for a long period of time. [8] describes a systematic approach for migrating mission critical enterprise systems whereas [28,42] discuss general strategies for opening up legacy systems.

4.1.2.4 Modelling Systems and Data

The quality of software systems is strongly influenced by their architecture. Quality aspects include e.g. the clear separation of functional units and their encapsulation in components, methods or subsystems; a well documented program code with agreed layout, interfaces and documentation of the systems; effectiveness and efficiency of the system which often is strongly influenced by the basic architecture; the size and resources needed for system execution *etc.* Various metrics have been developed to measure software quality.

An important step in developing well-designed systems is a clear *a-priori* architecture and a modelling of the system, which can be refined in a top-down manner. Different modelling approaches for software systems and the data, communication or control flow within the systems have been developed as well as modelling techniques for specifying how systems are used. In general two modelling streams can be distinguished: system modelling and data modelling. While data modelling is of big interest when dealing with large amounts of data (data organization, storage, and retrieval) system modelling is important when a complex system with functional units and behaviour is designed [31,34,48].

- ***System modelling***: A software system consists of multiple parts, which usually interact and are related in a complex manner. As mentioned in section 4.1.2.1 the development and programming principles for software systems have changed and therefore requirements to system modelling have changed, too. Current systems mostly follow the object-oriented or the component-based approach which raises different requirements for system modelling:
    - ***Component modelling***: Which components or objects are part of an overall system, and what is their behaviour? What are their characteristics and how are they related?
    - ***System usage modelling***: How can a system be used, what API's are supported and what user interface is offered? What interactions are

necessary for which task and how can system components be used and coupled together for overall tasks?

- ***Control and message flow modelling***: Which message flow between different parts of the system does exist? Which actions are started by what components, what is the calling sequence of components respectively objects? Which actions interact in which way with system interfaces and system users?
- ***System state modelling***: What states can a system be in? What action can have which effects? What state transitions are allowed?

Tools for system modelling have to consider as many of these requirements as possible. A popular approach is the Unified Modelling Language (UML) described in section 4.1.5.2. More information about system architecture and modelling can be found in [9].

- ***Data modelling***: A system cannot do anything without data in multiple forms, e.g. persistent data, runtime system data, or data communicated throughout different parts of the system. Especially where data is to be exchanged between components, the data representation and semantics must be clear for all participating components. An introduction to data representation is given in the next section. For data semantics various modelling methods have been developed for different purposes. One of the most important tasks is the definition of data models for databases holding persistent system data and making it accessible in different ways. A popular data modelling approach is the entity-relationship model based on data entities and relations between entities. More about data modelling can be found in [41].

4.1.2.5 Data Representation

When modelling a system the data within is important, too. The control and information flow within the system and through the interfaces relies on communication data, which can be modelled as well. From an abstract perspective, data can be classified as structured, semi-structured and data without fixed format.

- ***Structured data***: Structured data is data that fits a predefined schema. It is known in advance what the type of data is, how different parts of the managed data are interrelated, and which data fields are mandatory and optional. The data of a relational database is an example of structured data with a well-understood theory behind it. Normally, using structured data has the best performance considering data storage and retrieval but limits the flexibility of the system. Examples for structured data are the predefined relational schema, fixed communication protocols or system-dependent memory organisation.

- ***Semi- structured data***: Semi-structured data is getting very popular. For example, XML is a widely accepted and used data representation for semi structured data. The power of semi-structured data is its flexibility. In contrast to structured data, no schema is needed in advance to handle the data. The only concept is the entry, which can have any arbitrary content (even entries again) and certain characteristics (*e.g.* attributes in XML). In XML and some other frameworks for semi structured data these entries are organised in a hierarchy. Techniques for storage and retrieval of semi-structured data (esp. XML) are evolving rapidly but there are still limitations concerning this functionality because semi-structured data usually don't fit directly to underlying system hardware, structure or architecture.
- ***Free data structures***: The most flexible approach concerning the data format is the usage of free data structures without any given schema or restrictions. This leads to data management where all data is interpreted as binary objects. The most important disadvantage is the limited functionality in querying the data because no structures are given and therefore it is very difficult to implement powerful algorithms for searching certain data within a (large) collection of binary data.

Even if the different data representation approaches are of interest within applications and for internal component communication the data representation is important for interactions between a system and a system user. Therefore, the question of data representation is mainly an issue when defining system or component interfaces, which are published for further "external" use.

For better management of semi-structured data, most prominently XML, different approaches exist to restrict the complete freeness of data structure. Document Type Definitions (DTDs), in general, define a filter for semi-structured data and restrict data acceptance to specified classes of data. Multiple DTD specifications for various areas of application such as CML, MathML (all presented in [44]) or XMI [23] exist. A more detailed introduction XML is given in Section 4.1.5.3. More information about data, data management and representation can be found in [29,35,40].

4.1.3 SOFTWARE INTEGRATION ARCHITECTURES & STANDARS

This section introduces key concepts and issues related to software integration architectures and the need for industry standards to make these integration architectures practical and cost-effective. First, the industry need for software integration architecture is reviewed in the context of process engineering work processes. Then, the two major approaches to software integration architectures – loosely-coupled, asynchronous architecture and tightly-coupled synchronous architecture – are reviewed along with background information on the process industry standards efforts related to both styles of software integration.

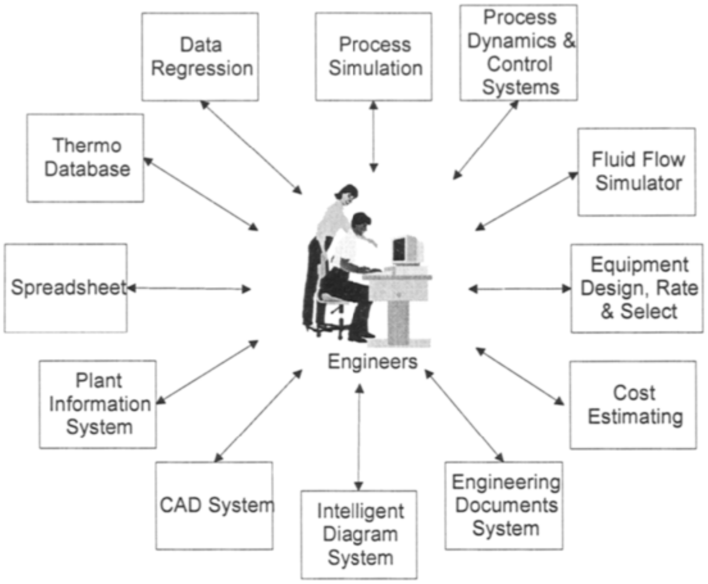


Figure 2. Traditional Software Integration

4.1.3.1 The industry need for software integration architectures

In the process industry, software applications have been built to solve specific problems associated with a highly complex technical work process involving many people, disciplines and companies. Typical examples from the process engineering problem domain are shown in Figure2 and are summarized in the next paragraph [5,38,49,50].

A data regression tool is used to evaluate thermodynamic experimental data obtained from a thermo database in order to construct an accurate thermodynamic prediction model to use in a process simulator, which is used to simulate steady-state process behaviour. Dynamic simulators simulate dynamic process response and evaluate control systems. Fluid flow simulators predict pressure drops in piping systems. Equipment design tools size and rate process equipment performance and produces mechanical designs. Cost estimating tools are used to estimate equipment capital costs. Various tools including spreadsheets and engineering databases are used to produce engineering documents such as equipment data sheets, equipment lists, line lists, utilities lists and various other engineering documents associated with design reports. Intelligent diagramming programs are used to produce engineering drawings such as process flow diagrams (PFD's), piping and instrumentation diagrams (P&ID's), isometric drawings and plot plan equipment layouts. Computer Aided Design (CAD) systems are used to built virtual 3D plant models. Plant information systems that are integrated with real-time process control systems and maintenance management systems are used to track, maintain and improve plant performance.

Of course, these are just the technical software applications. There are many other software applications used by engineers in a process industry enterprise, including office automation software, multiple technical and business database systems, *e.g.*, Enterprise Resource Planning (ERP) systems. To compound this already complicated software integration problem, the work process is a complex, team-oriented work activity with many people and disciplines spanning multiple enterprises, including owner-operators, technology licensors, engineering-procurement-construction (EPC) companies, and equipment suppliers. In general, these various companies who partner for particular projects, do not use the same software applications. Because of this inherently complex work process and the multi-company, multi-software vendor environment, software integration to support the process engineering work process has traditionally been accomplished by using people to do the software integration, as illustrated in Figure 3 [39].

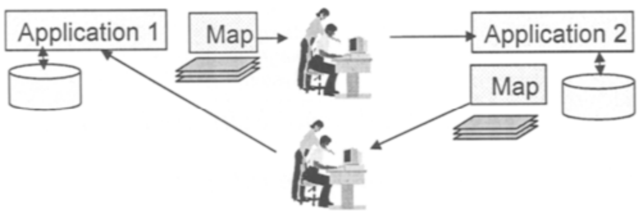


Figure 3. Manual Software Integration

Each application manages its own data files and interacts with users through normal interfaces (either displays or printed reports). Typically a subset of either common input or output data from Application 1 is needed as input to Application 2 or vice-versa. There is one overriding key requirement software integration: In translating data from one application to another, a translation table, or "map," is always needed to decide the meaning of data in one application and how that data maps onto data used in the other application. As we shall see, the cost of this data map is the key to achieving practical software integration architectures.

While labour-intensive, manual software integration is the primary approach to software integration today, the drawbacks are significant, including higher costs, longer schedules, and occasionally, lower quality due to inevitable human errors escape detection and correction. Practical and widespread automation approaches to software integration could have a dramatically beneficial impact on the process industry, estimated to be on the order of hundreds of millions to billions of dollars annually.

4.1.3.2 Loosely-coupled, or asynchronous approaches to software integration

The earliest (and still used) approach to software integration is to build point-to-point software interfaces from one application to another (Figure 4) [39].

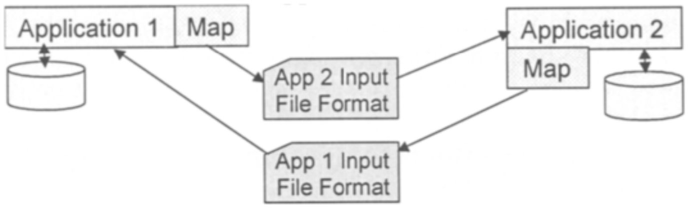


Figure 4. Point to Point Software Integration

In this approach, because input processing is usually more costly than producing specialized output reports, the data maps are often embedded into the application as a specialized output report to map the report-producing application's data onto the other application's input file format, or some easily parsed (computer-, but not human-readable) alternative input file format. Manual editing and merging of these special transfer files may still be needed for these interfaces to work. While this approach automates manual software integration, it is still a very expensive approach because custom software interfaces must be built for every application pair in the work process and maintained each time a new version of one of the applications is released. If you have more than just a few applications, the number of potential required interfaces required becomes very large, (N^2–N) interfaces, where N is the number of applications. Further, with this approach, each application owner is faced with developing and supporting N independent interfaces, and sometimes directly with competing software packages and vendors. In the process engineering domain, N is typically on the order of 10-30 or more applications across multiple software vendors, making point-to-point software integration impractical for process engineering.

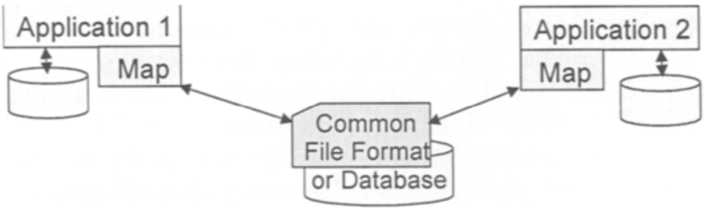


Figure 5. Common File Format and Database Integration

A major improvement on the point-to-point architecture is to use an industry-standard common data definition and file format, which may be either flat files (for paired application data exchanges) or databases (for multi-application data sharing) as illustrated in Figure5 [39]. The major advantage of this approach is reducing the potential number of application interfaces and associated maps from N^2–N to a much more manageable 2N interfaces. Furthermore, for any single application owner, support for only one interface mapping is required, regardless of how large N becomes. This is a major advantage for software owners.

However, a major challenge for this software integration architecture is obtaining consensus and agreement from the multiple companies and software vendors who own and use the applications to agree upon the common data definition and software implementation architecture to become a widely accepted industry standard.

4.1.3.3 Process industry efforts to develop asynchronous ISO standards

The ISO 10303 (STEP) standard [17] started in the 1980's and was largely driven by the need to exchange product information and detailed geometry and CAD information across multiple industries including the aerospace, automotive, electronics and, since the mid-1990's, for the process industry. In order to accommodate a multi-industry standard while still using a common data model and instantiation of data at the physical level, the STEP architecture employs a two-layer data mapping architecture, which is shown in Figure 6.

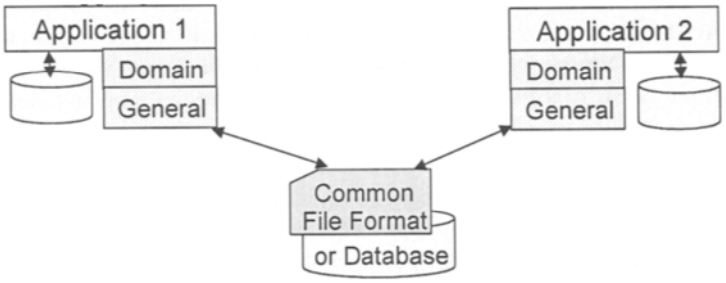


Figure 6. STEP Architecture for Software Integration

In the STEP approach, the application interface map consists of two-layers: (1) the application domain layer, which in STEP terminology is called the Application Resource Model (ARM) and (2) the general layer, which in STEP terminology is called the STEP Integrated Resources model. The physical instantiation of a STEP-compliant intermediate file or database is required to be compliant with the STEP Integrated Resources model so that physical instantiations will be independent of industry and problem domain. In each Application Protocol (AP) document, a document that describes each domain-specific part of the STEP standard, there is an explicit mapping of the domain (ARM) level data model to the general STEP Integrated Resources. The map from the ARM to the STEP Integrated Resources is called the Application Interpreted Model (AIM). The ARM and the AIM together comprise a STEP Application Protocol (AP).

Starting in the mid-1990's process industry consortia sponsored the development of three STEP Application Protocols:

- AP 221: Functional Data and Schematic Representation for Process Plant (sponsored by USPI-NL, PISTEP and EPISTLE in Europe, [3]. See Chapter 4.2 for more information)

- AP 227: Plant Spatial Configuration (sponsored by the US-based PlantSTEP consortium) [4]
- AP 231: Process Engineering Data: Process Design and Process Specifications for Major Process Equipment (sponsored by pdXi. See Chapter 4.2 for more information.)

By 2001, AP 221 and AP 231 have reached Committee Draft status, while AP 227 has reached International Standard status. More information about AP 221 is presented in 4.2, additional background information about AP 231 is included in Chapter 4.2.

There is an interesting contrast to note in the above three efforts. AP 227 and AP 231 followed a traditional STEP approach of developing explicit engineering domain object models using process and plant engineering terminology at the domain (ARM) level, and then mapping these to the STEP Integrated Resources model in the AIM. In contrast, AP 221 effort defined the plant engineering problem domain using another general abstract model (called the EPISTLE data model) at the domain (ARM) level, which then needed to be mapped to the general abstract STEP Integrated Resources level. To make software implementations possible, an instantiation of the AP 221 generic ARM model is required that maps engineering data from software applications. To meet this need, another modelling level was added on top of the ARM model, called the Reference Data Library (RDL). Therefore, to implement STEP AP 221 in software interfaces, three mapping layers are required – Application to RDL, RDL to EPISTLE/ARM, and EPISTLE/ARM to STEP Integrated Resources.

An alternative ISO standards effort, ISO 15926, [18] which is sponsored by the POSC-CAESAR consortium, emerged in the late 1990's in parallel to AP 221 to address the needs of data warehouses (databases) for oil and gas production facilities. ISO 15926 uses the generic, conceptual data model developed by EPISTLE [14] for AP 221 and, in cooperation with EPISTLE/AP 221 has developed an extensive Reference Data Library (RDL) to support the draft standard. Using the ISO 15926 standard allows the application interface maps to become similar to the STEP standards, where the RDL serves as the engineering domain level model and the EPISTLE/AP221 general model serves the same function as the STEP Integrated Resources model.

Both the STEP or ISO 15926 two-layered mapping approach to software integration make it possible to have multi-industry, multi-domain data exchange or data repository architectures using the same physical data instantiation in databases. This is an advantage for software owners, such as CAD vendors or vendors of data warehouse repositories, whose customers span multiple industry groups and the same basic database can be customized to different industry groups. One feature (some would say advantage, others would say disadvantage) of the AP221/EPISTLE/ISO15926 approach is that each time the standard is used, the two parties in the data exchange must first agree on the version of the RDL that they will use. This offers flexibility (advantage), but incurs extra cost to implement (disadvantage). One general disadvantage of the two-layered mapping approach to software integration, whether STEP or ISO 15926, is that, overall, it becomes very complex, difficult and costly for most software application owners, at least in the process industry, to understand and implement software application maps and interfaces. This high complexity and high cost of building software interfaces presents a high activation energy barrier for most software vendors to achieve fully STEP-compliant, or ISO 15926-compliant commercial software implementations in process industry software.

The ISO standards have been a long time in development and are generally expensive to implement in process industry software due to the high cost of building the application data maps to the highly abstracted general data models (STEP integrated resources model or EPISTLE model). This high implementation cost has been an inhibitor to their widespread use in commercial software.

4.1.3.4 Process industry efforts to develop asynchronous XML standards

The eXtensible Markup Language (XML) has recently emerged as an important Internet standard for electronic commerce and is introduced in more detail in Section 4.1.5.3. At this point, suffice it to say that XML is an Internet standard file format for the common data file format software integration architecture shown in Figure7.

However, by itself, XML is not sufficient to constitute a practical software integration architecture for the process industry. To achieve value and practical software integration architectures, the process industry still needs to reach consensus on a standard process industry XML vocabulary. Efforts in the process industry to develop XML standards have just begun to emerge. One of these efforts is aecXML [1]. aecXML was originally organized by Bentley Systems and now operates under the auspices of the International Alliance for Interoperability (IAI) [16]. The aecXML effort is largely oriented towards buildings and building construction projects, but does have a small Plant Working Group. The Plant Data XML effort [27], building on the previous ISO standards work on AP 231, is described further in Chapter 4.2.

4.1.3.5 Tightly-coupled or synchronous approaches to software integration

An alternative approach to software integration is needed when two applications need to execute synchronously in a collaborative way. This concept is illustrated in Figure7.

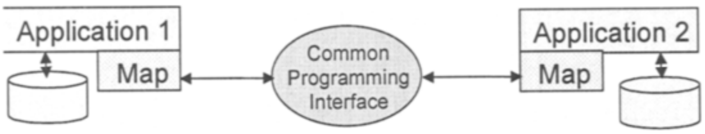


Figure 7. Tightly coupled synchronous integration

In contrast to the asynchronous, loosely coupled style of integration, there is no physical instantiation of data in separate files in synchronous integration. As with asynchronous integration approaches, each application remains responsible for its own data. In the synchronous architecture, one application program (client) requests services from the other application (server). The most common example of this style of software integration are the client-server architectures implemented in shared database systems, where Application 1 may be a problem-domain oriented program, perhaps written in Visual Basic or web client script, and Application 2 is a server-resident database program, which may be servicing many applications at the same time. Typically the client program makes a request and waits while the server program fulfils the request. In this synchronous architecture, as with the asynchronous architecture, it is possible for the applications to reside either on the same hardware or on different hardware.

However, the synchronous architecture need not be limited to "data only" clients and services. In the late 1990's and early 2000's, the process industry, through the CAPE-OPEN and Global CAPE-OPEN research consortia, has been developing synchronous data integration architectures for integrating *synchronous calculations* among separate applications through a standard programming interface called CAPE-OPEN (CO) [11]. For integrated calculations, synchronous integration approaches are required because the performance penalty of instantiating physical files in a loosely coupled approach is too severe to make loose coupling practical for this application. To summarize the approach in the context of software integration architectures, one application (*e.g.*, a "client" process simulator which supports CAPE-OPEN "socket" interfaces) calls a second application (for example, a third-party thermodynamic properties "server" which supports a CAPE-OPEN "plug" interface) in real-time to converge a process simulation model. In this way, user companies who have proprietary thermodynamic properties calculation or unit operation software, can write "plug" server modules and plug them into their preferred commercial process simulator, which provides the "socket" interface. Details about the CAPE-OPEN standard will be presented in section 4.3.3 and chapter 4.4

4.1.4 PROCESSES FOR DEVELOPING COMPONENT-BASED SOFTWARE

The alarming reality is that about one-half of the estimated 300,000 software projects in the United States in 2001 will fail due to poor planning, inexperienced project managers, or inept project teams. This section discusses the project management, method and process for building software system especially based on object-oriented components. Our objective here is only to introduce this subject and demonstrate to what extent this aspect is fundamental. This section addresses principally the processes.

Project management faces the challenges of producing top-quality software on time and within budget. As with many industries, managers in software development must keep their staff motivated, cost-justify their strategies, beat deadlines and balance budgets. But this becomes all the more challenging in an industry where terms, technologies and processes shift rapidly. To plan and execute a successful software project, some managers rely on a method.

A notation/language and a specific process define a method. Accompanying the success of the object development, we can count more than 50 object methods and we cannot find one rule, which is very formal and relevant to the different business domains. The UML language opened the unification field in merging the notations and in defining thoroughly the concepts. However, many organisations use the UML as a common language for their project artefacts, but they will use the same UML diagram types in the context of different processes. As a matter a fact, the UML is intentionally process independent, and defining a standard process was not a goal of the UML. The UML language is presented in Section 4.1.5.2.

Therefore, the remaining challenge is to unify the software development process. To specify one universal process would be very perilous. Indeed, experience has shown that different organisations and problem domains require different processes. For example, the development process for shrink-wrapped software is vastly different from building hard-real-time avionics systems upon which lives depend. Even in a single domain such as CAPE, to build a complete environment for process simulation and to build a thermodynamic library for electrolytes require different processes. In fact, the UML authors do not really unify the process but rather collect best practices of object-oriented software (component or not) development. In the future, we should find a family of processes coming from this work, grouped under the term Unified Process.

4.1.4.1 The unified process

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. The development for the World Wide Web, while making some things simpler, has exacerbated the problem. A well-defined process is a central element on the road of the success.

A process defines a set of partially ordered steps intended to reach a goal. In software engineering the goal is to build a software product or to enhance an existing one. A unified process (UP) [20] is a software development process that uses the UML language to represent models of the software system to be developed. It is iterative, architecture centric, use case driven and risk confronting. It derived primarily from the three market leading methods (Booch, Objectory and OMT) with ideas drawn from many other methods and input from many other parties. Ivar Jacobson, the principal developer of the UP, defines it as: "...a generic process framework that can be specialised for a very large class of software systems, for different application areas, different types of organisations, different competence levels, and different project sizes." [20]

Figure 8 summarises the overall structure of the UP. The process has two dimensions:

- The horizontal dimension represents time and shows the lifecycle aspects of the process (phases) as it unfolds.
- The vertical dimension represents core process disciplines (workflows), which logically group software engineering activities by their nature.

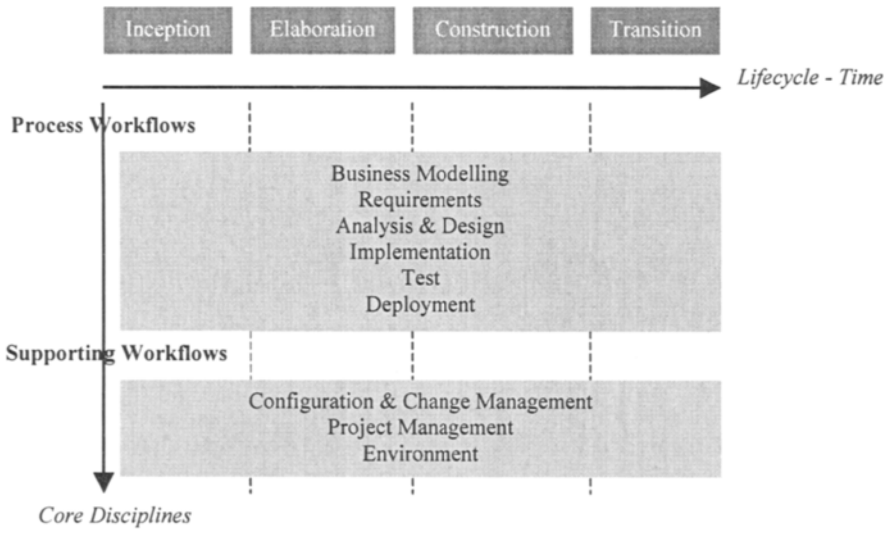


Figure 8. Structure of the unified process

The four phases are:

- Inception: defines the scope of the project and develop business case.
- Elaboration: Plan project, specify features, and baseline the architecture.
- Construction: Build the product.
- Transition: Transition the product to its users.

The horizontal dimension represents the dynamic aspect of the process expressed in terms of cycles, phases, iterations, and milestones. A software product is designed and built in a succession of incremental iterations. This allows testing and validation of design ideas, as well as risk mitigation, to occur earlier in the lifecycle. The vertical dimension represents the static aspect of the process described in terms of process components: activities, disciplines, artefacts, and roles.

The UP is a process framework that can be adapted and extended to suit the needs of an adopting organisation which can modify, adjust, and expand the process to accommodate the specific needs, characteristics, constraints, and history of its organisation, culture, and domain.

As main examples of unified process, we can cite the Two Tracks Unified Process (2TUP) [30] and the Rational Unified Process (RUP) [22]. The RUP, which is a commercial product from Rational, is defined as a more specific instance, or specialisation, of the more general UP. As with UML, the UP was created and promoted by Rational, but entered the public domain, so that now Rational is just one of many organisations with competence in its use.

It is worth noting that although there is nothing unique in the UP that can not be found in other modern development processes, it organises high-impact best-practices in a cohesive, thorough, and well-documented presentation. Nevertheless we should notice the critical review of the UP [2] that presents a survey of the alternate software processes, and synthesises a "more" robust process that addresses the complete breadth of real-world development and production needs.

4.1.4.2 CAPE-OPEN work process

The CAPE-OPEN work process is dedicated to the CAPE-OPEN standard and aims at delivering standard interface specifications and prototypes. In some way, this interface development process is a unified process even though it does not formalise clearly the different phases. It is iterative, relies on the object-oriented modelling and component technology, creates and manages a UML model, pays ongoing intensive attention to the definition and management of end-user requirements. It adapts the UP process workflows and consequently defines the analysis, design, specifications, implementation and test steps. More information can be found in section 4.3.3 and chapter 4.4.

4.1.5 SELECTED STATE-OF-THE-ART TECHNOLOGIES

This section provides an overview of approaches, technical solutions, and tools for dealing with the issues presented in the previous sections. Three middleware approaches (COM, CORBA, and Enterprise Java Beans) for synchronous software integration will be discussed and briefly compared. Subsequently, the Unified Modelling Language will be presented as a means for object-oriented systems design. An introduction to XML as a universal data interchange format as a means of asynchronous software integration and its practical application concludes this section.

4.1.5.1 Middleware solutions

The following section gives a short introduction to the three most popular middleware approaches: CORBA from the Object Management Group, COM from Microsoft, and the Java 2 Enterprise Edition (J2EE) technology from SUN. We also discuss their strengths and weaknesses and point out some middleware interoperability issues. The Cetus website [9] gives on-line resources for all technologies mentioned here. It also contains some tutorials on various issues.

CORBA

The Common Object Request Broker Architecture (CORBA) is an open middleware standard facilitating broad interoperability between object-oriented distributed components in highly heterogeneous environments [23]. It was developed by the Object Management Group (OMG), which represents a wide community of software vendors, developers, and users. The OMG specified the Object Management Architecture (OMA), which includes CORBA as a key part. It is important to know that the OMA and CORBA are just open specifications and no implementations. The implementation of these standards is left to the software vendors.

The OMA comprises an Object Model that defines how objects distributed across a heterogeneous environment can be described, and an architectural Reference Model that characterises the interaction between those objects. These models also include the definition of the Internet Inter-Orb Protocol (IIOP), which sets a standard for internet-based communication between software components for transferring messages and data. It also defines marshalling procedures on a binary level. This enables CORBA implementations from different software vendors to communicate with each other. The OMA Object Model defines an object as an encapsulated entity, which offers services that can only be accessed through well-defined interfaces. Clients send requests to objects to perform services on their behalf. An overview of the OMA Reference Model is given in Figure 9.

The central part of the OMA is the ORB, a kind of distributed object bus that is, as explained in Section 4.1.2.2, mainly responsible for facilitating communication between client and server objects through their interfaces. Several ORBs in the network from different vendors may communicate via IIOP, thereby maybe connecting different hardware platforms or operating systems. Additionally, several standardized services are part of the OMA, which are defined using the CORBA interface definition language (IDL). These services can be grouped in the following categories [43]:

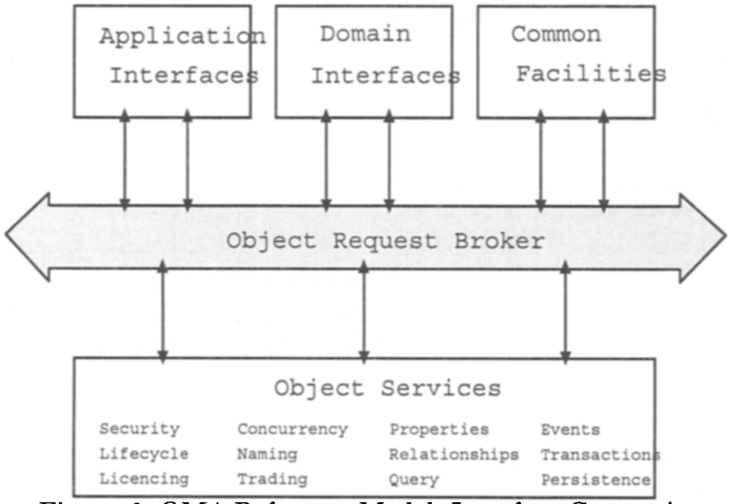


Figure 9. OMA Reference Model: Interface Categories

Object Services: These interfaces define the system-level object frameworks that extend the CORBA object bus. They are used by many application programs and provide services that are almost always necessary regardless of the application domain. Two examples are Naming Service, which allows to locate objects based on names, and the Trading Service -- a kind of yellow pages that enables to find objects based on their properties.

Common Facilities: Like Object Services these interfaces are horizontally oriented, i.e. independent of the application domain, but they are more oriented towards the end-user application. For example database services could be found here.

Domain Interfaces: These interfaces are vertically oriented and focus on specific application domains such as manufacturing, telecommunications, medical, and financial domains. These standard interfaces as well as the Common Facilities are designed by special task forces of the OMG.

Application Interfaces: These interfaces are specifically developed for applications and thus not standardized by the OMG.

The IDL used for defining these standard services and for all components developed outside the OMG is fully object-oriented and supports multiple inheritances of interfaces. To use these IDL definitions, the CORBA standard includes language bindings, which are translation rules from IDL to real programming languages. Therefore, it is possible to connect components developed in different programming languages using these standardized mappings and the other CORBA mechanisms. Additionally, the language bindings facilitate the development of IDL compilers, which can automatically generate all necessary code for the communication between the component implementation and the ORB. Currently, official standardized bindings are available for C, C++, COBOL, Java, Smalltalk, Ada, Python, and Lisp. There are also some bindings available for other languages (*e.g.* FORTRAN, Visual Basic) which have not been standardized by the OMG so far. Hybrid approaches can be used for integrating languages that no mapping exists for. For example, one could provide a C++ wrapper for a FORTRAN program and then use the wrapper to access CORBA.

Due to its strengths in the integration of software in heterogeneous environments and its robustness, CORBA has gained growing importance in large enterprise applications. It has proven useful for complex and mission critical applications and is therefore employed in many e-commerce applications as well as in the banking sector. The importance and maturity of CORBA is stressed by the fact that a wide range of stable and high-performance commercial (*e.g.* IONA Orbix, Borland VisiBroker) and non-commercial (*e.g.* OmniORB, Mico) CORBA implementations is currently available. There are also integrated tools in the market supporting rapid prototyping with CORBA (*e.g.* JBuilder, Delphi and C++ Builder from Borland)

COM

Microsoft's Component Object Model (COM) [47] and its distributed version DCOM are the foundation of Microsoft's compound document technology OLE (Object Linking and Embedding). In the beginning, OLE was a proprietary integration solution for Microsoft Office products, but now it has become very popular. Based on OLE there is another Microsoft component standard which has become very popular recently especially in the area of web-based applications: ActiveX. Many tools available today are based on these technologies, which allows them to integrate with each other seamlessly on the GUI level. Moreover, the next generation COM+ has been made available in Windows 2000, which offers advanced object services and targets the integration of systems developed in different programming languages.

In contrast to CORBA, COM is not an open specification but a binary interface standard that is directly connected to the Windows operating system. This makes

COM and its derivatives a platform dependent technology not very suitable for heterogeneous environments. However, there are some DCOM implementations available for other operating systems but these products have not gained much acceptance in the market so far. On the other hand, their strong relation to Windows has facilitated a quite efficient implementation for reasons that are explained below.

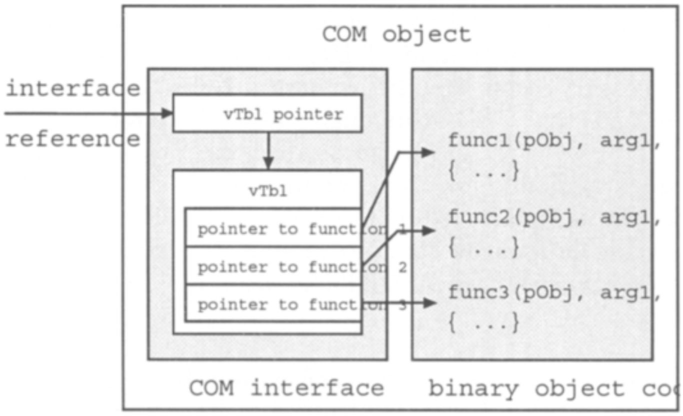


Figure 10: COM Objects and Interface References

The central idea of COM is to construct a binary interface (see Figure 10) using a data structure called virtual function table, or v-table (vTbl). This v-table contains pointers to the functions within the COM object. A reference to a COM interface implies that you have a pointer to a pointer to a v-table. Since COM provides no language mappings, the developer of a component is responsible for providing these tables. They match very closely with v-tables known from Visual C++, but it can be very difficult if not impossible to directly implement COM interfaces in other languages such as FORTRAN. However, Visual Basic, Visual C++, Visual J++, Delphi and others offer powerful tools that support COM such that less effort is required for these languages. Regarding FORTRAN, a hybrid can be built by wrapping FORTRAN with Visual C++ or Visual Basic. In this implementation model, there is no explicit piece of software handling all object requests as in CORBA. All the COM ORB functionality is built into the Windows operating system. Obviously, this kind of implementation is quite efficient but very proprietary.

On a higher level, the functionality of a COM component is described by the Microsoft IDL (MIDL). The MIDL whose syntax strongly resembles C++, is an object-oriented language which features single inheritance only. But in contrast to CORBA, a component is allowed to implement several interfaces, which can be used for simulating multiple inheritances.

Microsoft already defines many standard COM interfaces, which can be used to provide standard behaviour of application components. Most of them are closely related with OLE and define common services such as Structured Storage or compound document facilities. For application development it is necessary to create custom interfaces to define the services a component wants to provide.

In spite of some conceptual weaknesses, COM plays a very important role in purely Windows-based environments. Strong tool support is offered by various vendors hiding the inherent complexity of COM to a developer. Especially with Visual Basic, COM component development is very easy. No additional software is needed for using COM as it is shipped as part of the Windows operating system. But when choosing COM as a technology one should keep remember that there are rather limited options for interfacing non-Windows systems.

Java 2 Enterprise Edition

The Java 2 Enterprise Edition (J2EE) [36] is a platform designed for implementing distributed and web-enabled large-scale enterprise applications. The J2EE consists of a set of specifications for components and various component related services. Additionally, the J2EE contains a reference implementation of the specifications and a testing environment for application developers. Similar to CORBA, several commercial implementations of the specifications are available (*e.g.* Inprise Application Server, Bea Weblogic Server, IBM Websphere Application Server).

Part of the J2EE is the specification of the Enterprise Java Beans (EJB), which is Java-based component architecture. Other specifications, which are parts of the J2EE cover other areas of component-based applications such as naming and directory services (JNDI), and a basic infrastructure for inter-component-communication (RMI and RMI over IIOP). The J2EE goes one step beyond the COM and CORBA middleware architectures. It does not only offer a wide range services for handling components such as the basic communication infrastructure or object services but also contains a framework for creating dynamically extensible application servers into which EJB components can be plugged.

An application server is a piece of software, which can handle software components and their data. It is responsible for activating components, making them and their data persistent in a database, handling load balancing issues, transaction management, and many additional services. This dimension of working with components is only partially addressed in COM and CORBA even though there are some activities in that direction regarding COM+ and the latest CORBA 3.0 specification.

As J2EE and the EJBs are Java frameworks, all object-oriented concepts of Java are available for the EJB components. Therefore, all inheritance mechanisms are available and regular Java interface definitions are used for specifying components. But this Java focus is also a drawback. The only language that can be used for implementing J2EE components is Java itself. In contrast to COM or CORBA, it is not possible to integrate components written in different languages directly. Fortunately, the J2EE framework integrates with CORBA quite well thereby opening up itself to everything that is available on a CORBA basis. Additionally, the J2EE offers connections to COM exposing EJBs to the COM world and connector interfaces to existing legacy systems.

Middleware interoperability

As discussed above, COM, CORBA and J2EE have their specific strengths and weaknesses. COM is a good choice in a purely Windows-based non-distributed environment. It offers good tool support for different programming languages and is efficient. But it is not as clearly structured and as well designed from an object-oriented standpoint as the two other approaches. CORBA has its strengths in the integration of very heterogeneous IT-landscapes as it is available for most operating systems and many programming languages. It is clearly designed and can be used for mission-critical enterprise systems. On the other hand, it does not integrate as seamlessly as COM into the Windows environment, making it less runtime efficient. The J2EE combines the advantages of CORBA with additional support for many component services needed in an enterprise environment such as transaction handling or persistence of components and data. Web-based systems are also directly supported. But although there are connections to COM and CORBA, J2EE is strongly focussed on Java as the implementation language, which is unsuitable in domains where just raw computational power is required (*i.e.* number crunching).

For CAPE, the best solution seems to be a hybrid strategy, which combines the best of all worlds. To reach such a solution, so-called bridging software is needed exposing a component written for a specific middleware system to another system (*e.g.* make a CORBA component available to COM or *vice versa*). As mentioned above, CORBA and J2EE integrate very well and there is no additional software needed. On a conceptual level, the integration of these systems is also manageable because the principles do not differ widely. COM integration to J2EE or CORBA is not that easy because the systems differ conceptually. There are some products available (*e.g.* Iona Orbix COMet) which offer bridging capabilities for that area. COM-CORBA bridging is tackled extensively in [15]. The J2EE already includes specifications that define bridges to COM.

4.1.5.2 The Unified Modelling Language (UML)

Developing a model for an industrial-strength software system prior to its construction or renovation is as essential as having a blueprint for a large building. Good models are crucial for communication among project teams and to assure architectural soundness. As the complexity of systems increases, so does the importance of good modelling techniques. There are many additional factors of a project's success, but having a rigorous modelling language standard is one essential factor.

In the face of increasingly complex systems, visualisation and modelling become essential. The UML is a widely accepted response to that need. It is the visual modelling language of choice for building object-oriented and component-based systems. A vast amount of literature is now available on this language; we only cite here the books by the UML authors [31, 32].

Introduction to the UML

The UML is a graphical language for visualising, specifying, constructing, and documenting the artefacts of a software-intensive system. The UML represents a collection of best engineering practices in the modelling of large and complex systems. The UML focuses on a standard modelling language, not a standard process. Although the UML must be applied in the context of a process, experience has shown that different organisations and problem domains require different processes. Therefore, the efforts concentrate on a common meta-model (which unifies semantics) and on a common notation (which provides a human rendering of this semantics). However, the UML authors promote a development process that is use-case driven, architecture centric, iterative and incremental. The main goals in the design of the UML were as follows:

- Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models.
- Provide extensibility and specialisation mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modelling language.
- Encourage the growth of the object-oriented tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

Prior to the UML, there was no clearly leading modelling language. Users had to choose from many similar modelling languages with minor differences in overall expressive power. Most of them shared a set of commonly accepted concepts that were expressed slightly differently in these notations. These differences did not greatly expand the power of modelling, but threatened to fragment the object-oriented industry, and sometimes discouraged new users from learning visual modelling. Users longed for the industry to adopt one broadly supported modelling language suitable for general-purpose usage. They wanted a lingua franca for modelling. Thus UML is the result of collaboration and merging of three major object-oriented methodologies, namely the Booch method, OMT and the OOSE.

This unification started in 1994 and the OMG (Objects Management Group) acceptance was reached in 1997. Now, the open OMG process manages the UML that is becoming a worldwide standard for meta modelling and notation. The current version of this de-facto standard is 1.3 [23]. The complete UML version 1.3 specifications is provided as a single downloadable file and contains UML summary, UML semantics, UML notation guide, UML extensions, UML CORBA facility interface facility, UML XMI DTD specification and Object Constraint Language specification. For more information on what is XMI DTD, see Section 0 XML Tools. The following companies submitted or supported this OMG technology adoption: Rational Software, Microsoft, Hewlett-Packard, Oracle, Sterling Software, MCI Systemhouse, Unisys, ICON Computing, IntelliCorp, i-Logix, IBM, ObjecTime, Platinum Technology, Ptech,Taskon, Reich Technologies and Softeam.

UML does not guarantee project success but it does improve many things. For example, it significantly lowers the perpetual cost of training and retooling when changing between projects or organisations. It provides the opportunity for new integration between tools, processes and domains. But most importantly, it enables developers to focus on delivering business value and provides them a paradigm to accomplish this.

Logical model of the UML

The architecture of the UML is based on a four-layer meta-model structure, which consists of the following layers:

- user objects,
- model,
- meta-model,
- meta-meta-model.

This section is primarily concerned with the meta-model layer, which is an instance of the meta-meta-model layer. For example, Class in the meta-model is an instance of MetaClass in the meta-meta-model. The primary responsibility of the meta-model layer is to define a language for specifying models. Meta-models are typically more elaborate than the meta-meta-models that describe them, especially when they define dynamic semantics. Examples of meta-objects in the meta-modelling layer are: Class, Attribute, Operation, and Component. The complexity of the UML meta-model is managed by organising it into logical packages. These packages group meta-classes that show strong cohesion with each other and loose coupling with meta-classes in other packages. The meta-model is decomposed into the top-level packages Foundation, Behavioural

Elements and Model Management. Note that the Foundation and Behavioural Elements packages are further decomposed.

Overview of the UML

Without respecting the formal structure of the three top-level packages mentioned in the previous section, we only introduce four concepts: the elements, the relationships, the extensibility mechanisms and the diagrams.

Modelling elements: The elements are fundamental abstractions of a model. We can identify four types of elements in UML:

- Structural elements that enclose class (see Figure 11), interface, collaboration, use-case, active class, component and node.
- Behavioural elements that enclose interaction and state machine.
- Grouping elements that enclose package
- Other elements that enclose notes.

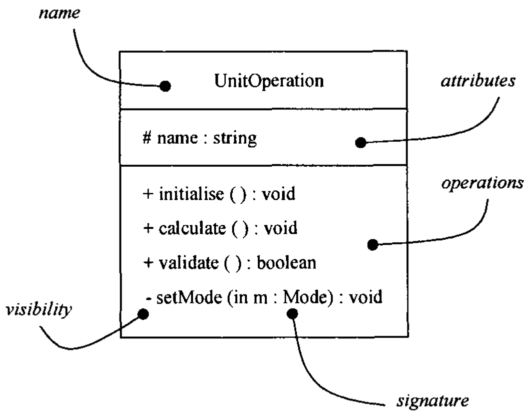


Figure 11. Class

Relationships: The relationships form the glue between the modelling elements. There are four types of relationships in UML: dependency, association (see Figure 12), generalisation and realisation.

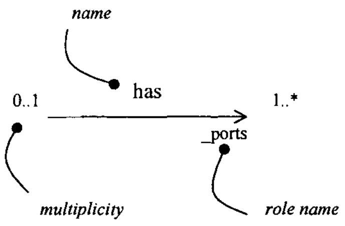


Figure 12. Association

Extensibility mechanisms: The UML provides a rich set of modelling concepts and notations that have been carefully designed to meet the needs of typical software modelling projects. However, users may sometimes require additional features and/or notations beyond those defined in the UML standard. Extensibility mechanisms allow model elements to be customised and extended through stereotypes, tagged values and constraints.

Diagrams: A diagram is a view into a model presented from the aspect of a particular stakeholder. It provides a partial representation of the system and is semantically consistent with other views. In the UML, there are nine diagrams as illustrated in the next figure (Figure 13).

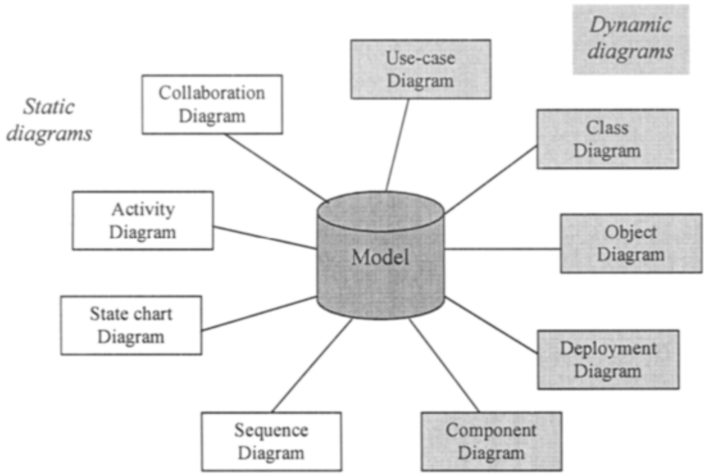


Figure 13. Model, View and Diagram

You can graphically depict an overview of the behaviour of your system with a use-case diagram. The collaboration diagram shows object interactions organised around objects and their links to one another. The state chart diagram provides additional analysis techniques for classes with significant dynamic behaviour. A state chart diagram shows the life history of a given class, the events that cause a transition from one state to another, and the actions that result from a state change. Activity diagrams provide a way to model a class operation or the workflow of a business process.

The system's logical architecture is captured in class diagrams that contain the classes and relationships that represent the key abstractions of the system under development. The component architecture is captured in component diagrams that focus on the actual software module organisation within the development environment. The deployment architecture is captured in deployment diagrams that map software to processing nodes, showing the configuration of run-time processing elements and their software processes.

You can view examples of UML diagrams in other parts of this book especially chapter 4.4.

Visual modelling tools

Visual modelling is the mapping of real world processes of a system to a graphical representation. Standardising a language such as UML is a necessary foundation for tools. Many visual modelling tools supporting the UML standard are available on the market. You can find from [9] the object-oriented analysis and design tools web page provided by the OMG. From this page you can access a list of products. These tools allow the visual modelling using UML language through object-oriented analysis and design. One interesting feature is that they facilitate code-generation from object models and vice-versa. This forward and reverse engineering is intended to allow software engineers to keep their designs up to date even when they are at the implementation phase.

Below, we provide some basic information on two specific tools; please note that the authors of this section do not recommend any modelling tool. Their features are given only in order to illustrate what these kinds of products are capable of.

Rose from Rational: Rational Rose is a visual modelling software solution that lets you create, analyse, design, view, modify, and manipulate components. You can graphically depict the system through the use-case analysis and object-oriented modelling. The Rose family includes Model, Professional, Enterprise and Real Time versions. It provides the following features to facilitate the analysis, design, and iterative construction of applications: user-configurable support for UML, COM, OMT, and Booch'93; semantic checking; support for controlled iterative development; round-trip engineering (code generation and reverse engineering); parallel multi-user development; documentation generation; integration with data modelling tools; scripting for integration and extensibility.

The add-in feature allows customising the product environment depending on your development needs. There are two types of add-ins, non-language (version control, web publisher, XMI, ..) and language (C++, CORBA, Java, Visual Basic, Ada, ..).

Objecteering from Softeam: Objecteering is a sophisticated object-modelling workshop, offering significant modelling support especially through the nine UML diagrams. It guarantees model consistency and has considerable capacity for generating code. The different versions are Personal, Open, Enterprise and Project edition. A university package is available. In the Enterprise version it is a multi-project and multi-user services tool. This tool offers advanced generation, modelling services and is destined for team application development. It allows the documentation generation and supports model exchange through XMI. It provides a universal model element identification mechanism and interbase model exchange services, as well as group work functions. This version also allows workshop parameterisation, both at the UML level and at the generation, model transformation and model request or operation levels, due to its specific UML Profile Builder module. Objecteering/Enterprise is an expandable tool. These modules, such as Java, C++ or SQL and CORBA , can be supplied and added at any time to any Enterprise site.

CAPE-OPEN standard and UML

Models are useful for understanding problems, communicating with everyone involved with the CAPE-OPEN (CO) development, modelling complex systems, preparing documentation, and designing CO interfaces. Modelling promotes better understanding of requirements, cleaner designs, and more maintainable systems. The CO standard adopts the UML and uses extensively its capabilities. In chapter 4.4 you can find an introduction to the CO Standard and its use of UML. Particularly the following concepts are employed:

- Use-case: It is a description of one particular usage of a part of the system. It captures the CO system functionalities as seen by users. A complete analysis requires many different use-cases, for example there are close to 30 use-cases for the CO unit operation interface alone. 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. It is a specific generic description of a use of the system. The use-case is attempting to capture the functional interactions rather than the physical appearance of the system.
- Sequence diagram: It shows a specific scenario through the sequence of events in a use-case. It captures time-oriented dynamic behaviour. 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.
- Interface diagram: It 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 is a collection of possible uses in order to specify through its operations the service of a class. The functions in an interface can be implemented by many different objects or no objects at all, just normal subroutines. In the CO context these interface definitions capture the vocabulary of the CO software standard.
-

4.1.5.3 The eXtensible Markup Language (XML)

The eXtensible Markup Language (XML) emerged as a key enabling technology and Internet standard for electronic commerce around the turn of the millennium. Similar to HyperText Markup Language (HTML), XML is an Internet standard published by the World Wide Web Consortium (W3C) [44] and is derived from the Standard Generalized Markup Language (SGML), which became the ISO 8879 international standard [19] in 1986. HTML, XML and SGML are all text-based "tag" languages where information is enclosed inside the markup tags. For example, <b>Bolded Text</b> is a markup language representation of **Bolded Text**. The primary advantage of the text-based approach of these languages is the ability of these languages to work on virtually any computer hardware/operating system platform and virtually any application program. This computing platform neutrality has been a major factor contributing to the success of HTML on the Internet.

SGML has been around since the mid-1980's, and its application in commercial applications has been successful, but because of its complexity it has been relatively limited in scope and impact.

In the 1990's, HTML, a very small subset of SGML that is primarily oriented to describing text formatting and display, was used to build most of the Internet that exists today. HTML has the key advantage of being very simple to use with a very limited set of markup tags, which provide adequately powerful formatting capabilities especially with Cascading Style Sheets (CSS) [44]. As a result, it has been very widely used in today's Internet with millions of web sites and hundreds of millions to billions of web pages.

However, the same simplicity that makes HTML ideal for quickly developing web site content, limits its use as an intelligent software development platform. HTML pages are "dumb" in the sense that there is no mechanism to describe the semantics of the information contained on these millions of web sites. There is no way to intelligently handle data. Therefore, HTML is unsuited for "intelligent" document/data exchange over the Internet.

In the late 1990's XML was standardized as a larger subset of SGML, chosen to streamline SGML usage in the Internet environment. The key feature XML offers beyond HTML is semantic information embedded in text files through customisable markup tags. Because of this ability, XML will be the key enabling technology to build the next generation "intelligent" Internet. Consider the following simple example contrasting HTML and XML:

<u>In HTML</u>:

```
<p>Centrifugal pump
<br>Model P280
<br>ABC Pumps
<br>$3,480
```

<u>In XML</u>:

```
<pump>
   <type>Centrifugal</type>
   <model>Model P280 </model>
   <supplier>ABC Pumps </supplier>
   <price>$3,480 </price>
</pump>
```

The HTML version offers a way to format and display the information. The XML tags can also be used for formatting and display, but offer the additional feature that inexpensive software can be written to make intelligent use of the semantic information contained in the same text data.

Because of this inherent flexibility for describing data, the next incarnation of the Internet, based on XML, will provide much more intelligent web sites and agents than are possible in today's Internet using HTML. XML will enable intelligent business-to-business (B2B) e-commerce, accommodating common things such as electronic catalogues and purchase orders between companies, but also more complex things such as equipment data sheets, which can be used for the complex e-procurement processes in the process industry. XML usage conventions such as ebXML [13] and BizTalk [7] will enable applications to send intelligent messages (requests and replies) across the Internet. XML enables a vendor-neutral, software neutral, version-neutral ways of storing information. This will enable data archiving spanning many years and software versions. Summarizing, XML offers a technology that allows cross-platform, synchronous or asynchronous application integration across the Internet and across firewalls and data archival across many years of time and software versions. This opens up an expansive range of opportunities for technical software integration, e-commerce, business to business collaboration and long-term archival of technical data.

Practical application of XML – the need for layers on top of XML

By itself, XML is just a file syntax for developing platform-neutral text files. It is possible to construct "well-formed" XML files just by complying with the syntax rules, e.g., the text information is enclosed between beginning and ending tags such as <pump> </pump>.

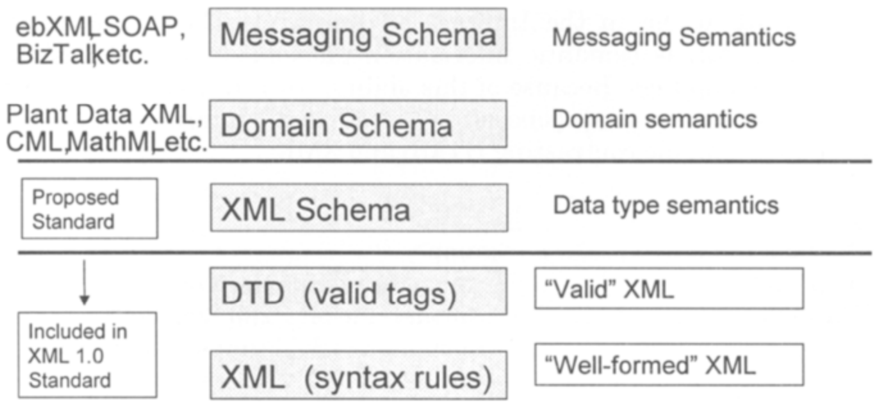


Figure 13. XML Usage Convention Layers

In order to be more useful, several layers need to be added to XML and eventually standardized, as shown in Figure13.

In the first layer above the base XML syntax, it is better to construct "valid" XML files. Valid XML files conform to a Document Type Definition (DTD). A DTD is simply a means to tell an XML parser what the valid tag names are in a given document. For example a DTD may specify that <pump> is a valid tag, but that <PUMP> is not. Valid XML files are therefore self-describing files, because the markup tags they conform to in a valid XML file conforms to the set of rules defined in the included DTD.

Even with DTD's it is possible for everyone to define their own DTD files and construct valid XML files. Therefore, additional layers are needed to make XML a technology for software integration. For example, it would be helpful to have a standard way of describing standard data structures and data types, such as integer, real, text, date, *etc.* XML Schema is an emerging standard [44] to describe data types in standard ways.

With XML Schema, we have reached the point where describing data with rich data type semantics and data structures is possible, but we still have complete freedom to define the tag labels themselves. For example, someone may choose to define an XML Schema or DTD tags in German, while another may construct them in French and still another in English. Even in a single language such as English, it would be possible to use different tag labels to mean the same thing, for example, <T>, <Temp>, <Temperature> and <temperature> are all acceptable and valid tags for temperature. To get the most value out of XML, the process industry needs a standard set of internationally recognized standard XML tags to describe process engineering data. This is the focus of the Plant Data XML effort, further described in chapter 4.2. Other domain-specific XML vocabularies include Chemical Markup Language (CML) [10], Mathematics Markup Language (MathML) [44] and XML Metadata Interchange (XMI) sponsored by the Object Management Group [23]. Each of these domain-specific vocabularies are needed to facilitate industry-wide integration of software applications using common terminology.

Finally, to enable cross-platform, application integration across the Internet, standard messaging protocols need to be developed. ebXML [13], organized under OASIS [26] is one such example. BizTalk [7] and SOAP [44], organized by Microsoft are other examples. ebXML has recently adopted SOAP has part of their standard, so convergence of these messaging standards is likely over the next year or so.

There are many other technologies related to XML, including XML Stylesheet Language (XSL) to enable translation from one XML schema to another and to display of XML files in browsers. Other XML technologies include XLink, XPointer, etc. Please refer to the World Wide Web Consortium (W3C) web site [44] and the OASIS [26] web site for more details about this fast-moving technology development. In addition, there are many good books available on XML.

XML Tools

There are many tools available for working with XML – far too numerous for any sort of exhaustive review here. Some of these tools are free. Others are commercial products. Rather than list them and describe them here, we will just list the types of tools that are available and web site resources to learn more about them. The types of tools that are available include XML Browsers, XML Editors, XML Parsers, XML/DTD Schema editors, XML serialization tools, XSLT editors, XSL formatters, XML and XSL utilities. Some resources for finding out more about these tools include [45] and [46].

4.1.6 REFERENCES

[1] aecXML web site : www.aecXML.org
[2] S. W. Ambler, L. L. Constantine, The unified process construction phase, Mc Graw Hill, 2000
[3] AP 221 web site : www.stepcom.ncl.ac.uk/epistle/ap221/ap221.htm
[4] AP 227 web site http://cic.nist.gov/plantstep/plantstp/ap227/ap227.htm
[5] J. T. Baldwin, T. L. Teague, and W. D. Witherell, "Info Transfer: An Emerging ChE Discipline," AIChE CAST Newsletter, Summer, 1998
[6] P. Bernstein, Middleware: A Model for Distributed Services. Communications of the ACM 39, 2 (February 1996): 86-97.
[7] BizTalk web site : www.biztalk.org/home/default.asp
[8] M. Brodie, M. Stonebraker, Migrating Legacy Systems. Gateways, Interfaces & the Incremental Approach, Morgan Kaufmann, 1995
[9] Cetus web site, www.cetus-links.org
[10] Chemical Markup Language (CML) web site : www.xml-cml.org
[11] CO-LaN web site www.colan.org, 2001
[12] D. D'Souza, A.C. Wills, Objects Components and Frameworks with UML: The Catalysis Approach, Addison Wesley, Reading, MA, 1999
[13] ebXML web site : www.ebxml.org
[14] EPISTLE web site : www.stepcom.ncl.ac.uk/epistle/epistle.htm
[15] R. Geraghty et al., COM-CORBA Interoperability, Prentice Hall, 1999
[16] International Alliance for Interoperability (IAI) web site : http://iaiweb.lbl.gov/
[17] ISO 10303 – Standard for the Exchange of Product model data, web site: www.nist.gov/sc4/www/stepdocs.htm
[18] ISO 15926 web site : www.nist.gov/sc4/www/oil_gas.htm

[19] ISO Standards Catalog (purchase) : http://www.iso.ch/cate/d16387.html
[20] I. Jacobson, G. Booch, J. Rumbaugh, The unified software development process, Addison & Wesley, 1999
[21] J.H. Kingston, Algorithms and Data Structures : Design, Correctness, Analysis (International Computer Science Series) Addison-Wesley Pub Co, 2nd edition (November 1997)
[22] P. Kruchten, The Rational unified process: An introduction, Addison & Wesley, 2000
[23] OMG web site: www.omg.org, 2001
[24] Orfali, Harkey, Edwards, The Essential Distributed Objects Survival Guide,Wiley, 1996
[25] Orfali, Harkey, Edwards, Client/Server Survival Guide, third edition, Wiley 1999
[26] Organization for the Advancement of Structured Information Standards (OASIS) web site: www.oasis-open.org
[27] Plant Data XML web site : www.plantdataxml.org
[28] R. Richardson et al., A Survey of Research into Legacy system Migration, External Technical reports, Trinity College Dublin Computer Science Department, 1997
[29] Rauh, Stickel, Konzeptuelle Datenmodellierung (in German), Teubner 1997
[30] P. Roques, F. Vallée, UML en action, Eyrolles, 2000
[31] J. Rumbaugh, I. Jacobson, G. Booch, Unified modelling language user guide, Addison & Wesley, 1998
[32] J. Rumbaugh, I. Jacobson, G. Booch, Unified modelling language reference manual, Addison & Wesley, 1999
[33] D. Serain, Middleware, Springer, 1998
[34] A. Solvberg, D.C. Kung, Information Systems Engineering, Springer 1993
[35] T.A. Standish, Data Structures, Algorithms, and Software Principles, Addison-Wesley Pub Co
[36] Sun Miscrosystems web site www.java.sun.com, 2001
[37] C. Szyperski, Component Software: Beyond Object-Oriented Programming, Addison Wesley Longman Ltd. 1998
[38] T. L. Teague and J. T. Baldwin, "The Emerging Discipline of Info Transfer," Proceedings of the Foundations of Computer Aided Process Design Conference (FOCAPD99), July, 1999
[39] T. L. Teague, "Using XML for Electronic Process Data Exchange," presentation to the South Texas Section CAST meeting, December, 2000
[40]T.J. Teorey, Database Modeling & Design: the fundamental principles, Morgan Kaufmann Publisher 1994
[41] B. Thalheim, Entity-Relationship Modeling : Foundations of Database Technology, Springer Verlag
[42] A. Umar, Application (Re)Engineering : Building Web-Based Applications and Dealing With Legacies, Prentice Hall, 1997

[43] S. Vinoski CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. IEEE Communication Magazine, Vol. 14, No. 2, 1997
[44] World Wide Web Consortium (W3C) web site www.w3c.org
[45] XML commercial resources site : www.xml.com
[46] XML commercial resources site : www.xmlsoftware.com
[47] Microsoft COM web-site www.microsoft.com/com, 2001
[48] J.W. Moore, Software Engineering Standards, IEEE Computer Society
[49] W. Marquardt, M. Nagl, "Tool Integration via Interface Standardisation? ", Dechema Monographie 135, 95-128, Weinheim: VCH, 1998
[50] M. Nagl, B. Westfechtl (Hrsg.), „Integration von Entwicklungsumgebungen in Ingenieuranwendungen – Substantielle Verbesserung der Entwicklungsprozesse", Berlin, Springer, 1999 (in German)

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 4.2: PlantData XML

T. Teague

4.2.1 INTRODUCTION

PlantData XML is an effort originally conceived in 1999, with work now starting in 2001, to combine existing process industry international data exchange standards and eXtensible Markup Language (XML) Internet standards to create a practical and easy to use electronic data exchange standard for the process industry. The PlantData XML standards, similar to many XML standards, are freely available to encourage widespread adoption and use in the process industry. The intention of creating PlantData XML standards is to enable both synchronous and asynchronous software integration architectures to be built that work both within and across companies in a vendor-neutral way.

The key ideas behind PlantData XML are to (1) develop a process-industry specific XML markup tag vocabulary that leverages and unlocks the value of the previously developed process industry ISO 10303 STEP standards by making these standards easier and less expensive to use and to (2) deploy the PlantData XML vocabulary in commercial software implementations to ensure that value can be obtained from the industry standard at a reasonable cost.

The key benefits of PlantData XML to the process industry include:

- Significantly improving the cost-effectiveness of engineering work processes through *vendor-neutral* electronic data exchange.
- Providing *vendor-neutral* technical data archival and reuse over the plant life cycle.
- Enabling more effective procurement and business-to-business (B2B) e-commerce in the process industry.

In this chapter, we first elaborate further the motivations for developing PlantData XML in the context of software integration architectures and previous standards work, which were described in more detail Chapter 4.1, section 4.1.3. Next, we discuss a software integration architecture using XML. In that section, we further elaborate the introductory material on XML presented in Section 4.1.5.3 to discuss the parts of XML software integration architecture related to creating application interfaces and use of XML parsing tools. We next review the potential scope and utility of PlantData XML using common industry work processes. Then, we describe a software development methodology for creating PlantData XML that adopts the iterative, incremental unified software development model advocated by Jacobson, Booch and Rumbaugh, and following this, we present a worked out initial example of PlantData XML. We conclude the chapter by discussing the initial application of PlantData XML in commercial software packages, which is anticipated to occur in the second half of 2001.

4.2.2 MOTIVATION FOR DEVELOPING PlantData XML

In Chapter 4.1, section 4.1.3 the general approaches to software integration architectures were reviewed and the process industry efforts to develop process industry standards were summarized for both synchronous and asynchronous software integration architectures. The initial focus for PlantData XML is to develop a standard to accomplish widely and easily implemented, asynchronous electronic data exchange among software applications. Potential other future uses for PlantData XML are discussed in section 4.3.1.8.

As illustrated previously in section 4.1.3 (see Figure 4.1.3.1), technical software applications in the process engineering domain include (at least) the following types of software applications:

1. Thermo Database
2. Data Regression (Thermo model parameter fitting)
3. Process Simulation (steady-state & dynamic)
4. Fluid flow hydraulics
5. Equipment design and rating
6. Detailed mechanical design
7. Cost estimating
8. Engineering Documents (*e.g.*, equipment data sheets, summary lists, *etc.*)
9. Intelligent Diagramming (PFD's, P&ID's, isometrics, layouts, *etc.*)
10. CAD models (3-dimensional geometry, visualization, *etc.*)
11. Plant Information Systems (operational and market data)
12. Maintenance Management Systems
13. Supply Chain Management Systems
14. ERP systems
15. Spreadsheets (used for many special purposes)

The engineering work processes that employ the above tools is a complex one that involves team efforts across multiple companies – owner operator companies, engineering-procurement-construction (EPC) companies and process industry supplier companies such as equipment manufacturers. Particular combinations of multiple companies collaborate for a given project, and then new sets of project team companies are assembled for other projects. In general, the various different companies use a mixture of commercial and internal software application programs, and, in general, the tools that are used in one company are not the same as those used by the companies that come together for a particular project.

Unfortunately, because the multiple companies involved use different software packages, labor-intensive, manual software integration is the primary approach to software integration today and the drawbacks are significant, including higher costs, longer schedules, and occasionally, lower quality due to inevitable human errors escape detection and correction. Practical and widespread automation approaches to software integration could have a dramatically beneficial impact on the process industry, estimated to be on the order of hundreds of millions to billions of dollars annually.

Both the STEP architecture and the ISO 15926 standard employ a two-layer data mapping architecture, which is shown in Figure 4.3.1-1. The application interface map consists of two-layers (1) the application *domain* layer and (2) the *general* abstract data model layer. In the case of STEP, this two-layer architecture was developed to enable multiple application domains to share a single physical implementation format. In the case of ISO 15926, the two-layer architecture was developed to allow high flexibility in defining the domain layer. *One general disadvantage* of the two-layered mapping approach to software integration, whether STEP or ISO 15926, is that, overall, it is a very complex, difficult and costly approach for most software application owners in the process industry to understand and implement software application maps and interfaces. This is due largely to the highly specialized knowledge of the general data model and the custom mapping that is required from the domain level to the general layer. Only a few hundred people worldwide have the expertise to build software implementations in STEP. This high complexity and specialized nature of the mapping interface results in a high cost to build software interfaces. For example, building one application interface for this style of architecture for the STEP AP 231 standard was estimated in 1999 to cost between 0.5 to 1 million US Dollars. The cost of building application maps is therefore a prohibitively high activation energy barrier for most software vendors to achieve current standards-compliant commercial software implementations in process industry software.

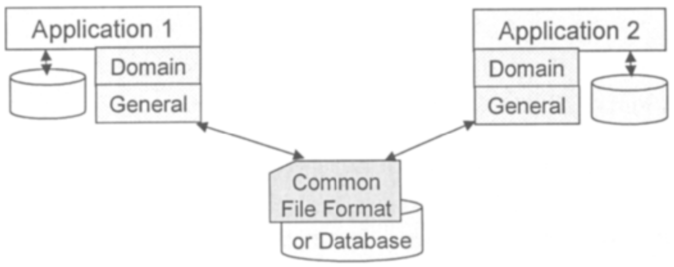


Figure 4.3.1-1 STEP Architecture for software integration

As shown in Figure 4.3.1-2 the PlantData XML standard uses a single layer mapping approach to a standard format XML file. The XML file employs a data schema that directly uses familiar process engineering terminology. Specifically, PlantData XML starts with the domain level data model from the AP 231 STEP standard and uses it to define a process industry XML vocabulary. Further, since the PlantData XML standard is being developed with software vendors and users to meet the needs of commercial software applications, practical validation of the XML vocabulary is achieved, where practical implementation of the AP 231 standard itself was never demonstrated in commercial software. The cost of building application maps using PlantData XML is one to two orders of magnitude less expensive than building a STEP implementation, making standards-based electronic data exchange practical and affordable.

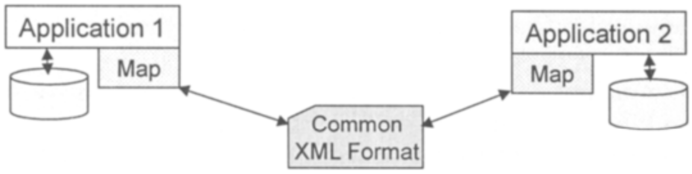


Figure 4.3.1-2 PlantData XML file format integration

4.2.3 XML SOFTWARE INTEGRATION ARCHITECTURE

A foundation to effective industry-wide software integration is the adoption of a Process Industry vocabulary as described in section 4.1.5.3. But equally important are the software components that make XML based software integration a practical reality.

To integrate applications XML interfaces must enable two data flows as shown in Figure 4.3.1-3:

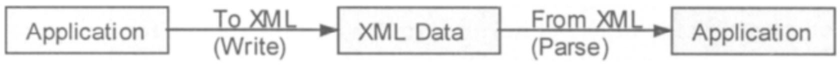


Figure 4.3.1-3 XML interface data flows

The "To XML" step, consisting of writing text in a format specified in a DTD or schema, is easy. The "From XML" step, consisting of parsing the XML text, is much more complex. There are currently two popular Application Programming Interfaces (APIs) that support the "From XML" process and to some extend, the "To XML" process. These are the Document Object Model (DOM) [1] and the Simple API for XML (SAX) [2].

DOM is an interface description being developed by the World Wide Web Consortium (W3C) of which Level 1 is currently a recommended standard. It can be implemented on any platform in any programming language. DOM presents XML data as a hierarchical tree of nodes based on the structure of the XML. SAX is an interface for event-based XML parsing and presents the XML data as a stream of events.

Both interfaces have their strengths and weaknesses. DOM is particularly appropriate when access and manipulation of all the data is required. DOM, being developed by the W3C, is mostly likely to become a widely accepted standard that will be fully supported on all platforms. SAX is a good candidate for handling large amounts of data that you do not want held in memory and which will be processed once through, such as feeding data to a database.

Freely available software already supports both DOM and SAX to varying degrees. Microsoft's MSXML Parser version 3.0 [3] provides a DOM interface with Microsoft specific extensions and a SAX like interface. MSXML is easily used from MS Visual Basic. The Apache XML Project [4] provides open source XML parsers for C++, Java and Perl available for a number of platforms, which, besides supporting DOM and a SAX like interface, also provides validation against schema.

So, with freely available PlantData XML Schema and freely available parser software, XML software interfaces become open for *anyone with the proper skills and enough time* to build a PlantData XML software interface.

While this single-layer map to XML architecture illustrated in Figure 4.3.1-2 is substantially less expensive to implement than the STEP architecture shown in Figure 4.3.1-1, there are some additional, substantial improvements and cost-reduction benefits that can be made by adopting a component-based architecture shown in Figure 4.3.1-4.

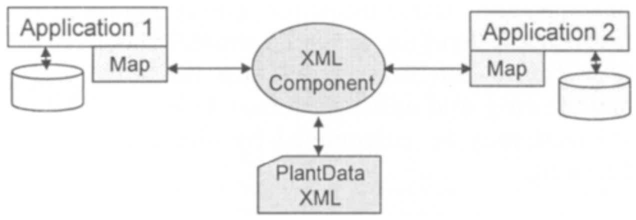


Figure 4.3.1-4 Component Architecture for XML software integration

The general purpose XML interfaces and the available software that implements them are oriented for software engineers' use on a wide range of non-engineering applications. There are additional issues that anyone building a process engineering XML interfaces will have to cope with. Examples of these include handling units of measurement and viewing the data objects in an XML file together in a more easily used engineering view of the data that aligns well with in a more easily used engineering view of the data that aligns well with the way current applications view the data. This reduces the cost of building the application maps by an *additional* substantial amount.

Another advantage of this component-based approach to data integration is that it allows direct synchronous data integration across applications using the common XML component. In this way the XML component can provide for both synchronous and asynchronous software integration architectures, depending on the needs of the users of the applications involved.

A process industry XML software component encapsulates the complexities of reading and writing the process industry XML file format, handling units of measure conversions and updates to the XML file format. The work to enable any application to be integrated with others is reduced to writing a small extension to the application that transfers the data between the application's internal data structures and a well defined, easily used, engineering oriented interface supplied by the XML component. In the Microsoft Windows' environment the process industry XML software component is most conveniently a COM/OLE object that can be used from Visual Basic, C++ or even Excel or Visio via Visual Basic for Applications.

A process industry focused XML software component acts as a catalyst to lower the activation energy barrier to implementing XML based software integration solutions and makes building application maps rapid, practical and cost effective (see Figure 4.3.1-4). Rather than every application owner having to face and deal with these common issues, there is an advantage to having a reusable process engineering XML component that hides the complexity of dealing with XML and dealing with the special process engineering data mapping issues.

An example of such a component is the ePlantData Software [5] COM XML gateway for parsing and writing process data. The component fully supports units of measurement and extensibility. It provides an interface designed for Process Engineers but also a fully W3C compliant DOM interface. The component uses a SAX like event-based parser to provide "load on demand" capabilities to preview data before loading required sections. The full ePlantData Software includes Excel based datasheets for easy viewing and editing Process Industry XML data in familiar data sheet formats that may be customized by end-users to match their own internal layout and format.

To summarize the problems and potential solutions to software integration and the benefits of various software integration architectures, we can draw an analogy to the energy levels associated with an exothermic chemical reaction, shown in Figure 4.3.1-4. There are very large "energy-release" economic benefits to be obtained by the process industry by moving from the current, highly labor-intensive work process to a standards-based electronic data exchange work process that is much less labor-intensive. In the case of a two-layer STEP architecture, the benefits are high, but so are the costs to implement interfaces. This places too high of activation energy barrier for most process industry companies to actually use STEP. Adopting a single-layer engineering oriented XML standard approach is a good catalyst, which reduces the activation energy barrier significantly. However, there is still a higher than necessary cost in terms of XML technology learning curves and duplication of effort across many companies on common issues such as units of measurement handling and engineering-oriented data mapping. Adopting a component-based approach is a better catalyst, offering the lowest cost approach of all, enabling standard XML data exchange to be widely achieved across the industry at lowest cost. Common industry-oriented XML software components such as ePlantData software that solve common application interfaces issues and implement industry standard XML vocabularies such as PlantData XML provide an effective solution to simplify and reduce the cost of integrating process industry software.

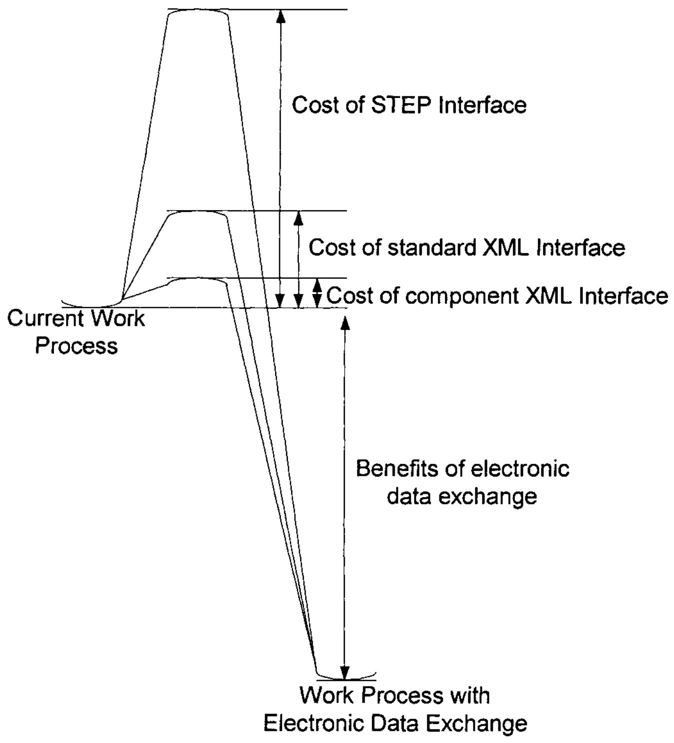

Cost of STEP Interface
Cost of standard XML Interface
Cost of component XML Interface
Current Work Process
Benefits of electronic data exchange
Work Process with Electronic Data Exchange

Figure 4.3.1-4 Energy activation barriers for electronic data exchange

4.2.4 POTENTIAL SCOPE FOR PROCESS ENGINEERING DATA EXCHANGE

One of the challenges for standardizing process engineering data exchange is the immense scope of technical data that *could* be exchanged. This immense breadth of scope was one of the difficulties for the previous STEP and ISO 15926 standard development activities, which contributed to the complexity and the length of time required to develop the standards. AP 227, which is the only process industry STEP standard that successfully reached the status of International Standard, had the narrowest scope of the all. The scope of AP 227 was confined to standard 3D representations of piping systems.

For AP 221 and AP 231, in international review of the scope of the draft standards, there was a strong pressure to expand the scope. For example, consider one of the key usage scenarios described in the AP 231 standard – that of exchanging an entire process design package from an owner-operator company who has developed the initial conceptual design to an engineering-procurement-construction (EPC) company to carry out a detailed design. Another scenario is to transfer the entire "as-built" facility data from the EPC contractor back to the owner-operator upon commissioning the facility. The amount and variety of process engineering technical information in such design packages is very broad.

Yet to achieve the very large promising benefits of electronic data exchange, this broad scope must eventually be achieved. So the practical question, if we are going to make progress on this immense problem becomes, "How do you eat an elephant?" The only practical answer to this question is "One slice at a time over some period of time."

The PlantData XML standard is being developed to cover the immense scope of process engineering data exchange by attacking it one slice at a time. One possible arrangement of slices is shown in Table 4.3.1-1. In this table the phases of the life cycle are shown horizontally, while the major subsets of process engineering data are listed vertically. The initial scope of PlantData XML is highlighted in Table 4.3.1-1

There are two areas of scope being addressed by PlantData XML in the initial focus of work. These are:

1. Process Materials
2. Equipment data sheets for shell and tube heat exchangers.

In the process materials area, PlantData XML is being developed to describe engineering aspects of process material data, in conjunction with the Design Institute for Physical PRoperties (DIPPR), which is a long-standing joint industry consortium organized under the AIChE. The DIPPR 991 project is currently undertaking an effort to harmonize and standardize the way process material data is described not only for its own databases, but for other important process industry consortia and data suppliers as well. These other sources of data include Dechema, NEL PPDS, API, TRC and the recent efforts by the IUCOSPED project to standardize electronic reporting of process material properties data in scientific and technical journals.

The second major focus area for PlantData XML is to develop and support electronic data sheets for shell and tube exchangers. The equipment data sheet was chosen precisely because it covers each phase of the life cycle from initial conceptual design through detailed design, procurement, construction and operation.

Table 4.3.1-1 Potential Scope of Data Exchange for Process Plants

	Design	Engineer	Procure	Construct	Operate
Process Materials					
- Component ID	DIPPR –				DIPPR –
- Chemical reaction	Plant-Data XML				Plant-Data XML
- Experimental data					
- Model parameters					
- Stream properties	PlantData XML and ePlantData Sheet				
Process Description					
- Process topology					
- Streams					
- Unit Operations					
+ Material transfer					
+ Heat transfer					
> heat exchanger	PlantData XML and ePlantData Sheet (initial)				
+ Mass transfer					
- Process control					
Plant Equipment					
- Plant items					
- Material transfer					
- Heat transfer					

> shell and tube	Plant Data XML and ePlantData Sheet (initial)
- Mass transfer	
- Instrumentation	
Engineering Doc's	
- Equipment lists	
- Data sheets	Plant Data XML and ePlantData Sheet (initial)
- PFD's	
- P&ID's	
PlantData	
- Site data	Plant Data XML and ePlantData Sheet (initial)
Project Data	Plant Data XML and ePlantData Sheet (initial)

As can be seen from Table 4.3.1-1, equipment data sheets cover a number of plant data sub-areas, including stream properties, unit operation data, equipment design data, engineering documents, site data and project data. Shell and tube exchangers were chosen because they are very common items of equipment in process plants and the design work process for these equipment items is sufficiently complex with multiple software packages by multiple vendors to where there will be large benefits achieved through electronic data exchange throughout conceive, design, procure, construct operate life cycle.

The initial scope of PlantData XML was deliberately chosen to be of limited scope so that beneficial results would be obtained quickly and delivered to users through commercial software. As user companies derive significant business value from applying the initial version of the standard, the PlantData XML standard will eventually be broadened in scope, in business-driven ways, to meet the entire scope of data that users collectively believe is cost-justified.

4.2.5 DEVELOPMENT METHODOLOGY FOR PlantData XML

4.2.5.1 Lessons learned from the STEP methodology

The ISO STEP methodology for developing standards is very rigorous and comprehensive, eventually resulting in highly reviewed standards documents. In this process the sponsor technical team produces a succession of draft standards documents, which are used to help define scope, which in the case of process industries were generally very broad scopes. The document is successively refined and reviewed in stages from Working Draft to Committee Draft, then to Draft International Standard and then to International Standard. At each stage, the draft documents are subjected to very extensive technical working committee reviews, independent international technical reviews and an international balloting process to gain international approval to move the draft standard to the next stage. Commercial software implementations are generally not done until the standard reaches the International Standard stage. International Standards must remain static for at least 3 years.

Unfortunately, at least in the experience of the Process Data eXchange Institute (pdXi) in developing the STEP AP 231 Committee Draft standard for conceptual process engineering data, there are several weaknesses in the STEP methodology as follows:

1. There is inadequate attention at the very beginning of the standard development process to ensure that the detailed work processes and usage scenarios are well understood and documented in depth and that economic benefits for each included data exchange scenario can be readily recognized and justified.
2. Building of demonstration software implementations by commercial software vendors is postponed until very late in the standards development process: it is not a requirement until the development of the Draft International Standard stage. If demonstration software implementations were required earlier in the standards development process on a very small subset of scope, such as a single usage scenario, then the difficulty of the two-layer STEP implementation architecture would have been uncovered much earlier in the development process.
3. The length of time and amount of effort required to produce and review the required documents at each stage is very long and costly, which tends to encourage sponsor groups to identify broad scopes so that they only have to go through these review cycles once. These broad scopes further aggravate the scope, schedule and complexity problem. The problem is further aggravated by the fact that the speed of standards development effort is largely governed by the availability of domain experts, who are usually participating on a volunteer basis.
4. The STEP architecture and methodology are very complex and was not well understood, at least by the AP 231 sponsor companies, until many years into development process. For example, when work started in 1991, pdXi sponsor companies required that the work be done in a manner suitable to build a STEP standard, but imposed this requirement without a full understanding the cost and complexity of the two-layer application mapping approach required by STEP. As early as 1994 pdXi built trial software implementations assuming that the data model could be expressed in physical files at the engineering domain level, and later discovered that these were not standards-compliant interfaces after all. It was not until 1998, 7 years after the pdXi effort started and 3 years after the AP 231 standard development effort started, when it was recognized that a STEP-compliant software implementation was required to work at the generic, STEP integrated resources level. Once the high cost of STEP software implementation was recognized, work stopped on continuing the standard development.

These were difficult, expensive lessons for pdXi and the AP 231 development team to learn about developing and implementing STEP standards. However, the technical work and the extensive international review of the engineering level data model are still a very worthwhile piece of technical work reflecting an investment of $1-2 million by the process industry companies that developed the draft standard. All that is needed now is a more cost-effective approach to software implementation. Implementing the engineering level data model using XML offers such an approach.

4.2.5.2 Methodology for Developing PlantData XML Standard

The methodology for developing the PlantData XML standard reflects the lessons learned from the STEP development process and adapts key ideas from the Unified Software Development process described in section 4.1.4. A key piece of the process is to use the engineering domain data models developed by the STEP standards to take advantage of the good technical work that has already been done.

The major tasks in developing PlantData XML are listed below:

1. Describe task-oriented work flows, use cases and data flows
2. Identify reusable subsets of process engineering data that participate in the data flows. Exclude any data that does not directly participate in a commercially important work process and data flow.
3. For a single identified subset of data, start with the draft ISO standards and develop a UML object data model for the reusable subsets of data that match the identified tasks, use cases and data flows.
4. Develop XML schema from UML object model, organizing the logically related reusable subsets of data into XML namespaces.
5. Define engineering-oriented application programming interfaces that support the engineering view of the data used in the identified data flows.
6. Develop a reusable software component that uses the engineering-oriented application programming interfaces to read and write XML files that conform to the defined XML schema.
7. Develop software implementations that use the XML software component for the identified tasks, use cases and data flows.
8. Repeat steps 3-7, incrementally adding data scope as needed to meet the needs of the identified tasks, use cases and data flows.
9. Publish the resulting XML schema as a standard that has been proven and used by commercial software implementations.

At the time of this writing (early 2001), PlantData XML development is just starting. In the following sections we include some initial prototype examples to show how PlantData XML is being developed using the above work process. For the most up to date versions of PlantData XML, consult the ePlantData web site [5].

4.2.6 PROCESS ENGINEERING WORK FLOW & DATA FLOW

The first task in developing PlantData XML is to describe the work process, tasks, tools and data flows to understand where the economic opportunities are for automating the technical data exchange. A portion of the process engineering work process including major roles, tasks, tools and data flows, which are being developed for the PlantData XML, are shown in Figure 4.3.1-5.

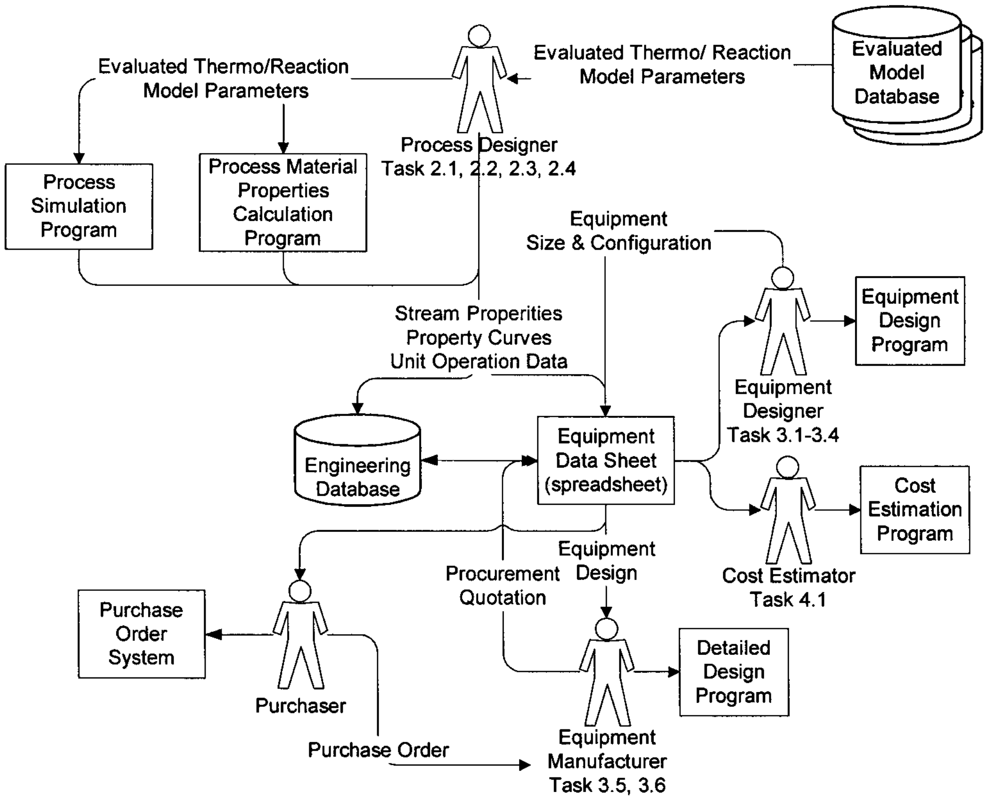

Evaluated Thermo/Reaction Model Parameters
Evaluated Thermo/ Reaction Model Parameters
Evaluated Model Database
Process Simulation Program
Process Material Properties Calculation Program
Process Designer Task 2.1, 2.2, 2.3, 2.4
Equipment Size & Configuration
Stream Properities Property Curves Unit Operation Data
Equipment Design Program
Equipment Designer Task 3.1-3.4
Engineering Database
Equipment Data Sheet (spreadsheet)
Cost Estimation Program
Cost Estimator Task 4.1
Equipment Design
Procurement Quotation
Purchase Order System
Purchaser
Detailed Design Program
Purchase Order
Equipment Manufacturer Task 3.5, 3.6

Figure 4.3.1-5 Process Engineering Roles, Tasks and Data Flows

The detailed task descriptions referred to by number in Figure 4.3.1-5 are shown in an accompanying set of tables, such as the one shown in Table 4.3.1-2.

Table 4.3.1-2 Work tasks associated with a portion of the process engineering work process

ID	Task	Done by	Task Input Data	Input From	To Task Application	Task Output Data
3.1	Specify process specification data for process equipment item, e.g., S&T heat exchanger	Owner-operator or EPC	Material streams Stream properties Stream Property curves Unit Operation	Process equipment simulation program Process Simulation program	Equipment data sheet program (e.g., spreadsheet) Front-end engineering database	Data Sheet information Project information Equipment Item Data Unit Op Data Material Stream and Stream Properties Stream Property curves
3.2	Design/size process equipment, e.g., heat exchanger	Owner-operator or EPC	Material streams Stream properties Stream property curves Unit Operation	Equipment data sheet Equipment simulation program Process Simulator	Equipment Design Program, e.g., S&T exchanger design program	Equipment size and configuration data, e.g., for S&T exchangers
3.3	Rate process equipment performance, e.g., heat exchanger	Owner-operator, EPC or equipment manufacturer	Unit Op Data Material Stream Properties Property curves Equipment size and configuration data	Equipment data sheet Front-end eng database Equipment Design Program	Equipment Design Program Process Simulation program	Equipment size and configuration data (revised)
3.4	Specify size and configuration specification data for process equipment item, e.g., S&T heat exchanger	Owner-operator or EPC	Material stream(s) Stream properties Stream property curves Unit Operation(s) Equipment size and configuration	Equipment Design Program Equipment data sheet program	Equipment data sheet program Front-end engineering database	Data Sheet information Project Information Equipment Item Data Unit Op Data Material Stream Properties Property curves Equipment size and configuration Material of Construction

A work flow narrative can be developed corresponding to the diagrams and tables. For the work flow diagram shown in Figure 4.3.1-5, we can write the following narrative.

A process designer sets up a process simulation for the process being studied. Alternatively, the process designer may set up and use a material properties simulator for a single equipment item. A key input requirement of the process simulation or material properties simulator is to enter the components, select a thermodynamic method and obtain the thermodynamic model parameters from an evaluated thermodynamic model database. Next the feed material streams, unit operations and process topology need to be defined to complete the process simulation.

Results from the process simulator or the material properties simulator including stream properties, property curves and unit operation data are loaded into appropriate process load equipment data sheets.

The process designer initiates a number of equipment data sheets, one for each equipment item in a flowsheet. The process designer adds information about the company, the site, the project and preliminary material of construction selection for each equipment data sheet. In some companies, information on the equipment data sheet may be maintained by a front-end engineering database system.

The cost estimator takes preliminary design information from the process load data sheet to obtain an initial cost estimate.

The equipment designer takes process information from the equipment data sheet and runs the equipment design program to size and rate the equipment. The equipment designer updates the equipment data sheet with the equipment size and geometric configuration design details.

The cost estimator update the cost estimate based on the equipment sizing results.

The equipment designer works with purchasing to request quotations from equipment manufacturers.

Equipment manufacturers run detailed design programs to complete the mechanical design and re-rate the thermal design if needed to guarantee performance. Manufacturers fill out equipment data sheets with all details and return the equipment data sheet with a cost quotation.

Cost estimators update the cost estimate with cost quotations.

The equipment designer checks the manufacturer's design and re-rates the design if necessary. The equipment designer selects the equipment based on the quotations and purchasing procures the equipment with a purchase order.

We can now use the diagram, tables and the narrative to identify reusable data groups that are involved in the data flows associated with this work process. For example the following data groups are suggested by the above example workflows.

1. Component Identification
2. Thermodynamic method and associated model parameters
3. Process material stream (includes composition, state, flow rate and stream properties)
4. Process material stream property curves
5. Unit operation data (*e.g.*, heat exchanger)
6. Site data
7. Project data
8. Document data for data sheets
9. Plant item equipment data
10. Material of construction data
11. Shell and tube exchanger configuration data
12. Shell and tube exchanger detailed mechanical specification data
13. Shell and tube cost data
14. Purchase requisition data

From here we can start to develop the UML data models and the XML schema in more detail. We will use a subset of the data groups listed above. For the remainder of this example, we will choose Task 3.1 – specify process data for design of a shell and tube exchanger.

4.2.6 EXAMPLE UML OBJECT DATA MODEL

For an introduction to the Unified Modeling Language (UML), please refer to section 4.1.5.2. For PlantData XML, we use the UML static class diagrams, where the class properties are defined. However, we do not make use of the class methods, because we are only building an object data model, not an object-oriented software application.

In this example, we will use a UML object data model for a small subset of the information that we find on equipment data sheets. This UML model was derived

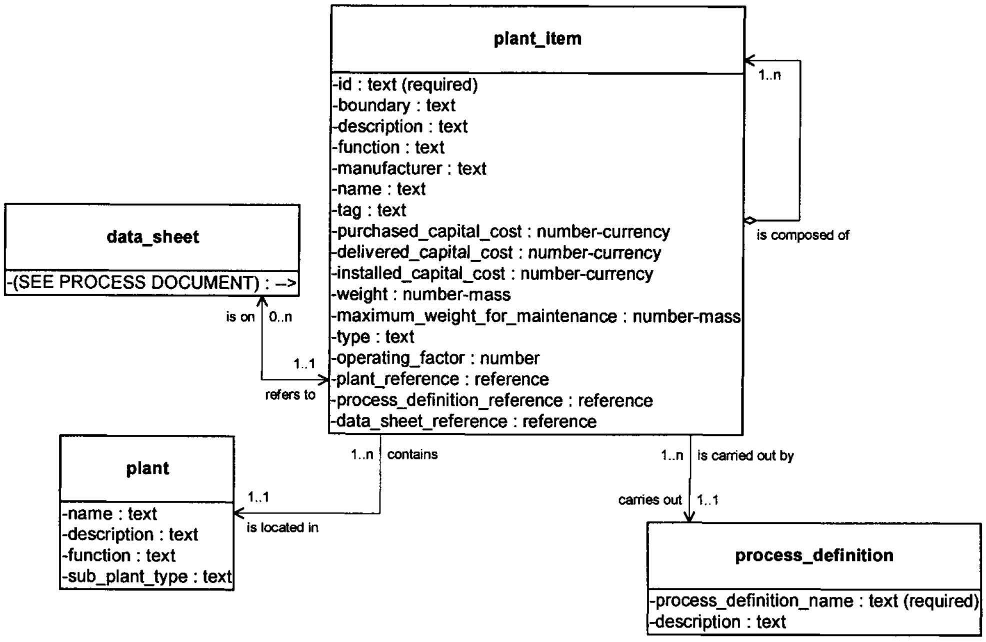
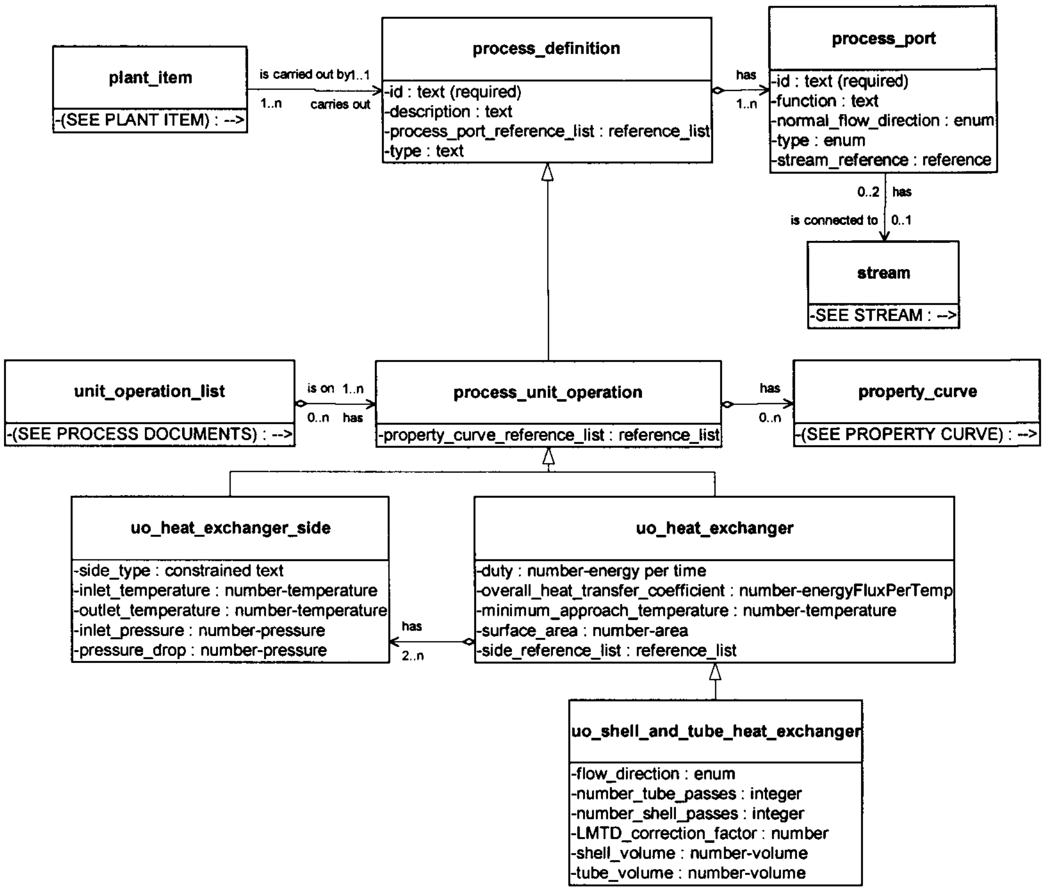


Figure 4.3.1-6 UML diagram for Plant Item

from the AP 231 Application Reference Model (ARM), which uses IDEF1X notation.

Figure 4.3.1-6 shows the general information that can be attributed to any plant item, regardless of what type of equipment item it is. This model shows that a data_sheet refers to exactly one plant_item, and that the plant_item is located in exactly one plant. The plant_item may itself be composed on one or more plant_items and the plant_item carries out a process_definition. There are number of attributes associated with a plant_item such as its tag and its manufacturer. Note that some attributes in plant_item are references, for example a reference to the process_definition, where we can find more information about the process service for the plant_item. These reference attributes were not originally in the AP 231 data model, but were added to the UML to resolve the "navigation" ambiguities that are present in the AP 231 data model. Resolving these navigation ambiguities is essential to obtain a practical XML schema and software implementation.



plant_item
-(SEE PLANT ITEM) : -->
is carried out by 1..1
1..n carries out
process_definition
-id : text (required)
-description : text
-process_port_reference_list : reference_list
-type : text
has
1..n
process_port
-id : text (required)
-function : text
-normal_flow_direction : enum
-type : enum
-stream_reference : reference
0..2 has
is connected to 0..1
stream
-SEE STREAM : -->
unit_operation_list
-(SEE PROCESS DOCUMENTS) : -->
is on 1..n
0..n has
process_unit_operation
-property_curve_reference_list : reference_list
has
0..n
property_curve
-(SEE PROPERTY CURVE) : -->
uo_heat_exchanger_side
-side_type : constrained text
-inlet_temperature : number-temperature
-outlet_temperature : number-temperature
-inlet_pressure : number-pressure
-pressure_drop : number-pressure
has
2..n
uo_heat_exchanger
-duty : number-energy per time
-overall_heat_transfer_coefficient : number-energyFluxPerTemp
-minimum_approach_temperature : number-temperature
-surface_area : number-area
-side_reference_list : reference_list
uo_shell_and_tube_heat_exchanger
-flow_direction : enum
-number_tube_passes : integer
-number_shell_passes : integer
-LMTD_correction_factor : number
-shell_volume : number-volume
-tube_volume : number-volume

Figure 4.3.1-6 UML data model for heat exchanger unit operation

Figure 4.3.1-6 shows the UML data model for the heat exchanger unit operation, which is a specific sub-type of process_definition, which carries out the process function of the plant_item in Figure 4.13.1-5. A process definition may have 1 or more process_ports, to which a stream may be connected. A unit operation may have 0 or more property_curves. A uo_shell_and_tube_heat_exchanger is a specific sub-type of uo_heat_exchanger, which may have 2 or more sides. The uo_heat_exchanger has familiar attributes such as duty and overall heat transfer coefficient. The uo_shell_and_tube_heat_exchanger has attributes specific to that type of exchanger such as number of tube passes, number of shell passes, *etc.*

For each item in the UML data model a data glossary definition is developed, again using the definitions that were originally developed for AP 231. STEP does not require that the data type and units of measure be included in the definition, but we believe this essential information for PlantData XML. For example, Figure 4.3.1-6 shows a sample of some glossary definitions for the Uo_heat_exchanger class.

Uo_heat_exchanger
A Uo_heat_exchanger is a type of Process_unit_operation that is the heating or cooling of a single process Stream, the exchange of heat between two process Streams, the exchange of heat between a process Stream and a Utility_stream, or the exchange of heat between multiple Streams by a heat exchanger.

The data associated with Uo_heat_exchanger are the following:
— duty;
— minimum_approach_temperature;
— overall_heat_transfer_coefficient;
— surface_area;

duty
A duty specifies the amount of heat transferred from the hot Stream to the cold Stream by the exchanger. Data Type: number; Units type: energy per time, *e.g.*, kilowatt

minimum_approach_temperature
A minimum_approach_temperature specifies the minimum local difference in temperature between the shell side fluid and the tube side fluid at all points of contact in the exchanger. Data Type: number; Units type: temperature, *e.g.*, Kelvin

overall_heat_transfer_coefficient
An overall_heat_transfer_coefficient specifies the overall rate that heat is transferred from the hot fluid on one side of the heat exchanger to the cold fluid on the other side. This parameter relates the total amount of heat transferred by the exchanger to its heat transfer surface_area and the driving temperature differences. Data Type: number; Units type: energy per time per area per temperature, *e.g.*, kilowatts/hour-K

4.2.7 EXAMPLE XML SCHEMA DERIVED FROM UML OBJECT DATA MODEL

At the time of this writing, XML Schema is at Candidate Recommendation status from the W3C. XML Schema provides an XML based approach for specifying standard data types and the ability to define structured data in XML files. Recalling from Figure 4.1.5-1, XML Schema sits on top of XML Document Type Definitions (DTD's). Domain schema such as PlantData XML sits on top of XML Schema, *i.e.*, uses XML Schema, which eventually results in DTD's to obtain, valid, well-formed XML data files.

For PlantData XML, we take the UML object data model and apply some rules by convention for converting UML classes, properties and relationships, and transform it into an XML Schema representation, a subset of which is shown in Figure 4.3.1-7.

```
<ElementType name = "UoShellAndTubeHeatExchanger" content = "eltOnly" order = "seq">
  <AttributeType name = "id" dt:type = "id" required = "yes"/>
  <AttributeType name = "UoShellAndTubeExchangerFlowDirection" dt:type = "enumera-
tion" dt:values = "cocurrent countercurrent crosscurrent multipass unspecified" default =
"countercurrent"/>
   <attribute type = "id"/>
  <attribute type = "UoShellAndTubeExchangerFlowDirection"/>
  <element type = "UoHeatExchangerDuty" minOccurs = "0" maxOccurs = "1"/>
  <element type = "UoHeatExchangerOverallHeatTransferCoefficient" minOccurs = "0"
maxOccurs = "1"/>
  <element type = "UoHeatExchangerMinimumApproachTemperature" minOccurs = "0"
maxOccurs = "1"/>
  <element type = "UoHeatExchangerSurfaceArea" minOccurs = "0" maxOccurs = "1"/>
  <element type = "UoHeatExchangerSide" minOccurs = "1" maxOccurs = "*"/>
  <element type = "UoShellAndTubeExchangerNumberTubePasses" minOccurs = "0"
maxOccurs = "1"/>
  <element type = "UoShellAndTubeExchangerNumberShellPasses" minOccurs = "0"
maxOccurs = "1"/>
</ElementType>

<ElementType name = "UoHeatExchangerOverallHeatTransferCoefficient" content = "tex-
tOnly" dt:type = "number">
 <AttributeType name = "energyFluxPerTempUnits" dt:type = "enumeration" dt:values =
"W.m-2.K-1 BTU.hr-1.ft-2.R-1" default = "W.m-2.K-1"/>
 <attribute type = "energyFluxPerTempUnits"/>
</ElementType>

<ElementType name = "UoHeatExchangerMinimumApproachTemperature" content = "tex-
tOnly" dt:type = "number">
 <AttributeType name = "temperatureUnits" dt:type = "enumeration" dt:values = "K C R F"
default = "K"/>
 <attribute type = "temperatureUnits"/>
</ElementType>

<ElementType name = "UoHeatExchangerSurfaceArea" content = "textOnly" dt:type =
"number">
  <AttributeType name = "areaUnits" dt:type = "enumeration" dt:values = "m2 ft2" default =
"m2"/>
  <attribute type = "areaUnits"/>
</ElementType>
```

Figure 4.3.1-7 Partial XML Schema derived from the UML data model.

Note that the schema uses class names and attribute names from the UML data model. Also, note that some attributes have units of measurement defined alternative valid character strings that are acceptable for designating units.

Taking this XML Schema, then it now becomes possible to instantiate XML data files that conform to the PlantData XML Schema. An example XML data file is shown in Figure 4.3.1-8.

```
<UoShellAndTubeHeatExchanger id = "101-E" UoShellAndTubeExchangerFlowDirection
= "countercurrent">
  <ProcessDefinitionDescription>Stripper Reboiler</ProcessDefinitionDescription>
  <ProcessDefinitionType>UoShellAndTubeHeatExchanger</ProcessDefinitionType>
  <ProcessPort id = "shell inlet" ProcessPortType = "material" ProcessPortNormalFlowDi-
rection = "inlet">
    <ProcessPortFunction>tower stage n-1 feed to reboiler</ProcessPortFunction>
    <ProcessPortStreamReference>T-101 Stg n-1 Liquid</ProcessPortStreamReference>
  </ProcessPort>
  <UoHeatExchangerDuty energyFlowUnits = "BTU.hr-1">3785400
                                  </UoHeatExchangerDuty>
  <UoHeatExchangerOverallHeatTransferCoefficient energyFluxPerTempUnits = "BTU.ft-
2.hr-1.R-1">69</UoHeatExchangerOverallHeatTransferCoefficient>
  <UoHeatExchangerMinimumApproachTemperature temperatureUnits = "F">20
                                  </UoHeatExchangerMinimumApproachTemperature>
  <UoHeatExchangerSurfaceArea areaUnits = "ft2">8471
                                  </UoHeatExchangerSurfaceArea>
  <UoHeatExchangerSide UoHeatExchangerSideType = "shell">
  <UoHeatExchangerSideInletTemperature temperatureUnits = "F">195.8
                                            </UoHeatExchangerSideInletTemperature>
  <UoHeatExchangerSideOutletTemperature temperatureUnits = "F">231.1
                                            </UoHeatExchangerSideOutletTemperature>
  <UoHeatExchangerSideInletPressure pressure-gUnits = "psig">234.2
                                            </UoHeatExchangerSideInletPressure>
  <UoHeatExchangerSidePressureDrop pressure-gUnits = "psig">8
                                              </UoHeatExchangerSidePressureDrop>
</UoShellAndTubeHeatExchanger>
```

Figure 4.3.1-8 Partial XML Data file based on XML schema

At this point we have shown partial examples of the PlantData XML development methodology to illustrate the following steps.

1. We identified a commercially important use case – specifying process unit operation data onto a shell and tube exchanger data sheet and passing

that information to a heat exchanger designer for subsequent use in a heat exchanger design program.
2. We developed a partial UML data model that was directly derived from the AP 231 engineering data model.
3. We used the UML data model to derive an XML Schema.
4. We used the XML Schema definition to populate and XML data file.

Continuing through the PlantData XML methodology, we next complete an engineering-oriented application programming interface, implement the reusable XML software component that reads and writes the XML data files and finally deploy that component in several commercial software packages. Then we iteratively and incrementally add capabilities until the commercially important scope of data, e.g., for a shell and tube exchanger data sheet has been completed.

4.2.8 SUMMARY

In this chapter, we reviewed the motivations for software integration architectures in the process industry. We described PlantData XML and have shown how a process industry standard XML vocabulary, combined with reusable XML software components can lead to a very cost-effective synchronous and asynchronous software integration architecture that also provides a sound technical basis for vendor-neutral data archival and more intelligent B2B e-commerce transactions in the process industry.

PlantData XML builds upon the previous ISO standards development work. PlantData XML unlocks the value in these standards by making these standards much easier and more practical to use as well as being orders of magnitude less expensive to implement in commercial software.

We have described a development methodology for PlantData XML that follows the principles of the Unified Software Development Process, whereby commercially significant use cases are developed first, and then used to iteratively and incrementally develop capabilities over a period of time to eventually handle the extremely large scope of process plant information. In illustrating this methodology by an example for shell and tube exchangers, we have shown how it is possible to derive a UML data model from the AP 231 IDEF1X engineering data model and then develop an XML Schema that reflects the structure in the UML data model.

PlantData XML is being implemented in commercial software as part of its development process. Several companies have committed to use PlantData XML and the ePlantData software component to achieve commercially successful software integrations before the end of 2001. Process designers from Institut Francais du Petrole (IFP) and equipment designers from Kellogg Brown & Root (KBR) will use process simulation programs, equipment design programs and equipment detailed mechanical design programs throughout the life cycle of heat exchanger design, engineering and procurement. Leading providers of these software packages will provide commercial software that implement the PlantData XML standard. These companies include Chemstations, Inc., a global supplier of process simulators and heat exchanger design packages, Heat Transfer Research, Inc. (HTRI), a leading provider of thermal heat exchanger design software, and COADE, Inc., a global provider of detailed mechanical vessel design software. WaveOne.com is an Application Services Provider who provides process engineering software, and XML data repositories to support team-based collaborative problem solving environments on the Internet. All of these forward-looking vendors have committed to implement PlantData XML interfaces into their commercial software in 2001. Users of these leading software companies' products and services will experience the benefits of PlantData XML software integration architectures first. As practical usage spreads and significant benefits are realized, we anticipate more vendors will support the PlantData XML industry standard as well, as their users demand the benefits of open software architectures and support for open industry data exchange protocols.

The PlantData XML standard is published and freely available to any companies in the process industries to use. A commercial, reusable XML software component, *ePlantData*, and an electronic data sheet, *ePlantData Sheet* is also available for process industry companies and software vendors to begin using PlantData XML right away at low cost. Through the standards and the software components, we anticipate that the process industry will have finally achieved a practical, inexpensive approach to multi-vendor, multi-company software integration. With the eventual widespread and broad application of electronic data exchange architectures over the entire life cycle of process facilities, process industry companies should reap annual economic benefits in the hundreds of millions to billions of dollars.

We can only speculate about the future of the Internet, but we believe strongly that XML is *THE* key enabling technology for the next generation of the Internet. At this time, the Internet is literally being rebuilt migrating away from using HTML technology to using XML technology. XML is likely to be used in a whole host of ways that we cannot even imagine now. For example, we can imagine, geographically remote collaborating teams sharing project information and technical details at fine grained levels rather than coarse-grained document levels like we do now. We can imagine using XML to remotely monitor process plants in a remote client-server extranet architecture. We imagine XML will be a key software integration mechanism across the Internet for all kinds of e-commerce transactions. For example, it should be possible to create Internet agents that search out qualified suppliers based on a highly detailed set of technical requirements that are part of electronic data sheets and electronic project and procurement databases. However, while the promise of using XML to its maximum potential for the benefit of the process industry is exciting to contemplate, it clearly depends on the process industry developing a comprehensive set of easy-to-use and inexpensive-to-implement industry XML standards. PlantData XML is an initial start down this exciting path to the future.

4.2.9 REFERENCES

[1] W3C Introduction to DOM web page:
www.w3.org/TR/WD-DOM/introduction.html
[2] David Megginson's web site on SAX: http://megginson.com/SAX/index.html
[3] Microsoft XML Developer Centre Web site:
http://msdn.microsoft.com/xml/default.asp
[4] Apache XML Project web site: xml.apache.org
[5] ePlantData web site: www.ePlantData.com

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 4.3: STEP for the Process Industries

R. Murris

4.3.1 THE CHALLENGE

Although the discipline of process engineering is well supported by computer-enhanced tools and software architectures, they are mainly focused and thus applicable to only a small part of the process plants lifecycle. The average 30 years process plant life span of a typical refinery generally outlives all operating systems, software and databases designs, which were in use when the first design drawing was conceived. The longevity of process plants and the ongoing functional and physical changes during its life thus set out a real challenge for systems and software designers alike.

During the conceptual engineering phase process plant design information flows intensively between specialized departments and once the plant is built operational data is recycled to be used as input for plant optimization and revamps. So data from many sources, created by a multitude of applications must be exchanged not only within the company but also increasingly with supporting companies like equipment suppliers, engineering and maintenance contractors. In the 1980's the exchange data consisted mainly of non-intelligent 2D schematics and 3D-shape representation and was dominated by the de-facto standards like AutoCAD's dxf/dwg formats and the more official but ambiguous standard called IGES.

During the mid 1980's Computer Aided Design support tools became mainstream in the process plant design cycle and the requirements for data exchange between the variety of implementations grew outside the scope and capabilities of the mentioned (*de-facto*) graphics data exchange standards.

4.3.2 HISTORY

In the 1980's the US department of defense (DoD) was one of the firsts to identify the growing data exchange problem. Efficient logistics around defense material and spare-parts for military weapons systems is one of the pre-requisites for fast military deployment. In 1980 the DoD relied on approximately 3500 third parties supplying equipment and parts. CALS (Continuous Acquisition and Logistics Support) was founded to reduce the huge amount of paper-based documents, the poor communication between the suppliers and to reduce the cycle time between order and supply. The CALS exchange requirements stretched beyond the boundaries of graphics alone. It also included business and product data like purchase orders, invoices, product specification, sound, images etc. Later additional requirements like 'openness' and 'software vendor independence' were added. In 1986 these became the founding principles of the ISO-10303 (STEP) standard.

4.3.3 ISO-10303 or STEP

The STandard for the Exchange of Product model data (STEP) is the acronym for the full ISO-10303 standard. STEP consists of parts each having a distinct function within the formalization of exact data exchange within a predefined set of business activities (the scope).

All the steps from the outset of the data exchange scope until the validation of an application protocol in the form of data test-suites are supported by this standard. The methods for creating an application protocol are all part of the STEP standard and are validated and quality assured through the ISO technical sub committee TC184/SC4. The acceptance of an application protocol as an International Standard (IS) although supervised by ISO is done by the p-members. These permanent members (countries) of ISO take part in several balloting procedures carrying the protocol from a NWI (new work item) to the final IS. Due to the formal and lengthy procedures it can take up to seven years for a particular application protocol to acquire the IS status.

4.3.4 APPLICATION PROTOCOL 221 (AP221)

In 1995, subsidised by the EC the ESPRIT III project ProcessBase started with the development of the ISO-10303 Application Protocol 221. The title "Functional data and 2D schematic representation" relates to the conceptual and detailed design and engineering stages of the process plant lifecycle. During these activities functional design data and their representations on 2D drawings are frequently passed within the plant-owner's organisation and between contractor and client. Both roles of client and contractor are played by plant-owners, engineering contractors and equipment suppliers in any combination.

Due to the increasing variety in software applications and the difference in data models owned and operated by these organisations, standardised exchange of data within the scope of AP221 became an urgent requirement. When the ProcessBase project ended in 1997 there was still no AP221 standard but it was embodied and further developed by a consortium of European companies. The development of an Application Protocol such as AP221 has turned out to be far more difficult than initially ex-

pected. Due to the enormous amount of different data types (meta-data) within the AP's scope it deemed impossible to create one single exchange datamodel to cover it all. The second challenge was the requirement of full lifecycle support. Thus within a single data exchange multiple versions of *e.g.* the same equipment must be passed without having redundant data. Especially the combination of Data Explicitly Defined Once (DEDO) and versioning within the same exchange scenario resulted in a new modelling concept, which will here be referred to as the Associative Paradigm. This modelling concept has been implemented, with some exceptions, throughout the full requirement model (Application Reference Model) of AP221.
The methodology to handle huge amount of process plant functional data and schematics using this Associative Paradigm is an unexplored area. However implementations with traditional relational data tables or objects oriented counterparts have proven until today to be either too inflexible or just impossible.

4.3.5 THE ASSOCIATIVE PARADIGM

Traditionally we find requirements of data models expressed in languages like EXPRESS (ISO-10303-11), NIAM, ERD and lately in UML. These descriptive methods, sometimes accompanied with a strict syntactic language all use a basic set of axioms to model models. In some cases they also define a symbol library to conveniently communicate the results of the modelling process to the end-user or developer. In the case of EXPRESS the symbolic notation capability (EXPRESS-G) is far less than what is supported by the modelling language itself. However a clever subset has been defined which combines powerful, compact presentation of the model without the end-viewer losing overview. Further details when required can be extracted from the EXPRESS code.

So when modelling languages fulfil the same requirements, is there a common meta-model for these languages as well? Probably there is and in the case of the EXPRESS language this meta-model is quite compact when we only consider the data requirement part and exclude the syntactic rules that deal with rules, procedure, functions and uniqueness constraints. Basically the EXPRESS language defines three relation types and five basic type of modelling objects.

The basic five objects are 'schema', 'entity', 'type', 'attribute' and 'data type'. The three basic relation types are 'specialisation', 'composition' and 'possession'. A schema can be composed of 0..n entities and types, an entity can possess 0..n attributes and an attribute possesses exactly one data type. Each entity can have 0..n specialisations to other entities. The following table expresses the relations between the association types and the object types.

Table 1 Meta model definitions for the EXPRESS language

Object type	*Role*	*Association Type*	*Role*	*Object Type*
Schema	Whole	Composition	Part	Entity
Schema	Whole	Composition	Part	Type
Schema	Whole	Composition	Part	Data Type
Entity	Possessor	Possession	Possessed	Attribute
Attribute	Possessor	Possession	Possessed	Data Type
Entity	Supertype	Specialisation	Subtype	Entity

In 'natural' (English) language the first record under the header reads as: '"An instance of ' 'object type' 'encoded as' 'schema' 'plays the role as' 'an instance of' 'role' 'encoded as' 'whole' 'in relation with' 'an instance of' 'association type' 'encoded as' 'composition' 'with' 'an instance of' 'object type' 'encoded as' 'entity' 'playing the role as' 'an instance of' 'role' 'encoded as' 'part'". This includes all the implicit relations made explicit between the table header information (meta-data) and the records represented in lines under the header. When we omit all this implicit information we get more readable sentences, however information is lost when only these sentences are transferred to any receiver who is not aware of this table header and structure. The information in the records of table 1 are made more readable and are given in table 2:

Table 2 Natural language equivalent of the information in table 1

'Schema' 'can be composed of' 'entities'
'Schema' 'can be composed of' 'types'
'Schema' 'can be composed of' 'data types'
'Entity' 'can possess' 'attribute'
'Attribute' 'can possess' 'data type'
'Entity' 'can be specialised as' 'entity'

Clearly the described cardinality constraints are missing from the previous table although defined in the explanatory text. This means that the natural language equivalents of the association types all begin with 'can...' whereas some relations require 'must...'. An example is the relation between 'attribute' and 'data type'. In table 1 this could be achieved by inserting two extra columns (cardinality 1 and 2, abbreviated as C1 and C2) with the possible values 1 and range value 0...n.

Table 3 expanded with cardinalities

Object type	*Role*	*C1*	*Association Type*	*C2*	*Role*	*Object Type*
Schema	Whole	1	Composition	0..n	Part	Entity
Schema	Whole	1	Composition	0..n	Part	Type
Schema	Whole	1	Composition	0..n	Part	Data Type
Entity	Possessor	1	Possession	0..n	Possessed	Attribute
Attribute	Possessor	1	Possession	1	Possessed	Data Type
Entity	Supertype	1	Specialisation	0..n	Subtype	Entity

The explicit sentences will really become unreadable because of the size, however the natural language equivalent where the same implicit information is omitted become:

Table 4 Natural language equivalent of table 3

'One' 'schema' 'can be composed of' 'zero to many' 'entities'
'One' 'schema' 'can be composed of' 'zero to many' 'types'
'One' 'schema' 'can be composed of' 'zero to many' 'data types'
'One' 'entity' 'can possess' 'zero to many' 'attributes'
'One 'attribute' 'must possess' 'one' 'data type'
'One' 'entity' 'can be specialised as' 'zero to many' 'entities'

The definitions and relations in tables 1-4 all describe what <u>can</u> be instantiated and not what <u>is</u> instantiated. So the information expressed in these tables can be used as a 'template' for instantiation of individual records in some implementation. Or the other way around: individuals matching the constraints can be classified as validated members of that template. One can either explicitly relate an individual with an instance of association type 'classification' or allow software automatically establish this association by invoking a matching procedure.

When we re-read the tables we probably infer that the records all belong to a specific information set. So implicitly they are grouped to convey some meaning. When the word 'template' was mentioned probably some grouping was meant. In this case the inclusion of the information expressed by the six different sentences. So without writing it out explicitly we probably inferred that the scope or view of the template included only those six expressions. One can assume however that the tables 1...4 can contain millions of records and without a mechanism to group these expressions no segregation can be made and its use would be limited. Let's assume that the definition of this template is part of the context of something we have defined as 'EXPRESS modelling' and recreate table 4 with this information:

Table 5 Expanded with context

'One' 'schema' 'can be composed of' 'zero to many' 'entities' 'is included in' 'EXPRESS modelling'
'One' 'schema' 'can be composed of' 'zero to many' 'types' 'is included in' 'EXPRESS modelling'
'One' 'schema' 'can be composed of' 'zero to many' 'data types' 'is included in' 'EXPRESS modelling'
'One' 'entity' 'can possess' 'zero to many' 'attributes' 'is included in' 'EXPRESS modelling'
'One 'attribute' 'must possess' 'one' 'data type' 'is included in' 'EXPRESS modelling'
'One' 'entity' 'can be specialised as' 'zero to many' 'entities' 'is included in' 'EXPRESS modelling'

What does the association type instance expressed as 'is included in' actually include? All the separate concepts expressed as words within single quotes or the explicit association instances between those concepts as described above in the very long sentence or both?

To include all the explicit associations between the concepts we first must make them visible. In the next table the first record (sentence) of table 5 is taken and the explicit associations are made visible (excluding the association with the context):

Table 6 Making the sentence explicit

Sub phrase	*Explicit*
'One' 'schema'	'One' 'is count of' 'schema'
'is count of'	'counter' 'is role of' 'object' 'is count of' 'counted' 'is role of 'subject'
'can be composed of'	'whole' 'is role of' 'object' 'can be composed of' 'part' 'is role of' 'subject'
'zero to many' 'entities'	'entities' 'must be in range' 'in between' 'zero' and 'many'
'must be in range'	'subject to range' 'is role of' 'object' 'must be in range' 'range' 'is role of' 'subject'
'in between'	'minimum count' 'is role of' 'object' 'in between' 'maximum count' 'is role of' 'subject'

Now we bring all the associations to the left and use a different notation form. Hereby we introduce the term 'FACT' and make use of line or record identifiers to allow for easy referencing. The presented notation form is identical to the data section body defined in ISO-10303 part21:

Table 7 Fact table for one sentence without context

Facts
#1=FACT('is count of',#2,#3);
#2=FACT('is role of','counter','object');
#3=FACT('is role of','counted','subject');
#4=FACT(#1,'one','schema');
#10=FACT('must be in range',#11,#12);
#11=FACT('is role of','subject to range','object');
#12=FACT('is role of','range','subject');
#13=FACT('in between',#14,#15);
#14=FACT('is role of','minimum count','object');
#15=FACT('is role of','maximum count','subject');
#16=FACT(#13,'one','many');
#17=FACT(#10,'entities',#16);
#20=FACT('can be composed of',#21,#22);
#21=FACT('is role of','whole','object');
#22=FACT('is role of','part','subject');
#100=FACT(#20,#4,#17);

By following '#100=FACT(#20,#4,#17)' one can fully reconstruct the sentence by travelling through the table following the references. For completeness we now also add the context and the inclusion of the just described #100=FACT:

Table 8 Fact table for one sentence with context

Facts
#100=FACT(#20,#4,#17);
#200=FACT('is included in','#201,#202);
#201=FACT('is role of','includer','object');
#202=FACT('is role of','included','subject');
#300=FACT(#200,'EXPRESS modelling',#100);

This whole procedure could be repeated for the next five sentences of table 4 and results in a similar listing found in table 7 and 8.

Although we have reduced the redundancy of data considerably, there are still multiple instances of the same terms. This is in conflict with the DEDO requirement. Thus we have to define all the facts and strings only once. For this purpose we introduce a new entity type that can facilitate this requirement. Table 8 is further normalised using DATA as the place-holder for the text encoding.

Table 9 An almost fully normalised dataset

Facts and data
#100=FACT(#20,#4,#17);
#200=FACT(#1000,#201,#202);
#201=FACT(#1001,#1002,#1003);
#202=FACT(#1001,#1004,#1005);
#300=FACT(#200,#1006,#100);
#1000=DATA('is included in');
#1001=DATA('is role of');
#1002=DATA('includer');
#1003=DATA('object');
#1004=DATA('included');
#1005=DATA('subject');
#1006=DATA('EXPRESS modelling');

All the terms within single quotes are encoding of concepts within the scope of a particular language (English). Another requirement of neutral data exchange is that any receiving party, irrespective of the language background can reconstruct the information. Although ISO-10303 allows encoding of the strings in almost any character set this still won't satisfy the DEDO requirement. Suppose we want to use two languages (English and Dutch) in the same exchange set and *e.g.* use the Dutch translation 'is rol van' which has the same meaning as 'is role of'. When table 9 is used and added with the extra Dutch term this could become:

Table 10 Dutch term 'is rol van' added

Facts and data
#201=FACT(#1001,#1002,#1003);
#202=FACT(***#1007***,#1004,#1005); // old one was #202=FACT(#1001,#1004,#1005);
...
#1007=DATA('is rol van');

Although there is no duplication of terms in this set there seems to be a duplication of concepts. Because 'is role of' and 'is rol van' are the same concepts but in a different language. To overcome this, a possible solution is that #1001=DATA('is role of') and #1007=DATA('is rol van') are referring to a language independent 'handle'. A simple but effective solution is the introduction of a unique identifier to which both DATA statements refer by using an identification relation:

Table 11 Adding identification of the concept

Facts and data
#201=FACT(***#1009***,#1002,#1003);
#202=FACT(***#1009***,#1004,#1005);
#1001=DATA('is role of');
#1007=DATA('is rol van');
#1008=DATA('is identification of');
#1009=DATA('1');
#3000=FACT(#1008,#1001,#1009);
#3001=FACT(#1008,#1007,#1009);

The association type of 'is identification of' is not modelled the same way as is done with other association types in the previous examples because one would rapidly run into recursion. Therefore the object and subject side roles of the FACT 'is identification of' are directly pointing to DATA instances instead of other FACTs as in the examples above.

The same identification relation and FACTs based on the same principles can be added for all DATA terms. In the end all FACTs attributes will refer to unique identifiers expressed as DATA('1') .. DATA(<n>) instances, except for 'is identification of'.

All information expressed above seems to be time independent. No explicit expressions can be found that refer to any time concept. In traditional data models and implementations timestamps are found to record either the conception data/time of the record or a timestamp for what is represented by the record or both. Thus it would be logical to add timestamps attributes to both FACTs and DATA entities. At least for the DATA entity this is not necessary because they represent strings that are considered to be always valid. The information is not captured with the DATA but with FACTs. Assume that all the FACTs records are stripped from the examples above or from a hypothetical large dataset. What remains is a long list of strings, representing all words, numbers *etc.* It would resemble the index of a book but without the content or any page references. In principle it doesn't convey any information.

The information in this associative paradigm is stored as FACTs. So can we add the time recording to FACTs and thereby support versioning and time dependent variances on e.g. design data? It would implicate that when a certain fact becomes obsolete at a certain point in time we must have a method to terminate its existence. This requires an extra timestamp attribute to be able to record its termination date/time. One could also simply delete this fact from the information set. However then there would be no way to capture history and go back to earlier versions of *e.g.* design data. It is clear that implementing this solution will not fulfil the lifecycle requirement.

In principle versioning information and the accompanied timestamp series are not aspects of the real or abstract world we are trying to represent with a data set. The time continuum is an integral part of existence because without it there is no existence. Thus if we want to record information about when e.g. the design of particular piece of equipment came into existence we are adding date/time information about its creation process. So let's assume that we want to record that a certain fact from table 11 started its life at March 21st 2001 at 12:34:23.000 then the table could be expanded with the following records:

Table 12 Time information added

Facts and data
#201=FACT(#1009,#1002,#1003);
#4000=FACT('starts life at',#4001,#4002);
#4001=FACT('is role of','object','object');
#4002=FACT('is role of','time','subject');
#4003=FACT(#4000,#201,#4004);
#4004=DATA('2001-03-21T12:34:23.000');

In table 12 the old notation (see table 7 and 8) is used to make it more readable. However full normalisation is assumed. With another set of FACTs and DATA one could also made explicit what type of calendar and time encoding has been used for DATA record #4004. The existence of the #201=FACT has now been recorded and a similar scheme can be used to explicitly record its termination date/time. To make it explicit that start and end of life FACTs are part of a specific snapshot (or version) the same method is used as with inclusion of FACTs in a context. Table 8 shows this for the context of 'EXPRESS modelling' but similar a context for a particular version or even variant can be constructed. When there is a requirement to record the deletion of a FACT between two versions or variants one could achieve that using one of two methods. You can either decide to create a data/time termination FACT in the context of the last version/variant or exclude that FACT using an explicit 'excluded from' FACT in the new version/variant. In both cases no information is lost, history remains intact and can be fully reconstructed. It seems a contradiction but in the associative paradigm deletion means addition of facts about the deletion process.

So even with this fully normalised model and the fulfilment of multilingual expression and versioning capability without violating the DEDO principles can we now actually convey information without any ambiguity? This is surely depending on the way the receiver interprets the association types and the data within the context given. This methodology of full normalisation will result in facts and data defined only once. One can also imagine that descriptions can be added following the same methodology. Although it could support more unambiguous interpretation of the received information one can never rule-out errors.

In the examples above we try to explicitly define information about a subset of the EXPRESS modelling language functionality. However when we have to deal with the explicit definition of all information in a process plants design cycle, including its full design history it would be more than useful to have some kind of reference dictionary. A dictionary that can be used as lookup table for terms and references to minimise the differences between the intended meaning and the actual perceived meaning by the receiver.

The upcoming AP221 standard in its current state has only about 200 entities and 200 different association types. The model has been designed to be very flexible and describes high-level concepts such 'individual', 'class', 'activity', 'physical object', 'event' and 'aspect'. In the AP221 standard however one cannot find *e.g.* entities called 'centrifugal pump', 'vessel' or 'column', pieces of equipment regularly found in process plants. The amount of equipment and part types for which we want to record the existence is currently exceeding 5.000. It would be impossible to add all these 'classes' as explicit entities into the AP221 model. ISO procedures only allow updating of the standard every three years. So intermediate updates and additions must be put in a queue until the next release of the standard. In the mean time however new equipment and parts are invented and introduced on a daily basis.

To allow faster update and distribution of the dictionary of process plant classes it has been decided to maintain the dictionary as a separate register. A set of process plant related classes and definitions have been created in the past six years using the knowledge of 100+ design experts all over the world and currently we recognise more than 17.000 classes. An Internet portal (www.STEPlib.com) facilitates the maintenance and updates of this dictionary better known as STEPlib. The combination of the associative paradigm, a high level model such as AP221 at its base and the solid definition of almost any piece of equipment maintained in STEPlib should facilitate unambiguous exchange of almost any information between partners in the process industries.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 4.4: The CAPE-OPEN Standard: Motivations, Development Process, Technical Architecture & Examples

J.-P. Belaud, B. Braunschweig & M. White

The CAPE-OPEN standard was developed in two successive EU-funded projects, CAPE-OPEN and Global CAPE-OPEN. Some aspects of CAPE-OPEN are presented elsewhere in the book (see chapter 3.3); this chapter gives a broad view of the standard and presents a few examples. The standard itself is available at the www.colan.org website.

We start by setting up the scene with a section on motivations and history. Then we present elements of the standard: its development process (section 4.4.2); its global architecture (section 4.4.3), an example of use (4.4.4) for unit operations and physical properties; and a further look at other portions of the standard, namely numerical solvers and access to physical properties data bases (4.4.5 and 4.4.6); the chapter ends with a short conclusion and a presentation of future activities and of the CO-LaN organisation.

4.4.1 MOTIVATIONS & HISTORY

The CAPE-OPEN project brought together software vendors, global chemical manufacturing companies and universities to solve how enterprise software for designing and optimising process plants could be made interoperable. The CAPE-OPEN project and its successor, Global CAPE-OPEN has had a profound impact on the strategic direction and markets for process simulation software.

At one time, chemical and refining companies built and maintained their own simulators. In the 1970s and 80s almost all of these global companies came to the conclusion that there was no strategic value in developing simulators, and dropped their own proprietary simulator products in favour of a handful of commercial software companies. Maintaining large in-house systems proprietary systems with negligible strategic value became too difficult and expensive for these large companies. This is a similar trend as other enterprise software solutions, including payroll, accounting and human resources, where companies opted not to continue developing in-house software and bought from external software vendors.

By the time CAPE-OPEN was conceived and formed all of the companies on the project had successfully divested their proprietary simulators in favour of commercial versions from companies like Aspen, Hyprotech or Simulation Sciences. The value of the software was in the plant models, thermodynamic packages and solvers that each of the companies had developed to optimise the design processes of their specific plants. Each global company still maintained strategic and proprietary plant models, which were extraordinarily strategic to their businesses. These models were built and used within the earlier proprietary simulation packages and now were integrated into commercial simulators provided by software vendors. The integration process for adding these models was time consuming, painful and expensive. To complicate matters, each software vendor had their own process and approach for integration into their respective system.

As a result of the difficulties these global companies faced integrating these software packages, combined with accelerating software technology trends, the CAPE-OPEN project was launched, to create an open system that would allow software vendors, global process operating companies and universities to define a standard that would allow plug and play of process modelling components across commercial simulation packages. CAPE-OPEN's goal and challenge was to create a single standard for integrating the components of software simulation across commercial software suppliers.

The markets for process simulators, although crucial to design and operation of a petrochemical plant, are mature. The CAPE-OPEN project enabled software suppliers to change the software architecture of their underlying applications and to better address the integration needs of their largest customers. The collaborative and competitive nature of the CAPE-OPEN project was a perfect environment to explore the technical and business direction of process simulation software.

Establishing a business, chemical engineering and software driven standard across competitive software vendors was a revolutionary concept. If an open system for process simulation could be created it would mark the first time that competitive software companies collaborated to define business standards and components across existing enterprise software packages. For the global operating companies, this project was a way to:

- Continue to divest non-core software;
- Influence standards of their domain;
- Reduce cost and time of integrating models and other simulation components;
- Gain efficiencies in process engineering workflow.

For the software company partners, the CAPE-OPEN project was a way to:

- Explore avenues for increasing overall market share;
- Improve integration strategies;
- Move from monolithic application to flexible component architecture;
- Transition to new architecture and technologies with direct user feedback;
- Examine new business models for selling and positioning simulation products.

The CAPE-OPEN project enabled new levels of collaboration across partner networks. Competitors and customers alike worked side-by-side to increase the value of the process simulation software markets. Each software competitor needed to open and discuss their competitive advantage so the market could be expanded for all.

4.4.1.1 The Projects

The CAPE-OPEN project was funded by the European Commission under the Industrial and Materials Technologies Program from January 1997 to June 1999. The partners comprised chemical or petroleum operating companies (BASF, Bayer, BP, DuPont, Elf, ICI), a process licensor (IFP, co-ordinator), major international vendors of process systems tools (AspenTech, Hyprotech and SIMSCI), European academic research groups in the process systems field (Imperial College, RWTH-Aachen, INP-Toulouse) and a software consultancy (Quantisci). The project developed standard interface specifications for unit operations modules, physical and thermodynamic properties systems, numerical solvers, and graph analysis tools.

The second stage of developing an open architecture for process engineering was the Global CAPE-OPEN (GCO) project, operating as an IMS activity involving interregional collaboration. The partnership in GCO gathered an unprecedented setting of highly skilled users, developers and researchers in CAPE. The partners represented 50% of the world users of CAPE software, 90% of the suppliers, and 10 amongst the top 12 research laboratories on the subject worldwide.

The first achievements of the GCO project were demonstrated during the project mid-term meeting hosted by AEAT Hyprotech and collocated with the Hyprotech2000 conference in Amsterdam. Painless interoperability of two commercial environments from Aspentech and AEAT Hyprotech was demonstrated, completed by demonstrations of CAPE-OPEN compliant software components from several companies and universities. New standards for additional components were presented, and the foundation of the CO-LaN was proposed.

The second half of the GCO project developed these technologies further, giving a sound operation basis for the forthcoming CO-LaN initiative. Results of the GCO project were demonstrated at its final technical meeting hosted by Aspen Technologies in Cambridge, UK; the results included full commercial interoperability of unit operations and physical properties from several large and small software companies, software prototypes for new standards, many migrated components from software vendors, universities and end-users.

4.4.2 THE CAPE-OPEN DEVELOPMENT PROCESS

CO and GCO relied on a structured process starting from users requirement and leading to software implementation, test and validation. The process is meant to be incremental, as shown on figure 1, and uses UML (Unified Modelling Language) notation for use case diagrams, sequence diagrams, component diagrams and class/interface diagrams. Interface specification documents contain a phased description of these diagrams, in order to allow future users and developers to better understand the specifications and how they were developed. Examples of UML diagrams and other technical elements are given in section 4.5.3.

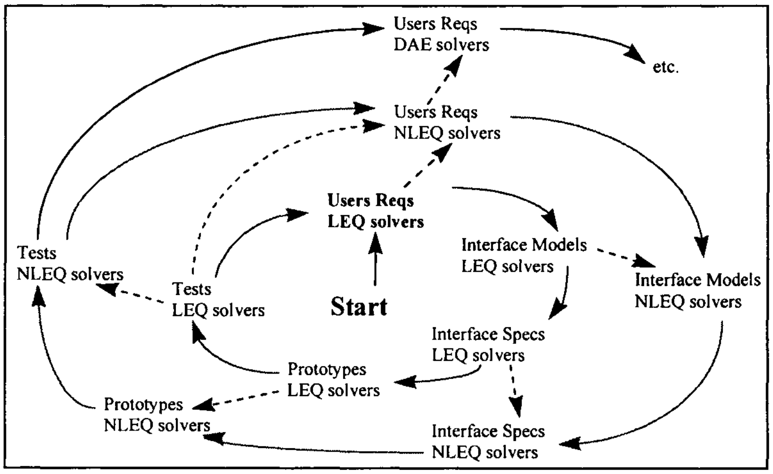


Figure 1: Incremental specification of solvers

In addition, we used an approval system based on an RFC (Request for Comments) process in order to make sure that all documents benefit from contributions from and are agreed by all concerned specialists. The approval process consists in a technical validation phase, followed by a period for commenting, and concluded by formal voting, with a possibility of recycling at any stage in case of failure.

The technical work process that was followed for each interface is presented in Table 1.

Table 1: Work Process Phases

Phase	Step	Goal
ANALYSIS	Users requirements, text	Requirements in textual format
ANALYSIS	Users Requirements, Use Cases	Use Case models
DESIGN	Design Models	Sequence, Interface, Component Diagrams
SPECS	Draft Interface Specifications	Draft Interface specifications in COM and CORBA IDL
OPTIONAL PUBLISH	RFC process	Approval of Draft Specifications
IMPLEMENT	Interface Implementation	Prototype implementation
TEST	Standalone Testing	Interface validation
TEST	Interoperability Testing	Validation on interoperability scenarios
SPECS	Final Interface Specifications	Final specifications in COM and CORBA IDL
PUBLISH	RFC process	Approval of final specifications

The need for such a formal approval process arises from the fact that reaching consensus on complex technical documents can be long and tedious, especially when many organisations with different backgrounds and different goals have to agree. The CAPE-OPEN approval process is based on examples from other international bodies, such as the Object Management Group (OMG), or the Petrotechnical Open Software Corporation (POSC). Compared with the OMG process, the CO process is simplified so that approval delays are shorter; on the other hands it explicitly takes into account the standards development process with software prototypes playing a major part in testing the specifications for interoperability. The process has been successfully applied in the Global CAPE-OPEN project and will be used by the CAPE-OPEN Laboratories Network.

4.4.3 THE CAPE-OPEN STANDARD

This section introduces the current status of the CAPE-OPEN (CO) standard from a technical point of view. After an introduction to the formal documentation, the architecture is mentioned. Then, the CO system is modelled concisely and three examples of CO specifications are given. Although this chapter deals with the version 0.9.3 of CO, up-to-date information on the standard can be found on the official web site [1].

4.4.3.1 CO formal documentation set

The CO standard is characterised by a unique and global version number and is described by a set of documents. Each document has its own versioning number in order to track its own life cycling, however it remains a subset of a CO version. The road map of the CO formal documentation set is shown in the next figure.

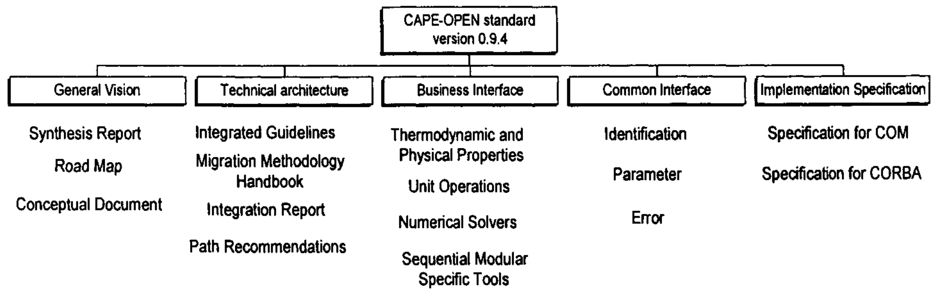


Figure 1 CO documentation set

The formal documentation set includes the following blocks. Altogether they define the current version of the CO standard:

- General vision contains documents that should be read first to get the standard general information such as general requirements and needs.
- Technical architecture integrates the horizontal technical materials and defines an infrastructure aiming at a process simulation based on the CO standard.
- Business interface contains all vertical interface specification documents. These interfaces are domain-specific interfaces for the CAPE application domain. They define CO components involved in a CO process simulation application. In some way these documents are abstract specifications, which create and document a conceptual model in an implementation neutral manner.
- Common interface encloses horizontal interface specification documents for handling concepts that may be required by any business interfaces. This is a collection of interfaces that support basic functions and are always independent of business interfaces. In some way these documents are abstract specifications, which create and document a common conceptual model in an implementation neutral manner.
- Implementation Specification contains the implementation of the business and common interface specifications for a given distributed computing platform. In order to produce CO compliant software components any developer has to use these official interface definitions.

4.4.3.2 CO architecture elements

The CO architecture elements section describes technical objectives and terminology and provides the infrastructure upon which supporting business and common interfaces are based. This input comes from the folder technical architecture (especially the integrated guidelines document) of the CO documentation set. This section identifies the technologies associated with the CO standard, includes the object model which defines common semantics and shows the reference model which embodies the CO interface categories, CO components and communication mode. The CAPE wide-scale industry adoption of this CO architecture provides application developers and end-users with the means to build interoperable simulation software systems distributed across all major hardware, operating system, and programming language environments.

Selected technologies

The following technical decisions are supported by the development of CO. More general information on these methods and tools is provided in chapter 4.2. Key elements are the software system modelling within co-operative working process and the distributed computing accessible over the network (intra/extra/internet).

The CO interfaces are developed and expressed using an object-oriented paradigm. This paradigm is currently the best technical solution for developing interface standards. It also encompasses the "conventional" procedural approach.

The standard relies on the (distributed) component based approach. The standard assumes that a process simulation tool can be made of several components.

The standard uses object-oriented middleware technology, namely Microsoft COM and OMG CORBA that basically carry out the communication. The implementation specifications are available for both middleware. Consequently the standard is independent from implementation languages, and is applicable to several hardware platforms and operating systems. Furthermore, interoperability of CO software components over a network of heterogeneous system is guaranteed.

The standard allows the encapsulation of legacy code, such as Fortran code, to deliver CO software components.

The Unified Modelling Language (UML) is extensively employed for visualizing, specifying, constructing and documenting the artifacts of CO.

CO object model

The object model inherits from the OMG and Microsoft object model. Due to some conceptual inner choices a specific model is built and becomes a CO object model. Its main characteristics are explained in a few words in this section.

The CO object model provides an organised presentation of object concepts and terminology. It defines common semantics for specifying the externally visible characteristics of interface objects in a standard implementation-independent way. In other words, it defines which interfaces, operations, formal parameters, attributes, data types, classes, objects and components are in the scope of CO system. The design of interfaces included in the business and common interface specification documents are built from this CO object model.

Interface: An interface (in fact object class stereotyped <<interface>>) is a collection of possible uses in order to specify through its operations the service of a class. To standardise the interfaces specific to the CAPE domain is one major objective of CO. They are classified as business or common interfaces. At this level, they are designed in an abstract manner, explicitly using the UML notation and the CO object model. English being the base language, the CO interfaces follow the naming convention such as ICapePackageIdentifier within a scope of packages organisation where the global package is CapeOpen.

From the conceptual models of business and common interface specifications, the implementation specifications are expressed in Microsoft Interface Definition Language and in OMG Interface Definition Language.

Object: An object is an instance of a class. An object satisfies a CO interface if it can be specified as the target object in each potential request described by the CO interface. It belongs to the implementation step and so is out of the CO scope. The development of a CO compliant software does not imply the choice of an object-oriented language. Indeed platforms such as COM and CORBA introduce a kind of pseudo objects.

Component: The component is a blob of software that encapsulates the implementation of some business process logic. It is important to distinguish the component and object technologies. The former is a packaging and distribution technology, focusing on what a component can do, while the latter is an implementation technology, focusing on how a component works. Thus a CO compliant component is a piece of software that wraps up the proprietary objects, which realise or use CO interfaces. The communication between CO component instances is defined by the standard.

In order to make the difference between the service provider and the service requestor, the standard distinguishes two kinds of CO software components: Process Modelling Component (PMC) and Process Modelling Environment (PME), the former providing services to the latter. Typically, the PMEs are environments that support the construction of a process model, and allow the end-user to perform a variety of different tasks, such as process simulation or optimisation. The distinction between these two components is not readily obvious. Furthermore, it is worth noting that, in the near future, it will be possible to assemble any number of PMCs to deal with a specific process simulation task.

CO reference model: The CO reference model, illustrated in the next figure, identifies and characterises the components, interfaces, and communication protocols. It includes the middleware component that enables the communication in a distributed environment, the CO components (PMEs and PMCs) and two categories of interfaces (common interface and business interface).

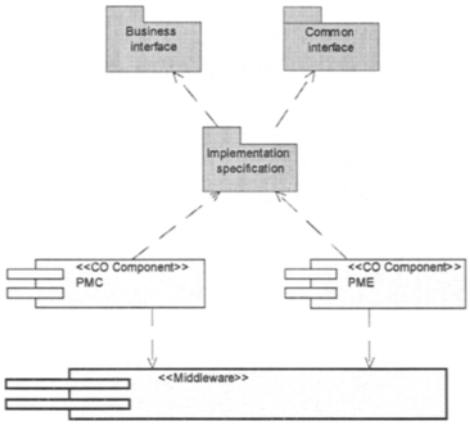


Figure 2 Reference model

The middleware component is the basic mechanism by which objects transparently make requests to and receive responses from each other on the same machine or across a network. It forms the foundation for building open simulation applications constructed from distributed CO components in both homogeneous and heterogeneous environments. According to the selected middleware technologies, the communication protocol is GIOP/IIOP for OMG CORBA and DCE RPC/COM for Microsoft COM.

The business interfaces are domain-specific interfaces for process engineering applications. Within this category, important classes of PMCs are identified such as physical properties, unit operation modules, numerical solvers, and flowsheet analysis tools.

The common interfaces are general-purpose interfaces that can be fundamental for developing useful CO components. They are a collection of interfaces that support basic functions, which allow reusing of design concepts. The adopted CO common interfaces are declined in parameter, identification and error interfaces. Parameter interface defines a parameter service for CO components that wish to expose their mathematical model internal data. Identification interface defines an identification service for CO components that wish to expose their names and descriptions. This information refers to a component-specific instance. Error interface describes how a CO component has to manage an abnormal execution termination. It defines a classification and a hierarchy of errors that may occur during a process simulation.

4.4.3.3 CO standard content

In order to describe the content of the current CO standard, this section exploits the UML model of CO system. After introducing its scope, the different classes of PMCs are detailed, as well as the relations between them and any PMEs. Those can be thought as the physical views of the model. Then, the analysis of packages and their dependencies will be done coming from the logical view.

Scope: Open architectures can benefit many different types of process engineering software. However, the CO standard is specifically focused on general tools for process modelling and, in particular, on their use for steady-state and dynamic simulation. Moreover, the standard recognises explicitly the de facto existence and widespread practical usage of two different types of such tools, namely the modular and equation-orientated ones. Currently the CO standard does not cover file format, physical model content, accuracy of models and data, graphical user interface, process synthesis and computational fluid dynamics.

Extracting CO components: physical view: The next figure illustrates the physical aspects from the component view in the frame of a CO simulation system. It shows the relations of dependency between the different PMCs and a PME.

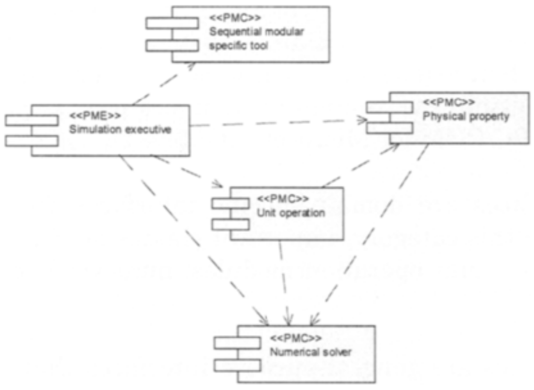


Figure 3 Relations between PMC and PME

The following sections consider in greater detail the actual scope of each type of component defined by CO. It also describes the key concepts underpinning each PMC and its main characteristics.

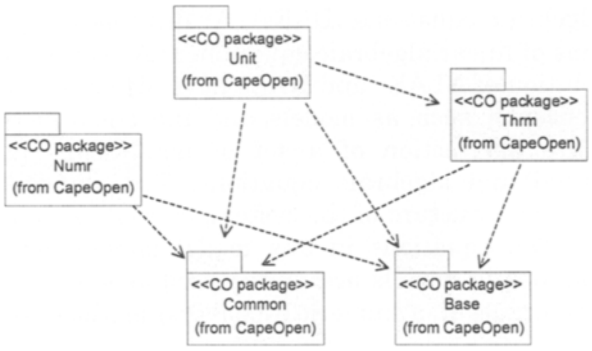
Thermodynamic and physical property component: In the area of physical properties, CO has focused on uniform fluids that are mixtures of pure components or pseudo-components, and whose quality can be described in terms of molar composition. The physical properties operations that have been provided with standardised interfaces are those required for the calculation of vapour-liquid-liquid-solid equilibrium or subsets thereof, as well as other commonly used thermodynamic and transport properties. A key concept in CO is that of a material object. Typically, each distinct material appearing in a process (in streams flowing between unit operations, as well as within individual unit operations) is characterised by one such object. Each unit operation module may interact with one or more material objects. To support the implementation of the above framework, CO has defined standard interfaces for material objects as well as thermodynamic property packages, calculation routines and equilibrium servers.

Unit operation component: CO has defined a comprehensive set of standard interfaces for unit operation modules being used within modular and steady-state PMEs. A unit operation module may have several ports, which allow it to be connected to other modules and to exchange material, energy or information with them. In the material case (which is also the most common), the port is associated with a material object. Ports also have directions (input, output, or input-output). Unit operation modules also have sets of parameters. These represent information, which are not associated with the ports but that the modules wish to expose to their clients. Typical examples include equipment design parameters (*e.g.* the geometry of a reactor) and important quantities computed by the module (*e.g.* the capital and operating cost of a reactor).

Numerical solver component: Here, CO has focused on the solution algorithms that are necessary for carrying out steady-state and dynamic simulation of lumped systems. In particular, this includes algorithms for the solution of large, sparse systems of non-linear algebraic equations (NLAEs) and mixed (ordinary) differential and algebraic equations (DAEs). Algorithms for the solution of the large sparse systems of linear algebraic equations (LAEs) that often arise as sub-problems in the solution of NLAEs and DAEs have also been considered. CO has introduced new concepts, such as models and the equation set object (ESO), which is a software abstraction of a set of non-linear algebraic or mixed (ordinary) differential and algebraic equations. The standard ESO interface enables an access to the structure of the system, as well as to information on the variables involved. The equations in any model may involve discontinuities. Discontinuous equations in models are represented as state-transition networks (STN). A formal presentation of numerical (solvers) methods is given in chapter 3.2..

Sequential modular specific tool component: A key part of the operation of sequential modular simulation systems is the analysis of the process flowsheet in order to determine a suitable sequence of calculation of the unit operation modules. Thus, typically the set of units in the flowsheet is partitioned into one or more disjoint subsets (maximal cyclic networks, MCNs), which may then be solved in sequence rather than simultaneously ("ordering"). The units within each MCN are linked by one or more recycle loops while calculations involving them are converged via the identification of appropriate "tear streams" and through iterative solution techniques. The above tasks are typically carried out using a set of tools that operate on the directed graph representation of the flowsheet. CO has defined standard interfaces for the construction of these directed graphs, and for carrying out partitioning, ordering, tearing and sequencing operations on them.

Organising services: logical view: We study now the standard from the logical view. The standard has a large scope and already defines an important number of interface classes. Furthermore this number will increase along the standard life cycle. As the interface is a structural element, which is too small in a real system such as CO, the standard uses extensively the package concept, which acts as a grouping element. A package forms a logical set of related interfaces with high internal coherence and low external coupling. The division in packages brings many advantages such as an easier management of the system complexity, an improvement of the maintainability and the life cycling. Coherence and independence are the fundamental principles in order to organise the structure of CO packages. It is interesting to notice that a logical package does not assign automatically a corresponding physical component. This is the case for CO where services of a CO component are specified through several packages. When one package, acting as a "client", uses another, acting as a "server", to supply needed services, the client package is said to be dependent on the server package. The structural model of analysis illustrates the dependencies between CO packages. The next figure only displays second level package dependencies (CapeOpen package being the "mother" package).



<<CO package>> Unit (from CapeOpen)
<<CO package>> Thrm (from CapeOpen)
<<CO package>> Numr (from CapeOpen)
<<CO package>> Common (from CapeOpen)
<<CO package>> Base (from CapeOpen)

Figure 4 Package dependencies

The business and common interface folders of the CO formal documentation set contain all the specifications that is under the package CapeOpen.

The logical view that contains the design view and its related structural organisation can be seen as abstract since it remains middleware independent. The implementation specification folder of the CO formal documentation set encloses the design of CO system applied to COM and CORBA. The standard files for COM and CORBA platform are respectively CAPE-OPEN.tlb and CAPE-OPEN.idl knowing that they are distributed as a whole. CAPE-OPEN.tlb is a type library (a compiled version of the Microsoft IDL source) required by the MS-Windows operating system. CAPE-OPEN.idl is a file expressed in OMG IDL. No corresponding compiled version is provided because that would be a strong contradiction to the CORBA objective of implementation independence.

4.4.3.4 Example 1: CO unit operation

In order to illustrate an open interface specification, this section gives some information on a CO unit operation component. The complete data can be found in the CO formal documentation set. The unit operation open interface specification document from the business interface folder is the central contribution.

To look at this specification we will follow the different phases defined by the CO development process through UML diagrams and parts of specification code. No information will be supplied on the implementation and validation phases.

Analysis: This specification describes the unit operation (UO) component of the CO System. The CO unit operation deals with straightforward unit operation using regular fluids in a sequential modular, steady-state simulator. It makes use of a physical properties interface. This behaviour is currently extended to deal with equation-oriented simulation mode and dynamic simulation but this has not been validated yet and so does not belong to the current CO standard.

The next figure shows the type of information passing through public unit parameters and the transfer of stream information between a sequential modular simulator and a CO UO. It describes a part of the system behaviour.

The global variables are essential for the calculation of heat and mass balances (e.g. stream temperatures, ...). They have mandatory names across all CO interfaces and are included in the CO standard. These variables are visible across all CO components.

The public unit parameters (PUPs) are used for control/optimisation (sequential modular simulators) and custom reporting. They are internal variables of a CO UO and are named and made accessible to CO interfaces. Their names are not part of the CO standard. PUPs can also be identified as "read only" if they are calculated by the UO and therefore should not be changed (otherwise a CO error is raised).

- *The latest values of input global and PUPs are requested by and supplied to the UO.*
- *The UO calculates new values of the output global variables and PUPs and passes them to the host simulator.*
- *Requests for derivatives at UO solution are passed to the UO, if required. However, UO supplier is not forced to provide derivative calculation, although, if he does, his UO will be more versatile and marketable.*
- *Derivative values are passed to the host simulator.*
- *During the solution of the UO, repeated thermophysical property requests and responses will occur.*

Figure 5: Some requirements on Unit Operations

Then the user requirements are expressed as a use case model. It identifies the "users" of the system, called actors, and describes in terms of use cases what they wish the system to do. It also identifies the boundaries of the system, which here is a unit operation model of a steady state, sequential modular process simulator. Different use cases categories and priorities are described. Actors are defined such as a flowsheet builder, a flowsheet user, a flowsheet solver or a simulator executive. One must notice that some actors are human ones while other are software ones. The two next figures present a reduced use case map and the description of a use case. Note that the actor flowsheet user usually interacts with the CO unit operation through a graphical user interface provided by the PME. The use case "evaluate a unit" is executed when the user runs the flowsheet. The flowsheet solver actor is responsible for the flowsheet to converge by adjusting the variables in order to reach a specified value criterion.

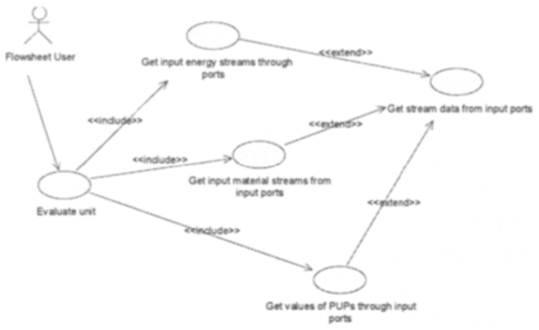

Flowsheet User
Get input energy streams through ports
<<extend>>
Get stream data from input ports
<<include>>
<<extend>>
<<include>>
Get input material streams from input ports
Evaluate unit
<<extend>>
<<include>>
Get values of PUPs through input ports

Figure 6 Reduced use case map

Evaluate unit
Actors: <Flowsheet User>, <Flowsheet Solver>
Priority: <High>
Classification: <Unit Use Case>, <Specific Unit Use Case>, ...
Status: <This Use Case is fulfilled by the following methods: Calculate>
Pre-conditions:
<[Add Unit to Flowsheet] has been used and successfully passed>
<The input and output ports (material, energy or information) have been connected>
Flow of events:
The unit is requested to calculate its required results. The Flowsheet Solver tells the unit to calculate. The unit then requests its unit ports to get its input stream data using the [Get Input Material Streams from Input Ports] and [Get Input Energy Streams from Input Ports] Use Cases, as well as the current values of any required Public Unit Parameters (including its own) using the [Get Values of Public Unit Parameters Through Input Ports] Use Case. If some specific data has been provided by the Flowsheet Builder, but has not yet been retrieved by the unit, the unit gets this specific data. The unit then attempts to perform its calculations. It may also make several requests for physical property ...
Post-conditions:
<The unit finished its calculations successfully or not>
Exceptions:
<The unit may not solve successfully>
Subordinate Use Cases:
[Thermo: Request Physical Properties]
...

Figure 7 Use case description

Design: The CO unit operation design model is described by a combination of text and UML diagrams so as to display the solutions derived for the requirements expressed in the previous section. The relevant UML diagrams are the interface, state and sequence diagrams. In this section, we propose some fundamental diagrams as an illustration.

Within the content of the CO package Unit, the following types are defined:

four interfaces (ICapeUnit, ICapeUnitPort, ICapeUnitCollection, ICapeUnitReport) and their corresponding arrays, two ordered lists of identifiers (CapePortType, CapePortDirection) and their corresponding arrays.

- ICapeUnit is the key interface and handles most of the interaction.
- ICapeUnitPort provides access to a unit port. This in turn gives access to material, energy and information streams provided by the PME.
- ICapeUnitCollection provides a means of merging lists of entities such as parameters and ports.
- ICapeUnitReport provides access to the reporting facilities. This is a basic design in the Unit package. The reporting facility is a candidate as a common interface specification for next releases of the standard. This interface could be removed from the Unit package in the future, being replaced by a link to the Reporting package.

Figure 9 is a simplified interface diagram for the Unit package. This diagram is abstract which means that it is independent of any middleware implementation. The Connect operation description, provided in Figure 9, illustrates how each CO operation is detailed.

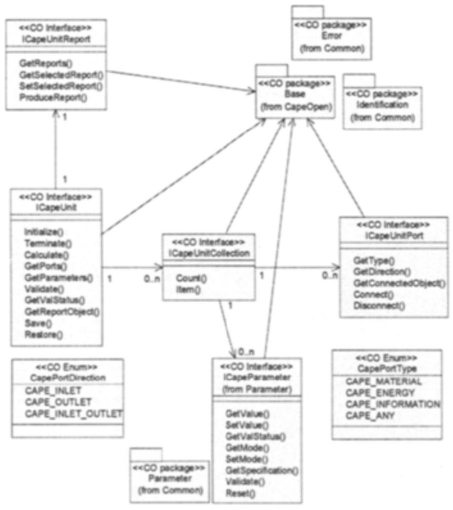
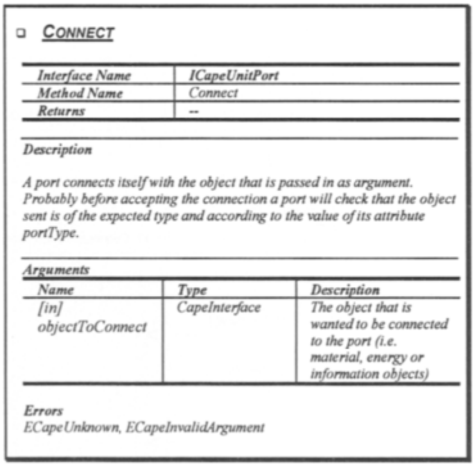


Figure 8 Interface diagram



- CONNECT

Interface Name	*ICapeUnitPort*
Method Name	*Connect*
Returns	--

Description

A port connects itself with the object that is passed in as argument. Probably before accepting the connection a port will check that the object sent is of the expected type and according to the value of its attribute portType.

Arguments

Name	*Type*	*Description*
[in] objectToConnect	*CapeInterface*	*The object that is wanted to be connected to the port (i.e. material, energy or information objects)*

Errors
ECapeUnknown, ECapeInvalidArgument

Figure 9 Description of an operation

The sequence diagram that captures time-oriented dynamic behaviour is greatly employed within all interface specification documents in order to show interactions organised around CO interface objects and their links to one another. The following figure proposes a possible scenario when the Flowsheet User manipulates a CO compliant unit operation through a PME. However let us note that this scenario illustrates only some messages among all the necessary interactions such as to add, create, edit, report, save and restore unit.

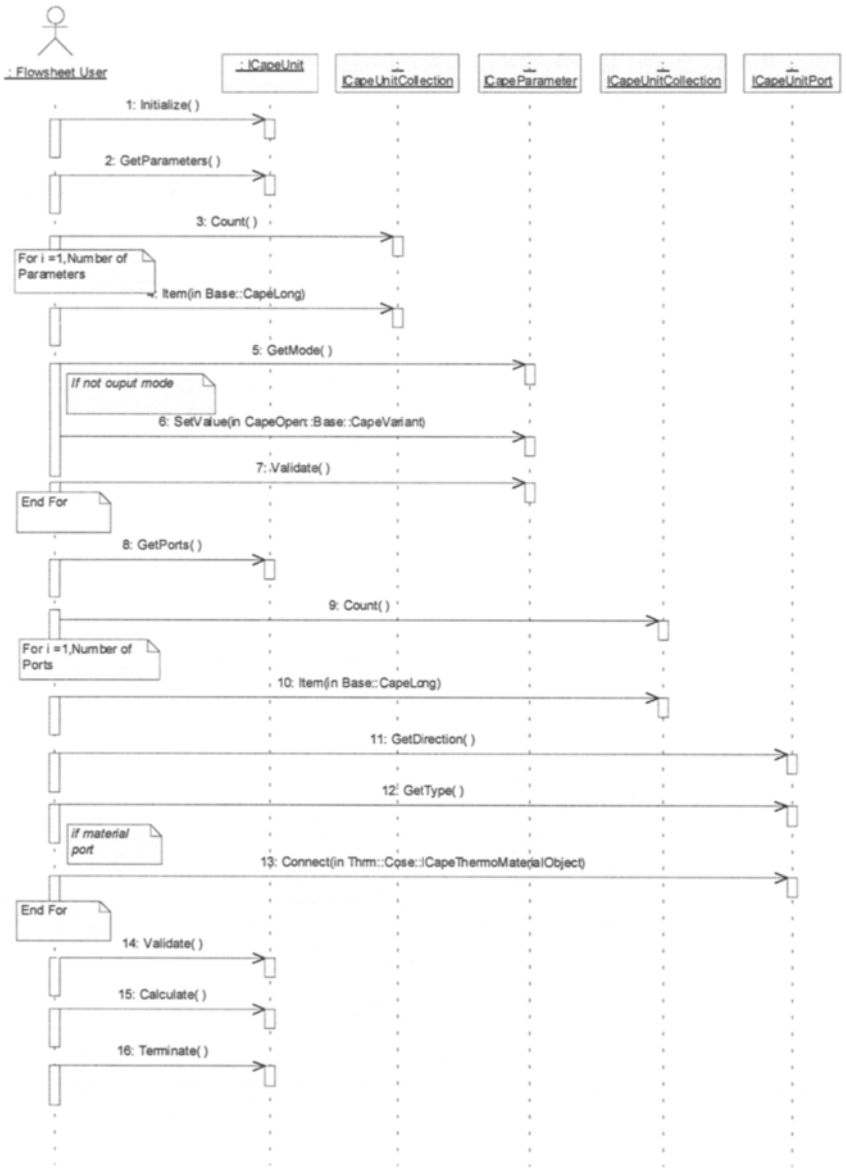


Figure 10 Manipulate unit

Specification: From the previous section, the implementation specifications are written for the Microsoft COM and OMG CORBA distributed computing platform. Due to differences between these two platforms, it is not surprising to get two codes, which are not so similar even if they come from the same abstract design model.

Below you can find Microsoft IDL code lines and OMG IDL code lines. These examples are based on the ICapeUnitPort interface and are extracted from the files CAPE-OPENv0-9-4.tlb and CAPE-OPENv0-9-4.idl in the implementation specification folder. From these extracts, we can notice the different approaches especially in regards to the inheritance scheme and the error handling.

```
[
object,
uuid(ICapeUnitPort_IID),
dual,
helpstring("ICapeUnitPort Interface"),
pointer_default(unique)
]
interface ICapeUnitPort: IDispatch
{
// Exceptions : ECapeUnknown, ECapeFailedInitialisation
[propget, id(1), helpstring("type of port, e.g.material, energy or information")]
HRESULT type([out, retval] CapePortType* portType);
// Exceptions : ECapeUnknown, ECapeFailedInitialisation
[propget, id(2), helpstring("direction of port, e.g. input, output or unspecified")]
HRESULT direction([out, retval] CapePortDirection* portDirection);
// Exceptions : ECapeUnknown, ECapeFailedInitialisation
[propget, id(3), helpstring("gets the objet connected to the port, e.g. material,
energy or information")]
HRESULT connectedObject([out, retval] CapeInterface* connectedObject);
// Exceptions : ECapeUnknown, ECapeInvalidArgument
[id(4), helpstring("connects the port to the object sent as argument, e.g.
material, energy or information")]
HRESULT Connect([in] CapeInterface* objectToConnect);
// Exceptions : ECapeUnknown
[id(5), helpstring("disconnects the port")]
HRESULT Disconnect();
};
```

Figure 11 Implementation specification for COM

```
interface ICapeUnitPort :
Common::Identification::ICapeIdentification {

CapePortType GetType() raises
(Common::Error::ECapeUnknown
Common::Error::ECapeFailedInitialisation);

CapePortDirection GetDirection() raises
(Common::Error::ECapeUnknown,
Common::Error::ECapeFailedInitialisation);

 Thrm::Cose::ICapeThermoMaterialObject
GetConnectedObject() raises
(Common::Error::ECapeUnknown,
Common::Error::ECapeFailedInitialisation);

void Connect(in Thrm::Cose::ICapeThermoMaterialObject
matObj) raises
(Common::Error::ECapeUnknown,
Common::Error::ECapeInvalidArgument);

void Disconnect() raises
(Common::Error::ECapeUnknown);
};
```

Figure 12 Implementation specification for CORBA

4.4.4 EXAMPLE OF USE

This example is reproduced by kind permission of the Global CAPE-OPEN's Interoperability Task Force team led by BP. It uses excerpts of an interoperability scenario developed by this task force.

Initially intended to test the specifications and to achieve the necessary steps towards full interoperability of CO-compliant unit operations and thermo components, the scenario also demonstrates the use of the CO interface specifications in current implementations.

The scenario lists actions and situations that would arise during typical day-to-day use of CO simulation components and CO-compliant Simulator Executives (COSE's). It is primarily concerned with the simulation user's actions, however it also lists software actions that are hidden from the user. Any specific implementation suggestions are included solely for the purposes of illustration.

Software actions are marked in *italic style*. These actions generally correspond to calls to CO interfaces, shown in `computer listing style`. We only show the methods invoked at each step, not their parameters.

For the purposes of this exercise, the following assumptions are made:
The CO unit component used in the tests is a mixer/splitter.
All streams, both inlet, outlet and internal, have the same material template.

Add a CO Unit to the Flowsheet

User selects a CO mixer-splitter from a palette of unit operations and places it on the flowsheet.

COSE asks the unit to initialise itself. (`ICapeUnit::Initialize()`)

The user selects or creates the streams and requests the COSE to connect them to the ports of the unit, preferably in a way consistent with the rest of the flowsheet. (`ICapeUnit::GetPorts(),ICapeUnitCollection::Item(),ICapeUnitPort::GetType(),ICapeUnitPort::GetDirection(), ICapeUnitPort::Connect()`)

Enter Unit Specific Data

The COSE requests the unit for a user interface (UI). If a UI is available, it is displayed. If not, the COSE could attempt to construct one from the unit's list of Public Unit Parameters, or report to the user. (`ICapeUnit::GetParameters(), ICapeUnitCollection::Item(),ICapeParameter::GetValue(),ICapeParameter::GetMode(), ICapeParameter::GetSpecification()`)

User inputs data, including selection of final and monitor reports. (`ICapeParameter::SetValue(),ICapeUnit::GetReportObject(),ICapeUnitReport::GetReports(), ICapeUnitReport::SetSelectedReport()`)

When the input is complete the unit checks its parameters for validity and consistency and reports any errors.
(`ICapeUnit::Validate(), ICapeUnit::GetValStatus()`)

Make Simulation Run

User asks the COSE to calculate the flowsheet.
COSE asks the unit for the reporting options chosen by the user and acts accordingly during the simulation run. (`ICapeUnit::GetReportObject(), ICapeUnitReport::GetSelectedReport()`)

COSE asks the unit to calculate itself. (ICapeUnit::Calculate())

The unit retrieves the materials associated with its streams by requesting them from its ports. (`ICapeUnit::GetConnectedObject()`)

The unit creates an internal material object. This is to contain the mixed feeds. The unit may create an internal array to contain the mixed composition. This is a possible way to minimise calls to the material object to set compositions. `(ICapeThermoMaterialTemplate::CreateMaterialObject())`

The unit requests the mixed stream and all the outlet streams to postpone calculation of their internal states until it has finished defining the compositions and conditions.

The unit retrieves the component flows of each feed from the associated materials and adds them to the mixed stream composition array. When complete, the composition is assigned to the mixed stream material object. `(ICapeThermoMaterialObject::GetNumComponents(), ICapeThermoMaterialObject::GetProp(), ICapeThermoMaterialObject::SetProp())`

The unit fetches the enthalpy from each feed material object, sums them and adds the specified heat input. The total enthalpy is assigned to the mixed stream. `(ICapeThermoMaterialObject::GetProp()),ICapeThermoMaterialObject::SetProp())`

The unit sets the pressure of the mixed stream to the minimum of the feed pressures. `(ICapeThermoMaterialObject::GetProp()),ICapeThermoMaterialObject::SetProp())`

The unit asks the mixed stream to calculate its temperature. `(ICapeThermoMaterialObject::CalcProp())`

The unit assigns molar and enthalpy flows to the output streams in accordance with the specified split factors. `(ICapeThermoMaterialObject::SetProp())`

The unit assigns the composition of the mixed stream to the output streams. `(ICapeThermoMaterialObject::SetProp())`

The unit assigns the pressure of the mixed stream to the output streams. `(ICapeThermoMaterialObject::SetProp())`
The unit requests the output streams to calculate their temperatures. `(ICapeThermoMaterialObject::CalcProp())`

Exit COSE
User saves the simulation.
User exits from the COSE.

Rerun Simulation
User opens the COSE.
User requests the COSE to load the simulation.

COSE detects any changes in the CO components and reports to user.
User makes any edits necessary.
COSE repeats the Simulation Run.

Use a CO Property Package
User creates a CO property package, using a CO-compliant property system, which registers the CO property package.
`(ICapeThermoSystem::GetPropertyPackages())`

User selects the CO property package in the COSE and assigns it to a unit component in the simulation. `(ICapeThermoSystem::ResolvePropertyPackage())`

User Reruns Simulation.

User repeats these actions with the same CO property package assigned to the whole simulation.

User repeats these actions, but first introduces a CO thermodynamic method component, e.g. SRK, into the CO-compliant property system, before creating the CO property package for use in the simulation.

The scenario was successfully demonstrated in several occasions using commercial versions of Aspen Plus / AspenProperties and Hysys. A video file of the software demonstration can be downloaded from the CO-LaN website, see reference.

4.4.5 NUMERICAL SOLVERS IN THE CAPE-OPEN SYSTEM

This section deals with the numerical solvers within the scope of the CO open architecture framework. Thus it will allow you to identify what the CO system can or cannot do for you whatever you are a numerical tools supplier or a user of these algorithms.

The features of the CO solver framework will be described. Then the analysis and design of this framework will be done. Finally some tools built on this framework will illustrate this section.

4.4.5.1 Overview and features

The CO numerical solver framework has focused on the solution algorithms that are necessary for carrying out steady-state and dynamic simulation of lumped systems. In particular, this includes algorithms for the solution of large, sparse systems of non-linear algebraic equations (NLAEs) and mixed (ordinary) differential and algebraic equations (DAEs). Algorithms for the solution of the large sparse systems of linear algebraic equations (LAEs) that often arise as sub-problems in the solution of NLAEs and DAEs have also been considered.

A technical difficulty encountered in this context is the large amount of information that is necessary for the definition of a system of non-linear equations. In fact, this amount increases as more and more sophisticated solution algorithms are being developed. For instance, most modern codes for the solution of large DAE systems require information on the sparsity structure of the system, as well as the ability to compute both the residuals and the partial derivatives of the equations. Even more sophisticated codes need further information on any discontinuities that may occur in the DAE system, the logical conditions that trigger these discontinuities and so on.

To overcome the above problem in a systematic manner, the CO standard has introduced new concepts, such as models and the equation set object (ESO) which is a software abstraction of a set of non-linear algebraic or mixed (ordinary) differential and algebraic equations. The standard ESO interface allows access to the structure of the system (i.e. the number of variables and equations in it, and its scarcity pattern), as well as to information on the variables involved (*i.e.* their names, current values and lower and upper bounds). It also allows the ESO's clients to modify the current values of the variables and their time derivatives, and to request the corresponding values of the residuals and partial derivatives (Jacobian matrix) of a subset or all of the equations in the system.

The equations in any model may involve discontinuities (*e.g.* arising from transitions of flow regime from laminar to turbulent and vice versa, appearance and/or disappearance of thermodynamic phases, equipment failure and so on). Discontinuous equations in models are represented as state-transition networks (STN). At any particular time, the system is assumed to be in one of the states in the STN and its transient behaviour is described by a set of DAEs, which is itself an ESO. Transitions from one state to another occur when defined logical conditions become true; the model interface provides complete access to the structure of these logical conditions as well as allowing their evaluation. Such information is essential for the implementation of state-of-the-art algorithms for handling of discontinuities in dynamic simulation.

Any CO compliant code for the solution of systems of LAEs, NLAEs or DAEs provides a "system factory" interface. Typically, client software starts by creating a model that contains a complete mathematical description of the problem being solved. It then passes this model to the appropriate system factory to create a "system" object that combines an instance of the solver with the model to which the solver will be applied. The system object then provides appropriate operations for solving the problem completely (in the case of a NLAE system) or advancing the solution over time (in the case of DAEs).

The primary aim of the introduction of the ESO and model concepts is to support the operation of CO compliant non-linear solvers. However, an important side benefit is that ESOs also provide a general mechanism for PMEs to expose the mathematical structure of models defined within these PMEs. Thus, it may fulfil the role of "model servers" providing the basis for the development of new types of model-based applications beyond those that are supported by the PMEs themselves.

4.4.5.2 Analysis, design and specification

Analysis

The *Numr* package is subdivided into four packages:

- The *Solver* package focuses on the solution algorithms for the solution of large and sparse systems of linear algebraic equations (LAE), non linear algebraic equations (NLE) and mixed differential and algebraic equations (DAE).
- The *Eso* package contains the ESO concept (which is an abstraction representing a square or rectangular set of equations). These equations define the physical behaviour of the process. An ESO is a purely continuous mathematical description: the equations remain the same for all the possible values of the variables.
- The *Model* package introduces the Model object to embody the general mathematical description of a physical system. The fundamental building block employed for this purpose is a set of ESO. A Model may additionally encompass one or more STNs.
- The *Utility* package contains the public parameter concept which allows some customisation of each CO Solver component.

The following package diagram details the various dependencies between the *Numr* packages. The black arrows within the picture display the relations that are in the CO scope. The standard defines the services proposed by the Solver components. Currently the way one builds an *Eso* or a *Model* within any PME, or accesses them, is not standardised by CO. This task is left to the flowsheeting tool suppliers. However the publication of services to the Solver component is CO standardised.
So, software suppliers, industrials and academics can provide CO compliant Solver, or call numerical services from any CO compliant PMC.

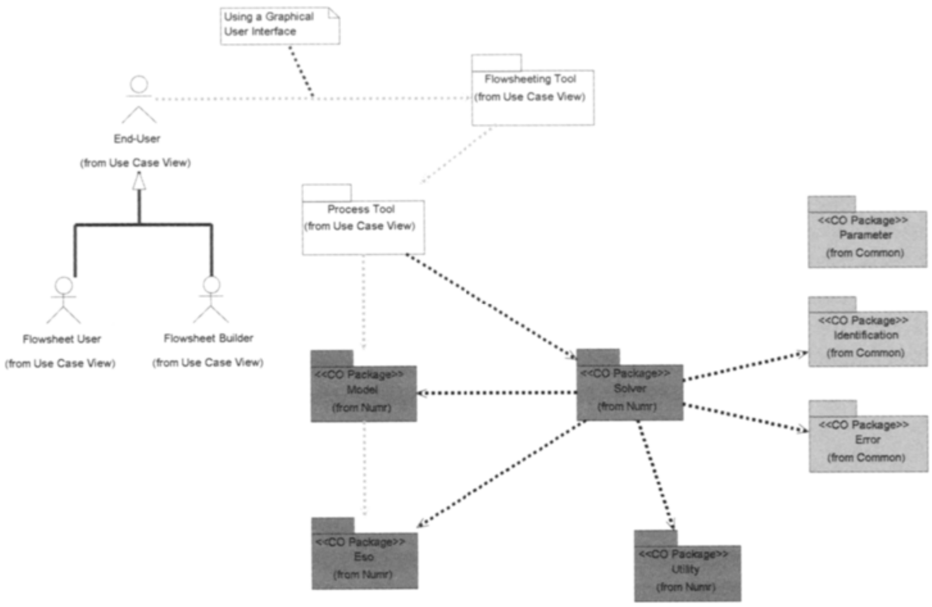

Using a Graphical User Interface
Flowsheeting Tool (from Use Case View)
End-User (from Use Case View)
Process Tool (from Use Case View)
Flowsheet User (from Use Case View)
Flowsheet Builder (from Use Case View)
<<CO Package>> Parameter (from Common)
<<CO Package>> Identification (from Common)
<<CO Package>> Model (from Numr)
<<CO Package>> Solver (from Numr)
<<CO Package>> Error (from Common)
<<CO Package>> Eso (from Numr)
<<CO Package>> Utility (from Numr)

Figure 13 Numerical package dependencies

Design

The *Solver* package, which is responsible for driving the resolution of the problem using all the information from the Model and the Eso, contains the five following interfaces:

- ICapeNumericSolverManager acts as a factory and creates any kind of solver for a specific ESO from a specific type, linear, non-linear or differential.
- ICapeNumericSolver is the base interface of the solver hierarchy and so defines general facilities for identifying the various algorithmic parameters that are recognised by a numerical solver, for altering their values if necessary.
- ICapeNumericLASolver defines facilities, which are specific to solvers of LAE systems. No specific methods have been defined for this kind of solver. It is assumed that the Solve method gets the A matrix and the b vector of the $A \cdot x = b$ system using the already defined methods.
- ICapeNumericNLASolver defines facilities, which are specific to solvers of NLAE systems. It defines methods that allow obtaining and setting convergence tolerance and the number of iterations.
- ICapeNumericDAESolver defines facilities, which are specific to solvers of DAE systems. It defines methods that allow obtaining and setting relative and absolute tolerance.

The next diagram displays the interface diagram of the *Solver* package and its relationship with the *Utility* package.

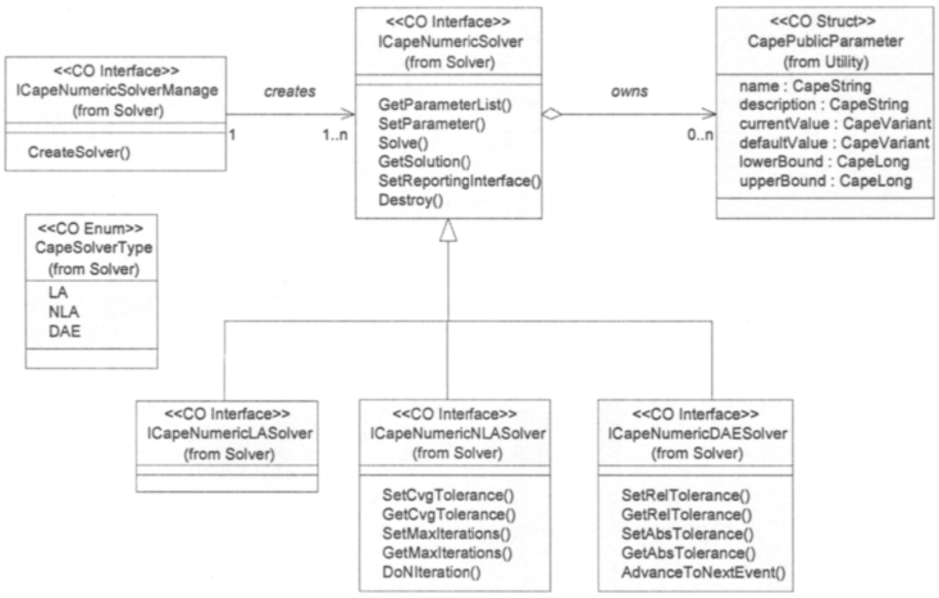


Figure 14 Interface diagram of the Solver package

Specification

The open interface specification document on Numerical Solvers can be found in the *Business Interface* folder from the CO formal documentation set. The implementation specification for COM and CORBA system is enclosed in the CO libraries such as CAPE-OPENv0-9-4.tlb and CAPE-OPENv0-9-4.idl.

4.4.5.3 Implementation and tools

The CO system being new no commercial product based on the CO numerical solver framework is currently available on the CAPE market. Nevertheless we can cite software prototypes from the academics. These tools are more or less validated, they provide important implementation feedback and then they promote proposals for future improvements for the benefit of the entire CAPE community. Already they demonstrate all the advantages of this open software architecture for the next generation of solver tools, such as great interoperability, increased commercial viability for specific algorithms and (web-enabled) distributed calculation.

As examples we can notify the non linear equation solver from RWTH Aachen Germany and the NSP tool from LGC-INP Toulouse France. These components are compliant with the CO system for solvers. According to the previous analysis and design, they realise the *Solver* package depending on the *Model* and *Eso* packages. Following the CO architecture for CORBA system, they act as numerical servers through the Object Request Broker, employ the Identification Common Interface and follows the CO Error Handling strategy. Obviously, these software are separate processes that can be used on a single machine, or across the network within an internet-based enterprise business system.

A PME application, which is compliant with the CO standard, can bind to them in order to resolve its process mathematical model. Some basic CO compliant PMEs have been developed in order to validate the framework, they interact with the above CO solver components playing some test cases such as isothermal flash and Raleigh distillation. In this way they implement and supply the *Model* and the *Eso* interfaces. The next figure illustrates the physical view of these CAPE applications. The two components, PME and Solver PMC, can be on the same host or on a remote host.

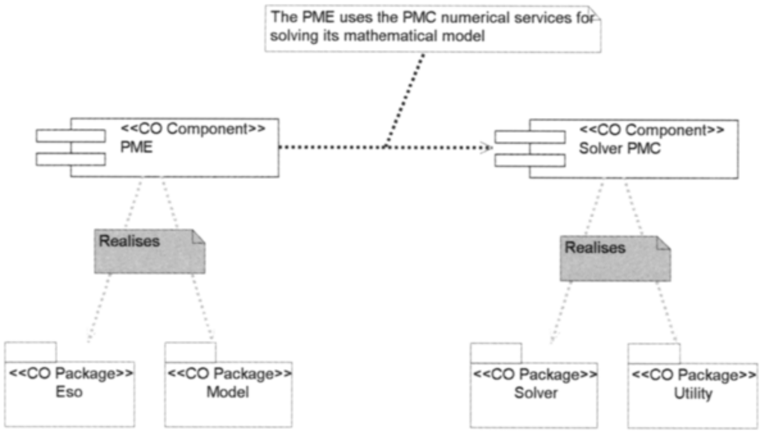


Figure 15 Component diagram

The non linear equation solver from RWTH Aachen comes from the migration of a non linear equation solver developed in FORTRAN into a CO solver component based on the CORBA system. The algorithm NLEQ1s from the Konrad-Zuse-Zentrum für Informationstechnik Berlin has been used as the underlying implementation.

The NSP (Numerical Services Provider) application combines Java and C/C++ codes as well as Fortran 77 legacy codes thanks to the wrapping technique. It supplies the LAE, NLAE and DAE objects, jointly to the configuration parameter objects.

The linear algebraic solver is the result of the wrapping of a FORTAN solver.

The non-linear algebraic solver deals with the system $F(x)=0$. It relies on the Newton-Raphson algorithm and profits from the previous linear solver for solving $\frac{\partial F}{\partial x}(\delta x^k)=-F(x^k)$.

The differential algebraic equation solver manages the system $F\left(t,x,\frac{\partial x}{\partial t}\right)=0$. Its strategy is based on the Gear method with variable order and step.

4.4.6 PHYSICAL PROPERTIES DATABASES IN THE CAPE-OPEN SYSTEM[1]

A physical property data base (PPDB) is an abstract model for all types of collections with thermophysical property data and with parameters of models for calculating thermophysical property data that have been taken from the literature, have been measured or processed in one's own laboratory or have been obtained from other sources. Such a database can be implemented in a variety of physical manners. It can be a relational database, an online service, or a neutral file.

A PPDB consists of data tables, which have a header with the definition of the properties and their units and a body with the numerical values. There is no distinction between dependent and independent variables. Each table is linked to a mixture description and a set of bibliographic specifications.

A PPDB can contain any type of data. However, all properties must fit into a catalogue of properties. This catalogue is part of the CAPE-OPEN standard. It also contains a list of calculation methods, for which model parameters are stored, and the exact form of their equation. This catalogue makes the whole standard very flexible, because additional properties can be added easily, and this means, that the standard will evolve.

In principle, a PPDB can contain any type of pure compounds and mixtures with any state of aggregation: organic, inorganic, plastics, materials (metals and alloys), and emulsions. However, in the first stage of standardization, there is a restriction to regular chemicals. A "regular chemical" is defined as a substance that is listed in one of the registers of the Chemical Abstracts Service (Columbus, Oh, USA) and has a Chemical Abstracts Registry Number (CAS-number). But the standard also accepts substances having no CAS-numbers. For

[1] This section uses selected text from the 1.0 CAPE-OPEN PPDB interface specification, authored by H. Langer et al., available on www.colan.org.

these substances a special enumeration scheme is developed. Especially technical mixtures like mineral oils, heat transfer fluids or victuals (food) have to be taken into account. They should be treated as "pseudo-compounds".

Because a PPDB can have any internal structure, there must exist a general interface for linking PPDBs to user programs that is independent of the internal structure of the PPDB.

The internal implementation of a PPDB interface-set may be complicated, because there are so many different types of PPDBs, but it is not of interest for a client. He needs to see only two objects with the following access methods.

1. a catalogue of the available data bases
   - □ display contents of catalogue
2. an object representing a PPDB
   - □ opening a data base
   - □ closing a data base
   - □ list of available compounds
   - □ list of dictionary information, *e.g.* available journals or stored properties
   - □ search for properties
   - □ obtain overview of tables found
   - □ fetch numerical data and experimental errors contained in found tables
   - □ fetch model parameter contained in found tables
   - □ fetch chemical mixture belonging to found tables
   - □ fetch bibliographic specification belonging to found tables

In order to keep this set of interfaces as simple as possible, model parameters are in many aspects treated as property data.

User programs have two different ways to use data stored in a PPDB:

1. Direct retrieval of the data
2. Retrieval of models or model parameters, which can later be used for calculation property values.

The main interfaces of the PPDB standard in CAPE-OPEN are:

- □ IcapePdbdRegister, knows all about the different types of PPDBs, which are accessible, and makes this knowledge available to the clients. The system administrator and not client users manage it.
- □ IcapePpdb opens and closes the databases, and manages information about their contents: structures, lists of properties and compounds, bibliographic references;
- □ ICapePpdbTables selects tables from the PPDBs and queries them for data; two queries methods, simple and extended, are specified;
- □ IcapePpdbModels queries PPDBs for models and models parameters, in a similar way as IcapePpdbTables queries for compounds data; it also provides simple and extended querying mechanisms;

Finally the PPDB specification needs definitions of properties, units, methods or equations, phase equilibrium information, states of aggregation in order to invoke the interface methods in unambiguous ways: these definitions are part of the CAPE-OPEN standard.

4.4.7 CONCLUSIONS & OUTLOOK

There is no doubt that the CAPE-OPEN project was one of the most successful standardization projects of its kind. The obstacles to success were many and in other enterprise software domains, such as Enterprise Resource Planning (ERP), where similar integration problems were faced, solutions to connect competitive systems such as Baan and SAP, emerged as commercial products from companies such as WebMethods, Extricity, IBM and Vitria. The CAPE-OPEN standard benefited all members, enabling them to have a hands-on and collaborative approach to solving CAPE integration problems that they would have never been able to solve alone.

One of the key reasons why the CAPE-OPEN project was successful was that there was tremendous timing between the software technology curve, the problems faced by engineers designing plants and processes, and the maturity of the CAPE software markets. The collaborative nature of the project provided huge benefits to all of the players.

The key to the future of standards in process engineering and plant design will be how well these open collaboration processes are fostered and developed across emerging technologies and business processes. The first step was taken with the establishment of the CO-LaN, an online collaboration to further develop and refine process-engineering standards. Success in the future for CO-LaN will continue to be measured by how well the CO-LaN bridges the gap between engineering business problems and software technology. Design in general, is playing an increasing role in the business processes of large global corporations, and is being looked to for bottom line results. Consortiums such as Industria, in the process industry and Converge in the semi-conductor industry have been heavily funded to create seamless solutions that link the design process with procurement processes.

Just like a company, and like the CAPE-OPEN project, the more valuable the CO-LaN is to its members, the greater the outlook for success. Increased collaboration across process industry standards organizations will provide huge, and more importantly, measurable business process benefits to all members. CO-LaN will ultimately be measured by the value it brings to its members and how it solves new problems, integrates new technologies, ideas and goals.

Finally, the fact is that people set standards, not organizations. Consensus required across disparate and many times competitive companies is not a trivial feat. Strong leadership, and commitment from many talented individuals combined with a truly open forum of communication led to the establishment of a lasting CAPE-OPEN standard and innovation over fear.

4.4.7.1 CO-LAN

The CAPE-OPEN standard is now under responsibility of the CAPE-OPEN Laboratories Network (CO-LaN), a not-for-profit user-driven organisation for the testing and management of the CAPE-OPEN standard. CO-LaN also helps to facilitate implementation of standard interfaces in commercial software. The missions of the CO-LaN are:

- User priorities for CO standards: work with software vendors to clarify user priorities for process modelling software component/environment interoperability and also to promote communication and cooperation among CAPE software vendors to insure that the CO standards actually translate into commercially valuable interoperability;
- Exploitation and dissemination: promote the CO standard to end-users and distribute CO information and technology internationally;
- CAPE-OPEN specifications life cycle management: organise the maintenance, evolution, and expansion of the specifications;
- Testing, interoperability facilitation: supply compliance testers to support development of components, organise interoperability tests between suppliers of PMCs (Process Modelling Components) and PMEs (Process Modelling Environments)
- Training/Migration facilitation: ensure that training modules, guidelines and tools to facilitate component wrapping are developed and available.

Activities of the CO-LaN are extensively presented on its web site, www.colan.org. Membership is organised in two categories:

- Full members are end-user operating companies: companies who use CAPE tools in their business activities and can be considered as end-users of the CO standard. These are mainly operating companies, process licensing companies, and (not academic) research institutes. End-user organisations pay membership fees. These full members drive the society.
- Associate members are all others: software suppliers, universities, individuals, governmental organisations, etc. These associate members do not have to pay membership fees (but they can always make donations), and have no voting rights. They do, however, participate in the society's activities.

4.4.7.2 Future developments

It is obvious that the CO standard is an evolutionary system. Among other activities, the CO-LaN consortium is in charge of managing its life cycle. The main actions, which could result in a powerful development of the standard, are listed below. Some of them already started but are not finalised when this book is written.

At the technical architecture level, many technologies are or will be studied such as EJB, COM+, the .NET and Web Services facilities, CORBA 3 (with its associated CORBA Component Model). Tools such as compliance testers, wizards for CO component development, and a repository for the standard management are part of the current work in progress.

New business interface specifications are integrated in the 1.0 release of the standard, including physical properties of petroleum fractions, chemical reactions, the above mentioned physical property data base interface, distributed parameters systems, parameter estimation and data reconciliation, optimisation of linear and non linear systems, discrete and hybrid unit operation, connection to real time systems. This version also includes common interface specifications. New implementation specifications according to the previous evolutions will be available.

The CO-LaN will produce an updated formal documentation set identified by increasing version numbers of the standard. This way, the CAPE-OPEN standard will be increasingly reliable, complete and in line with the most up-to-date computing technologies.

4.4.8 REFERENCE

[1] CO-LaN web site: www.colan.org

Part V: Using CAPE-Tools

The chapters of this part highlight, through illustrative examples, the use and/or application of the relevant frameworks (of Part III) and methods, tools & standards (presented in Part IV) to the solution of interesting CAPE problems. It also presents what are the current computer aided tools (process simulators, databases, design, etc.) that are currently available in industry and academia. Contributors from academia as well as industry who are regarded as world experts, have provided input to the chapters in this part.

Chapter 5.1 (Eggersmann et al.) is devoted to the application of models in process systems engineering. The introductory section gives a general overview on model use in different application areas. The general problems encountered during model construction and applications are discussed and highlighted through a case study of model use during the design of a process for polyamide6 production. The design process includes models constructed with spreadsheet, flowsheeting and equation-based modelling tools. After a thorough presentation of the models constructed during the design the role of these tools as used to support the work process is also analysed.

Chapter 5.2 (Bogle and Cameron) discusses the off-line use of CAPE Tools, that is, the process to which the tools are to be applied either does not yet exist, or it exists and on-line data are not readily available. Even though almost all the tools discussed are applicable throughout the product and process lifecycle, the discussion has been limited to off-line simulation, design and analysis only. The discussions have highlighted the business process (activities) in terms of research & development, conceptual design & process synthesis, detailed process design and off-line analysis of collected data.

Over the last two decades, dynamic simulation has matured as a tool for training operators of capital-intensive and safety-critical equipment (such as oil platforms, power stations and nuclear reactors). Dynamic simulators (together with on-line systems) are perhaps the area of CAPE where user *requirements are most stringent. Chapter 5.3 (Cameron et al) describes current practice for the use of dynamic simulators in the process industries where a discussion of dynamic process modelling followed by a brief historical overview of dynamic simulation in the process industries and related areas is given. This is followed by a brief discussion of how and where dynamic simulators can be used in the process lifecycle together with a presentation of the requirements of a specific, but typical industrial operation. Finally, the tools that are currently available are described.*

Hybrid production systems include continuous as well as batch/discrete process characteristics. Supporting computer tools for such systems should therefore comprise software for steady state process simulation and analysis, as well as support tools for the handling of batch process dynamics and discrete decisions involved. Chapter 5.4 (Puigjaner et al.) discuss the software tools for batch processes in terms of two broad classes. The first class of tools involves modelling, simulation, and analysis of the physico-chemical processes that take place during batch operations and may be associated with the process simulation methodology. The second class of tools is designed to support the decision processes at the different managerial and operational levels of the plant operational hierarchy. Puigjaner et al. review a representative set of tools from these two classes that are available commercially.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 5.1: Applications of Modelling – A Case Study from Process Design

M. Eggersmann, J. Hackenberg, W. Marquardt & I. T. Cameron

This chapter is devoted to the application of models in process systems engineering. The introductory section gives a general overview on model use in different application areas. The general problems encountered during model construction and applications are discussed. The main topic of this chapter is a case study of model use during the design of a process for polyamide6 production. The design process includes models constructed with spreadsheet, flowsheeting and equation-based modelling tools. After a thorough presentation of the models constructed during the design stage, the role of these tools as used in order to support the work process is analysed.

5.1.1 PROCESS MODELLING APPLICATIONS

5.1.1.1 Application areas

Process modelling plays an extremely important role within the process engineering lifecycle. That lifecycle consists of many interconnected steps ranging from product conception through feasibility studies, conceptual and detailed process design, operations and terminating with plant decommissioning. Modern lifecycle analysis relies heavily on the use of a wide range of models, some economic, others focused on environmental issues or plant design and performance.

The application of models in process engineering is extensive and continues to grow. Modelling is an application driven activity in that specific outcomes are being sought from the modelling effort. The following table (Table 1) outlines some important categories of model use covering key process engineering activities together with an indication of typical outcomes expected from the models. It is by no means exhaustive.

Table 1. Some key modelling application areas in process engineering

Application Area	Typical model use
Process design	Feasibility analysis of novel designs. Technical, economic, environmental assessment. Effects of process parameter changes on performance. Structural and parametric process optimisation. Analysing process interactions. Design for waste minimization.
Process control	Examining regulatory control performance. Design of model-based control schemes. Model predictive control. (Dynamic) real-time optimisation. Set-point tracking performance for batch operations.
Process operations	Optimal startup and shutdown policies. Scheduling of process operations. Data reconciliation. Fault detection and diagnosis. Operator training simulators. Trouble-shooting plant operations. Environmental impact assessment.
Process safety	Analysis of major hazards and their control. Land-use planning and risk assessment. Review of safety critical systems.

This chapter deals with modelling applications and in particular shows the use of modelling to address issues related to process design via spreadsheets, traditional flowsheet modelling as well as specialized, tailored equation-based models built on the understanding of the physics and chemistry of key unit operations in the process.

In the next section we look at the challenges of industrial modelling and then in subsequent sections the applications of these modelling issues to a process system for polyamide6 production during conceptual design are discussed.

5.1.1.2 Challenges of industrial modelling practice

Modelling within an industrial environment is a major challenge for process engineers. There are many aspects of industrial processes that create substantial difficulties for those who are modelling parts of the complete process lifecycle. Here we touch on some of those important issues. More discussion on these issues can be found elsewhere, *e.g.* Hangos and Cameron, 2001 or Foss *et al.*, 1998.

A modelling methodology

Process modelling consists of an interconnected set of iterative tasks, which direct the modelling effort (Hangos and Cameron, 2001). Amongst the key steps are the statement of the modelling goal, identification of key process characteristics to be captured and the relevant system parameter values, model building and solution, followed by model calibration and validation. Inevitably, the process modeller returns to earlier steps to refine, check, and improve the model to meet the desired outcomes. In what follows we pick up on some relevant issues for further discussion.

Fit-for-purpose

The models, which are built or used, must tackle the goal for which they were built (Minsky, 1965). This clearly puts the onus on the modeller to be clear about what the model must represent and the types of behaviour the model must mimic. A clear functional specification is essential before modelling begins, otherwise, there is no means of determining when the modelling efforts have reached a satisfactory conclusion. Too often an overly complex model is written for what is a simple application task, resulting in lost time, unnecessary effort and possibly major frustrations for the modeller!

Understanding the system under study

In many cases we simply do not understand the underlying mechanisms taking place within the process. We have a "partial" understanding. It is vitally important to the quality of our modelling that we elucidate the key mechanisms within the system. In some cases the complexity is such that the modelling needs to resort to "black-box" or empirical modelling aspects within a more phenomenological framework. In reality, all process models we write are a mixture of fundamental and empirical models – the so-called "grey-box" models.

In many industrial-modelling applications it is important to carry out specific dynamic testing either at laboratory, pilot plant or full-scale level. The aim of this testing can be two-fold. In the first case we might have little knowledge of key mechanisms and we wish to verify the existence of a particular phenomenon or the relative importance of a known mechanism. The second case relates to obtaining key parameter values within the model. These can sometimes be so numerous that a sensitivity study on a preliminary model might be used to rank the importance of the parameters and thereby direct the effort of the parameter estimation task.

Parsimony

This concept addresses the care we must take in having a model that does the job with the minimum information content. The extremes are models that do not contain enough detail to do the job to a model that contains every known system mechanism that leads to unnecessary complexity. The challenge here is to identify up-front the main mass and energy holdups as well as mechanisms such as key reaction kinetics, heat, mass and momentum transfers which need to be captured in the model for its specific use. It is important to properly document these underlying assumptions made about the system so that validation can be performed efficiently. This usually means revisiting the original assumptions and testing their appropriateness. Documentation still remains a major priority in industry (Foss *et al.*, 1998).

Model validation

The validation of process models is a non-trivial issue and one that is often done poorly within industrial modelling practice. There are many challenges here for the process engineer. Dynamic models are often the most difficult to validate and the process data requirements to carry this out are extremely important.

The design of plant testing for validation is difficult and despite the best efforts can be compromised by such problems as inadequate process excitation, poorly calibrated sensors, failure of data-logging equipment and a range of errors from sampling practice, gross data errors and laboratory analysis errors. Even after extensive data treatment the subsequent model calibration and validation can be a time consuming task. However, without adequate calibration and validation for the intended range of application any model is suspect. Further discussion on this topic can be found in Romagnoli and Sanchez (1999).

5.1.1.3 Modelling in Process Design

The specific model application area of this study is process design. Here we discuss some of the key issues, which underpin the development, and use of models within the design task.

The aim of process design is to construct a process that is optimal concerning economics, environmental impact, and safety. To determine the optimal process, several alternatives consisting of different combinations of unit operations employing a variety of physico-chemical phenomena must be examined. This involves understanding and describing the behaviour and interactions of the unit operations and phenomena. To this end experiments and mathematical models can be employed.

The use of mathematical models is preferred because of the possibility of investigating more alternatives in a shorter time compared with an experimentally based procedure. However the construction of models requires experiments for parameter estimation and validation. In contrast with model applications in process control there is no operating plant as a source of data. Process design models are used to predict the behaviour of non-existent processes.

During the design more and more information is generated about different process alternatives and the investigations increase in their detail. This implies the use of different models which range from very simple ones at the beginning to highly sophisticated ones at the end of the design process (see Fig.1). The kind of model depends on the information, which is available, and the purpose it is being used for. Depending on the level of knowledge and detail, today different tools such as spreadsheets, flowsheeting and equation-oriented tools are used for modelling in an industrial setting.

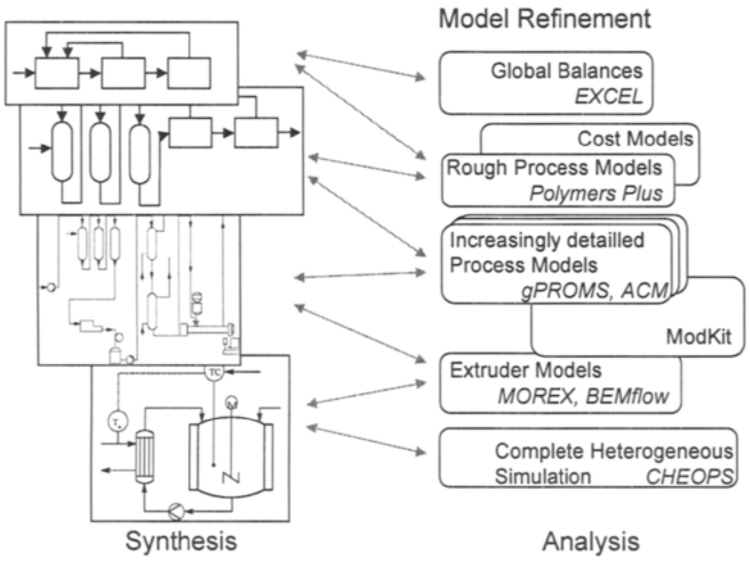


Figure 1. Different models during process design.

Spreadsheet tools are used in the very early stages where only a limited amount of information is available. Examples include rough balances, cost estimations, calculating the economic potential, *etc.* They can be regarded as the pocket calculators of today's engineers or as very simple flowsheeting tools with no predefined models.

When the design proceeds and it has been determined which unit operations might be used, flowsheeting tools are employed to investigate different process structures. Modelling with flowsheeting tools is characterized by selecting and connecting predefined model blocks. These blocks can then be parameterised to describe the behaviour of the process the designer has in mind. The level of detail is fixed by the model block and cannot be easily adjusted by the user.

In the latter stages of the design non-conventional equipment is often encountered. These non-conventional units usually exploit a particular combination of physical, chemical or biological phenomena to fulfil their function. Therefore it is unlikely that a flowsheeting package will provide a suitable model for such units.

There are many cases when standard models from flowsheeting tools and model libraries are not applicable because most of them neglect spatial distributions or assume equilibrium, *etc.* These simplifications might make it impossible to describe the correct behaviour of non-conventional equipment or processes.

A new custom model must be developed in order to study the units under different operating conditions as well as its interaction with other process sections. To cope with the particular characteristics of non-standard units a highly flexible modelling methodology is needed. A set of predefined model blocks will always lack this flexibility. Therefore an equation-oriented approach is usually employed for these tasks.

The next sections describe the use of different models in the course of the design of a polyamide6 (nylon6) process. Knowing that modelling occurs in all stages of the design life cycle, we focus our case study on the conceptual design phase and describe the iterative process between the synthesis of a design and its analysis by the use of mathematical models. Thereby we concentrate on the steady-state models employed. Starting with very coarse models more and more detailed models are used when the design advances and more information is becoming available. In the next section, the polyamide6 process is described from a workflow perspective, which takes into account the people involved, the tools used, and their interaction (see also Chapter 7.1).

5.1.2 DESIGN CASE STUDY: POLYAMIDE6 PRODUCTION

Polymers are materials whose molecular structure consists of long chain-like, branched (and sometimes cross-linked) macromolecules, which are formed from a large number of small monomeric molecules. During the polymerisation reaction the individual monomers are interconnected to a chain.

Polyamide6 (PA6) is a thermoplastic polymer. The world production capacity is currently about 3.3 million tons per year. The most frequent use of PA6 is the production of fibres, which are used in home textiles, bath clothing and for carpet production. In addition, PA6 is used, as an engineering construction material if high abrasion resistance, firmness and solvent stability are required. Glass-fibre reinforced and mineral material-filled PA6 is a preferred material if a combination of rigidity, elasticity and refractory quality characteristics are required.

The task of the case study is to design a process to produce 40.000 t/a of polyamide6 *via* the hydrolytic route. The quality specification of PA6 is as follows:

-
- Residue of ε-caprolactam < 0.1 %,

- Residue of cyclic dimer < 0.04 %,
- Residue of water < 0.01 %,
- Relative viscosity in m-cresol of 2.7.

PA6 can be produced via an anionic and a hydrolytic reaction route. The anionic mechanism is mainly used for special polymers (Hornsby *et. al.*, 1994) whereas the hydrolytic one is more often applied industrially. The structure of PA6 is shown in Fig. 2. PA6 has two different end groups, namely an amide end group and a carboxyl end group, which can react with each other to form longer polymer chains.

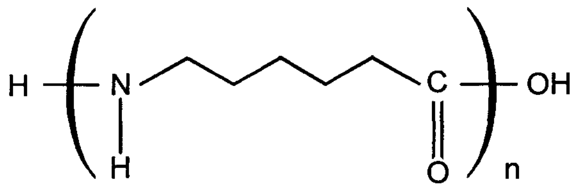


Figure 2. Molecular structure of polyamide6.

The hydrolytic reaction mechanism is described by Reimschüssel (1977) and Tai *et. al.* (1982) and is the one investigated in this case study.

$$CL + H_2O \leftrightarrow ACA \tag{R1}$$

$$P_n + P_m \leftrightarrow P_{n+m} + H_2O \tag{R2}$$

$$CL + P_n \leftrightarrow P_{n+1} \tag{R3}$$

$$CD + H_2O \leftrightarrow P_2 \tag{R4}$$

$$CD + P_n \leftrightarrow P_{n+2} \tag{R5}$$

It consists of three main reactions, which are the hydrolytic ring-opening (R1) of ε-caprolactam (CL) to form aminocaproic acid (ACA), the polycondensation (R2) between two polymer chains (P), and the polyaddition (R3) of ε-caprolactam to a polymer chain. Reaction 4 is the ring-opening of the cyclic dimer (CD), which is formed in an intramolecular reaction of a linear dimer (P_2) (reverse of reaction 4); and reaction 5 is, similar to reaction 3, the polyaddition of P_2 to a polymer chain. In order to control the molecular weight, an organic acid is often being used as stopping agent and stabilizer. The carboxyl-end-group reacts with an amide end-group and thereby one reacting end-group disappears.

The polymer produced has to fulfil certain requirements. The physical properties depend very much upon the molecular weight and its distribution. Especially when spinning the polymer to fibres, purity is a critical factor. Concentrations of water, ε-caprolactam, and cyclic oligomers must not exceed certain limits. These aspects have to be covered by a mathematical model of the process.

In the course of the design process many different models are employed. These start from very simple ones and end with sophisticated ones. Models are a necessary means to study a process' behaviour without having to conduct too many experiments.

5.1.3 PLANT STRUCTURE

5.1.3.1 Input-output structure design

The above mentioned information has been collected in a literature search. This information forms the basis for the following synthesis and analysis steps. At first the input/output structure of the process is synthesized, as shown in Fig 3.

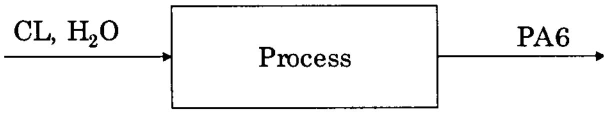


Figure 3. Input-output structure.

This approach is consistent with the design methodology proposed by Douglas (1988). After this synthesis a simple model is needed which allows calculating the amount of PA6 produced from the flow rate of the raw materials. This model only consists of simple mass balance and serves to calculate the approximate amount of feed material needed in order to do a first economic evaluation. The following rationale stands behind this model: When looking at the structural formula of polyamide6 it can be observed that it mainly consists of ε-caprolactam repeating units. A molecule with a degree of polymerization of for example 100 has been formed from 100 molecules of ε-caprolactam and one molecule of water. Hence the amount of water can be neglected as a first estimate. A desired production amount of 5000 t/h of nylon therefore requires the same amount of ε-caprolactam. This allows a first economic evaluation based upon the cost of feed materials and the profits from the product. The calculated value of the necessary ε-caprolactam flow rate will be used as a starting point in further calculations. One may say that this is not yet a model but it has all the characteristics of a model: a simplified, abstract view of a physical system.

5.1.3.2 Recycle structure design

After this analysis the design is refined into a number of major subprocesses. It is known that due to the equilibrium reactions complete conversion of ε-caprolactam cannot be obtained. Therefore a separation of caprolactam and water from the polymer is necessary in order to reach the required specification. The separation is followed by a compounding step, in which fibers or additives are added to the polymer. The resulting flowsheet is shown in Fig 4.

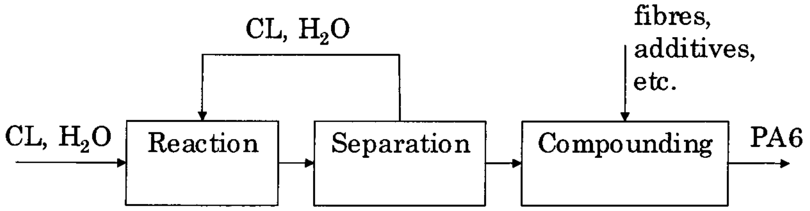


Figure 4.. Recycle structure.

At this stage of the design process it is not reasonable to analyse the complete flowsheet as not much knowledge exists about the reaction and the separation, respectively. Therefore the process units are analysed and refined separately. Because the separation strongly depends on the reaction, it is investigated first.

5.1.4 REACTOR DESIGN

The purpose of modelling and simulating the reactor is at first to understand the reactions. The five simultaneous reactions prohibit an obvious understanding of what is happening inside a reactor. Understanding the reactions means comprehending the influences of different parameters such as conversion and selectivity. Modelling different scenarios provides information, which leads to the best reactor alternatives.

5.1.4.1 Representation of molecular weight distribution

When modelling polymerisation reactions, the representation of the molecular weight is a major issue. Therefore different representation alternatives are presented here. Unlike other substances, a polymer does not consist of one single type of molecule but of similar molecules that are differing in their chain length. In the case of polyamide6 a polymer chain can consist of up to several hundred repeating units. Therefore a polymer can be compared with a multi-component mixture occurring in petroleum processes. Different approaches exist on how to model such systems. One solution is to represent every single component. A decision has to be made about which is the longest molecule to be modelled. The more molecules of varying molecular weight that are taken into account, the more accurate are the calculations going to be, at the cost of a greater numerical effort. For each molecule the chain building reactions have to be implemented. This model can be simplified by grouping similar chain lengths into so-called pseudo components. This reduces the number of components to be described.

Another possibility to represent the molecular weight and its distribution are moments. For a definition and explanation of moments see Dotson *et al.* (1996) and Ray (1972). Moments of molecular weight distribution allow calculations with reduced numerical effort.

Before explaining the modelling of the reactor one model is necessary to transform the polymer specification. This model is not related to the process but to the material. The chain length distribution of the polymer determines its properties, but cannot be measured directly. It is often correlated with the viscosity of the polymer melt. Since the viscosity depends on the shear rate, a common quality specification is the relative viscosity of the polymer in a solvent (*e.g.* m-cresol). This property is not incorporated in the reaction model but has to be transformed into a molecular weight. Respective equations can be found in Saechtling (1995) and Reimschüssel and Dege (1971).

5.1.4.2 Selection of appropriate thermodynamic model

As in any simulation problem in the area of chemical engineering an essential step towards a realistic simulation is the choice of the right method for calculating thermodynamic properties. It is not required for the equilibrium model described below because this model only considers one phase but becomes important as soon as two-phase-behaviour is modelled. The calculation of phase equilibria is required for the calculation of mass and energy balances. Even if it is not directly related to reactor modelling, it will nevertheless be explained here as it first becomes important when the reactor shall be modelled in more detail.

Two general approaches exist: activity coefficient models (ACM) and equations of state (EOS). ACMs correct the assumption of the ideal solution of Raoult's law by introducing an activity coefficient, which depends on the temperature but not on the system pressure. EOS describe the system pressure as a function of temperature, molar volume and composition. These functions allow us to derive most thermodynamic properties. The suitability of these different methods mainly depends on the pressure of the system and the properties of the substances. ACMs can be applied at low and medium pressure up to 10 bar and also when the mixture contains polar components such as alcohols and water. EOSs in contrast can be used at high pressures and temperatures because they take into account the compressibility of liquids and even close to the critical point of the occurring substances (Boki, 1999).

In the polyamide6-case the literature study shows that temperatures between 250 and 300 °C and pressures below 10 bar are to be expected. The main substances are ε-caprolactam, water, ACA, cyclic dimer, and polyamide6. Arguments for the use of ACMs are the presence of water as a polar component and that the temperature and pressure ranges are far away from the critical points. Possible ACMs, which are appropriate for mixtures containing polymers, are Poly-NRTL, Flory-Huggins and UNIFAC. Poly-NRTL describes the phase behaviour of polymer solutions with two binary interaction parameters for each possible combination of substances. Flory-Huggins (FH) uses one interaction parameter for the combination polymer – solvent. UNIFAC is based on molecular structures, and needs group interaction parameters. Poly-NRTL covers wider temperature and pressure ranges than FH and is more exact due to the parameters being specific for each segment. UNIFAC is usable if molecular parameters are not known but may not provide very accurate results. We therefore decided to use the Poly-NRTL approach.

5.1.4.3 Equilibrium model

The first model to analyse the reaction, which takes into account the different reactions, is an equilibrium model (see Fig.5). In our case all reactions are equilibrium reactions. The aim of this model is to understand the equilibrium of the reactions and the effects of changes of temperature, pressure, and feed composition on the molecular weight and the amount of side products produced. For such a coarse model certain assumptions are justified: No kinetics are considered but chemical equilibrium is assumed. This corresponds to a batch reactor with infinite residence time. Evaporation is neglected and hence only one phase is taken into account. With this model no statements are possible about reactor size or residence time. As the polycondensation reaction (reaction 4) is an equilibrium reaction and high water content causes the reaction to proceed in the reverse direction, i.e. the decomposition of polymer. This means that the final water concentration in a reactor has to be below the calculated equilibrium content. It can be seen that each initial water concentration corresponds to a specific molecular weight. The less water is fed to the reactor, the higher is the resulting molecular weight.

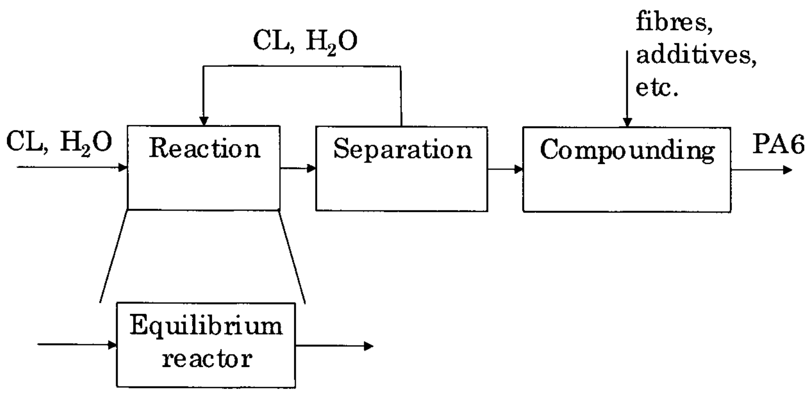


Figure 5. Refinement of reaction section to an equilibrium reactor.

Mallon and Ray (1998) postulate a model for handling the effect of water, which takes into account that the equilibrium constant of the polycondensation reaction varies with the amount of water. Their aim is to develop a model, which is easy to calculate, and comprehensive with a minimum of constants. To this end two states of water are considered: bound (hydrogen bonded to the carbonyl group in nylon polymer) and free (simply unbound water). Their results show good agreement with experimental data.

5.1.4.4 Continuous stirred tank reactor

The equilibrium model assumes a reactor with infinite residence time or respectively infinitely fast reactions. It does not account for different reactor types. In the next step, the reactor is refined into a continuous stirred tank reactor (CSTR) (see Fig. 6).

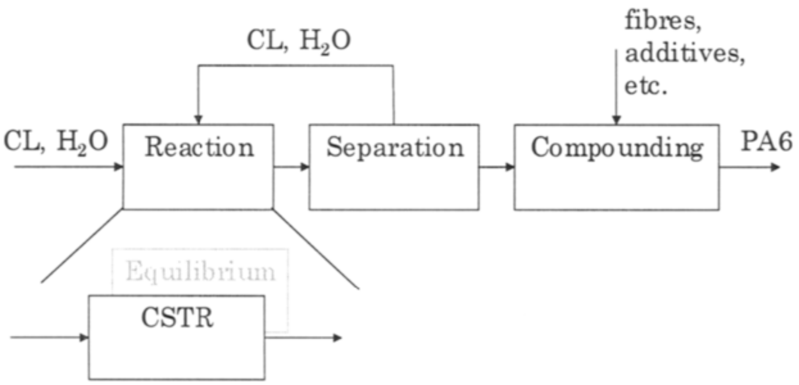


Figure 6. Refinement of reactor to a CSTR.

Modelling the reactor on a higher level of detail means constructing a kinetic model of the CSTR. In a first step only this single reactor was modelled, with the aim to understand the effects of different parameters on a single reactor. In contrast to the equilibrium model this model allows the investigation of the influence of reactor size and residence time. The aim of this model is to get information about the relationships between reactor volume, feed rates, conversion rate, and molecular weight. It takes into account reactions R1 to R5, hence including the formation of the cyclic dimer as a by-product. Knowledge from the equilibrium reactor, especially about the relations between feed composition, reactor temperature and molecular weight were used during simulations of the CSTR.

In our case study this model was implemented using Polymers Plus from Aspen Technology (Polymers Plus, 2001). Even if in Polymer Plus a CSTR model already exists, the specification of the model by the user can be considered a modelling activity. A major task in creating the first model is to describe the reaction scheme, which then can be used in other models, such as when the CSTR is being replaced by another reactor type. In this model every reaction was specified three times: unanalysed, catalysed by carboxyl end groups of polymer and by those of aminocaproic acid. PA6 is being modelled as consisting of amino- and carboxyl-end groups and repeating units. The reaction takes place between two different end-groups, which can either lead to the formation of longer chains or, if the two end groups belong to the same molecule, to a cyclic polymer. Equal reactivity of the end groups is assumed. As this intramolecular reaction is unlikely for longer polymer chains it is not considered for chains with a degree of polymerization of three and above. Only the formation for cyclic dimer is incorporated into the model. As mentioned above, the concentration of this cyclic dimer needs to be minimized. No thermodynamic data are available about the cyclic dimer and the aminocaproic acid. In the model they are considered as non-volatile. This assumption can be justified by the fact that ε-caprolactam has a boiling point of 268°C and that of CD with the same structure and twice the molecular weight must be even higher. Aminocaproic acid (ACA) is an intermediate product that only exists in very small concentrations (due to the kinetic constants). After it is formed it is quickly consumed and can hardly evaporate due to the diffusion barriers of the polymer. The liquid molar volume can be described using the Rackett model. When modelling the reactor volume, Polymers Plus allows different ways of describing the valid holdup phase. Two relevant possibilities are vapour and liquid or only liquid holdup. The first gives improved results during simulation but the second one results in less numerical and convergence problems. Nevertheless the user has to be aware that the assumption of only a liquid holdup phase may lead to wrong results, especially at low pressures, high temperatures or large amounts of volatiles. The study of pressure influence is only relevant if vapour and liquid phases are considered. A possible simplification of this model would be to neglect the formation of cyclic oligomers, which would reduce the complexity of the model but exclude any statement about the content of cyclic dimer. Such a model only makes sense if the content of cyclic dimmer is not important or sensitive to the study.

As ε-caprolactam and cyclic oligomers in the polymer deteriorate the fiber-spinning characteristics, their amount shall be minimized in the reaction. Therefore the production of ACA and cyclic dimer must be represented in the model of the reaction. Higher cyclic oligomers only occur in much smaller concentrations and don't have to be considered. Details about modelling of higher cyclic oligomers can be found in Kumar and Gupta (1997).

We decided not to model the influence of the controlling acid, as this is not yet relevant at this stage of the design process.

Polymers Plus provides good functionality if standard phenomena are to be modelled. The implemented model for polycondensation covers the different reactions, the evaporation of light components and calculates the molecular weight distribution (MWD). The MWD is only determined in terms of the zeroth to second moments, which might not be sufficient for industrial applications.

There are no possibilities of computing the viscosity from the molecular weight. Therefore it is not possible to determine the required mixing power of a CSTR.

5.1.4.5 Plug flow reactor and reactor combinations

In order to investigate different reactor realizations the reactor is alternatively refined to a tubular reactor (see Fig. 7). This design has to be compared with the CSTR. A characteristic difference between the two reactor configurations is the residence time distribution: In the tubular reactor nearly every molecule has the same residence time, whereas in the CSTR a molecule may leave the reactor immediately after entering it. A model of the tubular reactor must allow determining the influence of operating conditions and reactor dimensions on conversion and molecular weight. The modelling of the tubular reactor is done using Polymers Plus because large parts of the CSTR model can be reused. Once the model of a single CSTR has been established, it can be easily extended to describe a plug flow reactor (PFR) or combinations of these two reactors. Therefore the complete reaction kinetics can be kept and only the reactor has to be changed. Parameters like the size and the temperature of the reactor have to be specified. Due to the reaction equilibrium, reactor combinations have a similar performance as a single reactor. The user does not have to deal with the fact that a PFR in contrast to a CSTR is a distributed system. Concentrations, pressure and temperature change over the length of the reactor, whereas in a CSTR complete mixing is assumed. Polymers Plus handles these differences.

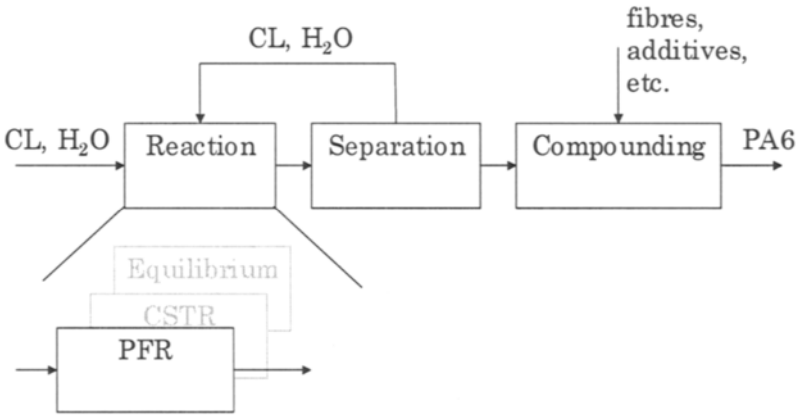


Figure 7. Refinement of the reactor to a PFR.

Polymers Plus offers different possibilities to model the temperature in a PFR: either a constant temperature, a cooling liquid or an adiabatic reactor is assumed. In the case of a cooling liquid, one out of three assumptions (constant coolant temperature, co-current or counter-current coolant) can be selected.

5.1.4.6 Separation between reactors

Simulation studies with these models show that those reactor combinations serve either to attain polyamide6 with the desired molecular weight or an acceptable conversion rate. It is not possible to reach both at the same time. High water content leads to a high conversion of ε-caprolactam but low water content is necessary to achieve a high molecular weight. Therefore a first reactor shall be operated with high water concentrations (to promote conversion) and a second one with low water concentration (to allow a high molecular weight). These considerations lead to the synthesis of a reactor section consisting of two reactors and an intermediate separation to remove water. Hence the current design and model should be extended by an intermediate water separation between two reactors (see Fig. 8).

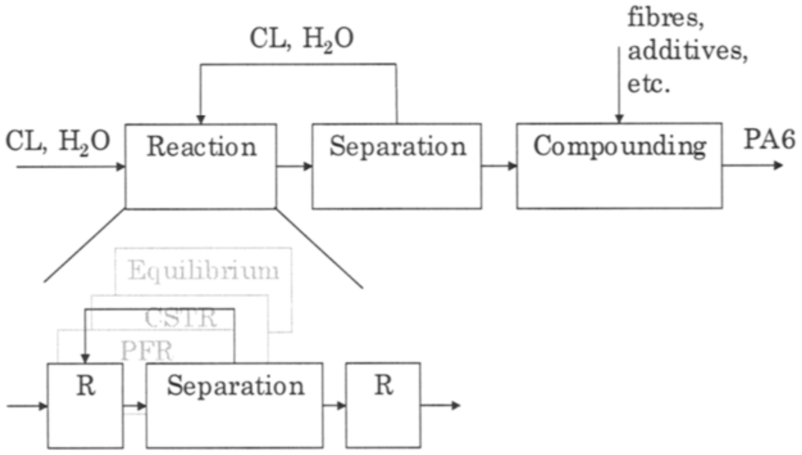


Figure 8. Refinement of the reaction to reactors with intermediate separation.

A first coarse model incorporates a simple splitter, which allows the analysis of the effects of water separation but gives no prediction about a possible realization. In the next refinement step a flash replaces this splitter. The flash model provides information about the feasibility of the separation and the process conditions. Problems of such flash calculations are related to thermodynamic data of cyclic dimer and aminocaproic acid not being available and the high viscosities of the polymer hindering the evaporation of volatiles. No statements can be given about the feasibility of such a unit. The models of the reactors with intermediate separation are implemented in Polymers Plus, as the former reactor models can be reused. More detailed separation models do not exist in Polymers Plus.

VK-tube

The literature study leads to further possible refinement of the reactor design: Industrially, the process is often performed in a so-called VK-tube (VK is derived from the German „Vereinfacht Kontinuierlich" = simplified continuous) (Günther, E. *et. al.*, 1969), a vertical tube with a mixing section at the top and a plug flow section below (see Fig. 9). ε-caprolactam and water enter from the top and polyamide6 leaves at the bottom. Water evaporates in the mixing section and hardly enters the plug flow section. No predefined model exists in Polymers Plus for this type of reactor, so a customized model has to be built with the aim of describing the performance of the VK-tube and to determine its optimal operating conditions.

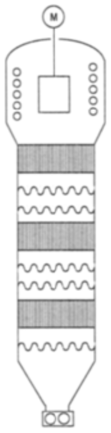


Figure 9. Schematic representation of a VK-tube.

As in Aspen Plus a reactor only has one outlet port, the top section cannot be modelled using a single reactor but has to consist of at least one reactor and one flash. To properly model the effect that the liquid continuously moves down and the evaporated volatiles move up, an infinite number of reactor-flash combinations must be used. The vapour that leaves a flash has to be recycled to the CSTR beforehand. A possible simplification would be just to use two or three of those reactor-flash combinations. The lower part of the VK-tube can be mapped to a PFR model. This configuration is shown in Fig 10. Because it only contains small amounts of volatile components, it can be assumed that it only contains a liquid phase (compare: holdup considerations above).

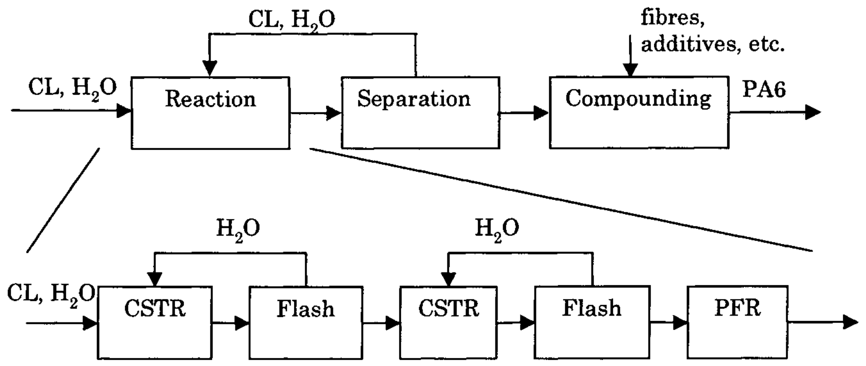


Figure 10. Possible representation of a VK-tube within Polymers Plus.

Recycles

So far, the reaction part of the process has been analysed separately, but as can be seen in Fig. 4 the recycle from the separation section affects the reaction. The equilibrium character of the reactions mean that complete conversion cannot be reached and unreacted ε-caprolactam is leaving the reactor and shall be recycled. Therefore a model is needed, which takes into account the recycles and their effects on the reactor section. The aim of this model is to get a realistic estimation of the process conditions in the reactor. The effects of the recycle loop on the reaction section can only be calculated if separation models are available. By using simple separation models this can be done without further refinement of the separation design. In this case a simple splitter model was used, which allows an estimation of the recycle effects. This model combines a detailed representation of the reactor with a simple version of the separation. It was implemented in Polymers Plus, which allows an easy incorporation of a splitter in the reactor model. This splitter will be refined in section 5.1.5. Here it only serves to know the approximate recycle content in order to revise the reactor section simulations.

5.1.5 SEPARATION DESIGN

The products leaving the last reactor still contain undesired components like water, ε-caprolactam and cyclic dimer. So the next step after modelling and designing the reaction is to refine the separation design and to analyse it by means of modelling and simulation.

In contrast to the reaction section, the separation section is not designed from scratch. Instead a literature study was carried out, which showed two design alternatives for the separation:

1. Devolatilisation in a wiped film evaporator (Riggert and Terrier, 1973; McKenna, 1995),
2. Removal of low molecular weight components by leaching of polymer pellets followed by a drying step (Reimschüssel, 1977).

To proceed with the design these alternatives can be investigated in parallel. They are discussed in the next sections.

5.1.5.1 Wiped-film evaporator

At low pressures and high temperatures water and ε-caprolactam can be evaporated from polyamide6, but due to the high viscosity and resulting low transport coefficients this happens only very slowly. A way to increase the speed is to provide a large polymer surface and to promote internal transport by mixing. A suitable apparatus is a wiped film evaporator (WFE), a vertical tube with wiping blades inside. The polymer mixture enters from above, flows down at the inner walls of the tube, and is wiped by the blades. Volatiles enter the inner gas phase and can be removed. An advantage of this process is that the melt doesn't need to be cooled and heated again.

In order to analyse the technical and economic potential of this apparatus, a model was written, which is mainly based upon a model described by Gesthuisen *et al.* (1998). It consists of three partial models: mass balances, energy balances, and melt transport in axial direction. For more details about the melt transport see McKelvey and Sharps (1979). The occurrence of the reaction was also taken into account. When water evaporates the reaction can further proceed towards higher molecular weights. The construction of this model is not discussed in detail here. Instead we will focus on the construction of a leacher model in the next section.

One possibility to improve the design and to reduce the necessary size of the wiped film evaporator is to support its function by the compounding section. The last part of the process is a polymer processing extruder, which serves to add fibres and additives (pigments, stabilizers, *etc*) to the polymer. In combination with the wiped film evaporator it can also be used to degas remaining volatiles. The requirements for the purity of the polymer leaving the WFE could be therefore reduced. A model describing the extruder must allow the determination of the necessary diameter, the internal design and the power of the extruder as well as the amount of volatiles, which can be removed. Such a model has been realized in the simulation tool MOREX (Haberstroh and Schlüter, 2000).

5.1.5.2 Leacher

Instead of evaporating the monomer from liquid polymer this process removes the monomer from solid polymer. The melt leaving the reactor is cooled and processed to pellets of about 2 mm in diameter, which are then washed with water in a leacher in order to extract unreacted caprolactam, which is soluble in water. The pellets are dried in a next processing step using nitrogen as a drying agent. This process allows higher purities than the devolatilisation. One disadvantage of this process alternative is that the polymer has to be solidified before leaching and again has to be melted during compounding.

The leacher is shown schematically in Fig. 11. It is used to extract caprolactam from polyamide6 pellets. Polymer pellets enter the apparatus at the top. Within the apparatus they form unstructured packing which moves down the apparatus due to the higher density of the pellets compared with water. At the bottom the pellets are removed from the apparatus. The leaching agent water is pumped into the apparatus at the bottom and is removed at the top. The monomer is transferred from the pellets to the water phase during the residence time of a pellet in the apparatus.

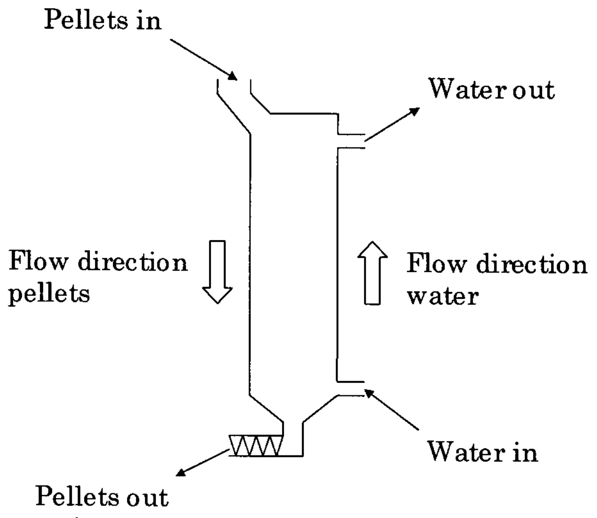


Figure 11. Extraction apparatus for leaching of polymer pellets in water.

Analysis goals

As for the wiped-film evaporator there is no suitable model available for the leacher in a flowsheeting tool. We must develop a custom model, which can serve to determine if this alternative is feasible and which are the operating conditions. Since we start from scratch we must address the requirements of the model in more detail compared with using a block model in a flowsheeting tool.

Since we want to address the technical and economic potential of the leacher the model must be able to describe certain characteristics of the stationary behaviour of the apparatus. First the model must describe the influence of the leaching agent flow rate and composition on the product quality. The product quality is defined by the monomer content of the pellets in the bottom outlet. The influence of the apparatus and pellet geometry on the obtainable product quality must also be studied. The apparatus geometry is necessary to estimate the investment cost for the leacher.

Not only the composition of the pellets is of concern, but also the load of monomer in the leaching agent at the top of the apparatus. This quantity determines the cost of recovering the monomer from the leaching agent.

Key to describing the behavioural characteristics mentioned above will be modelling of the mass transfer of water and caprolactam from the pellet to the liquid phase. Since we do not know this transfer process in particular we assume that the transport process of water and caprolactam within a pellet will be the rate-determining step. However this assumption has to be verified by experiments at later stages of the design process.

Model structure

At first a structure for the model of the leacher has to be chosen. To derive the model structure a method suggested by Marquardt (1995) was used. The resulting structure of the leacher model is depicted in Fig. 12.

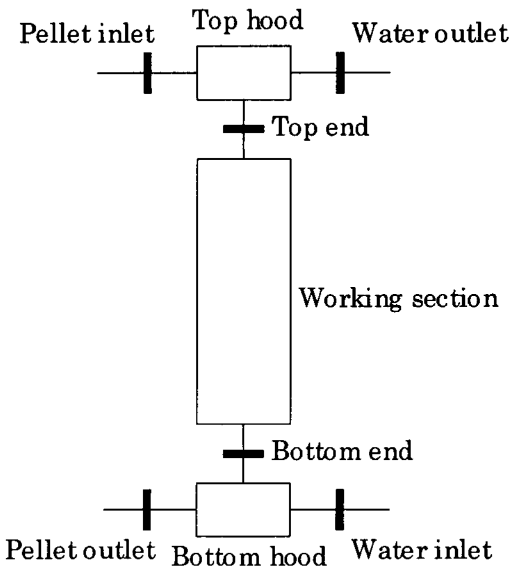


Figure 12. Model structure of the leacher.

The white boxes indicate *devices*. They abstract those parts of the leacher, which have the ability to store an extensive quantity such as mass, energy, etc. in the real world. The black boxes indicate *connections*. They abstract flows between devices. The lines represent *couplings* between devices and connections.

The equations describing the behaviour for the hoods and all connection models are easily derived from global balance equations. Therefore only the derivation of the equations for the working section model is described in the following.

Describing the behaviour of the working section

This section describes how the equation system capturing the behaviour of the working section is derived. The equation system is constructed step-by-step starting from a set of balance equations (Bogusch and Marquardt, 1997). At each step one or more equations are added to the model, which can be interpreted as constraining some of the variables already appearing in the model. The new equations might introduce new variables, which must be refined by other new equations. After each step of adding equations, we analyse the degrees of freedom of the current equation system. For simulation purposes the model is complete if there are enough equations to constrain all occurring variables not considered as inputs or parameters to the model.

5.1.5.3 Momentum transport and resulting flow patterns

To describe the behaviour of the working section some initial assumptions have to be made before we can set up equations describing the water phase and the pellets from first principles. These assumptions are related to the movement of the pellets in the leacher.

At first we establish that the working section is always fully filled with water and pellets. Thus the constant balance volume formed by the cylindrical walls of the leacher will only include a liquid and a solid phase. We do not have to deal with a third vapour phase when representing a partially filled working section.

We also assume that the pellet flow to the working section is high enough and that it is always fully filled with a packing of pellets. Thus we do not have to consider a varying level of pellets in the working section.

Since our model should be used in flowsheet simulations of the polyamide6 process we only need to describe the stationary behaviour of the leacher. This assumption removes all time dependencies from the model.

Concerning the pellets and the packing we make the following assumptions:

- A single pellet occupies a constant volume. This assumption is valid, if the polymer forms a rigid lattice structure. The void volume of the lattice will be used by water and caprolactam molecules. The existence of these molecules should not have any influence on the lattice.
- The packing of the pellets has the same porosity in the whole apparatus. During the movement through the apparatus there is no re-organizing of the pellets.

From these assumptions it follows that some model variables do not vary over the working section length. In particular these are the volume flow of pellets and the cross section of the pellets. These on the other hand determine a constant velocity of the pellets moving through the apparatus.

It is pointed out that the assumptions made in this section can be considered as a simplified model of the momentum transfer phenomena between the pellets, the liquid phase and the vessel. If these assumptions should be fully relaxed we must include momentum balances and assumptions on the momentum transfer mechanisms to the model.

5.1.5.4 Liquid phase

To set up the mass balances the dispersed content of the leacher is considered as a particulate phase system. The mathematical model assumes that at every (mathematical) point within the control volume of the working section there is liquid as well as solids. The smaller the pellets are the more accurate this approximation is. The mass-based partial densities refer to the total volume of liquid and solids.

In the following it is assumed that the liquid phase (index *l*) consists only of water (index *W*) and the monomer (index *M*). There will be one total mass balance for the liquid phase and one for the monomer.

A procedure similar to one presented by Gerstlauer *et al.* (1993) is used to derive formulations of the mass balances. The procedure starts from a general 3-dimensional balance formulation. In a first step the general formulation is simplified by applying assumptions on which fluxes can be neglected. The result is then integrated over spatial coordinates to get lower dimensional balance formulations. At this step, boundary conditions at the border of the integration domain enter the model.

The basis for the derivation of the mass balances is a 3-dimensional stationary balance:

$$0 = -\nabla \cdot \rho_i v_i + j_{p,i} . \tag{1}$$

We have not considered any reactions. $j_{p,i}$ describes the mass transport between the liquid and solid phase. Integration of the mass balance on the cross section of the working section and introduction of mean values on this plane yields:

$$\frac{\partial}{\partial z}\overline{\rho_i v_{z,i}} = \overline{j_{p,i}} \,. \tag{2}$$

At this point we have removed the information on the profiles of the state variables in the radial and azimuthal directions from the mathematical model.

We assume that the *z*-coordinate origin is at the bottom of the working section.

From the formula above we can easily derive the following model equations for the total liquid mass and the mass of monomer in the liquid:

$$\frac{\partial}{\partial z}\overline{\rho_l v_{z,l}} = \overline{j_{p,l}} \quad \forall z \in \Re : 0 < z \le L \,, \tag{3}$$

$$\frac{\partial}{\partial z}\overline{\rho_{M,l} v_{z,M,l}} = \overline{j_{p,M,l}} \quad \forall z \in \Re : 0 < z \le L \,. \tag{4}$$

These two equations are partial differential equations, which have to be augmented by boundary conditions. The boundary conditions are derived from modelling the surfaces at the top and bottom end of the working section.

The mass balances at the top and bottom end of the working section are derived from a general three-dimensional mass balance for interfaces and subsequent introduction of mean values on the cross section areas after integration. In a procedure similar to the one described above this yields:

$$M_{top,l} = -A_{app}\,\overline{\rho_l v_{z,l}}\,|_L \,, \tag{5}$$

$$M_{top,l} w_{top,M,l} = -A_{app}\,\overline{\rho_{M,l} v_{z,M,l}}\,|_L \,, \tag{6}$$

$$M_{bot,l} = A_{app}\,\overline{\rho_l v_{z,l}}\,|_0 \,, \tag{7}$$

$$M_{bot,l} w_{bot,M,l} = A_{app}\,\overline{\rho_{M,l} v_{z,M,l}}\,|_0 \,. \tag{8}$$

Within the working section a closing condition on the mass flows per area holds:

$$\overline{\rho_l v_{z,l}} = \sum_{i \in \{M,W\}} \overline{\rho_{i,l} v_{z,i,l}} \quad \forall z \in \Re : 0 \le z \le L . \tag{9}$$

We can also include closure conditions on the mass fractions at the top and bottom of the leacher:

$$1 = \sum_{i \in \{M,W\}} w_{top,i,l} , \tag{10}$$

$$1 = \sum_{i \in \{M,W\}} w_{bot,i,l} . \tag{11}$$

Since we want to describe the equilibrium at the interface between liquid and solid phase (see section 5.1.5.6) we also have to define the mass fractions of the species in the liquid.

$$w_{i,l} = \frac{\overline{\rho_{i,l} v_{z,i,l}}}{\overline{\rho_l v_{z,l}}} \quad \forall (z,i) \in \Re \times \{M,W\} : 0 \le z \le L \tag{12}$$

This definition holds if we neglect diffusion or dispersion in the liquid phase.

An analysis of the partial model shows that all unknowns can be calculated from the derived equations except the flux quantities representing the mass transport between the liquid and solid phase in the leacher.

Thus in the following we aim at deriving a model for this mass flow. These modelling steps can be considered as deriving a new independent submodel within the overall modelling process.

5.1.5.5 Solid phase

A straight forward way to complete the model would be to include a population balance for the solid phase. An introduction to this kind of balance has been given by Ramkrishna (1985).

However one modelling goal states that diffusion of monomer and water within a pellet is the rate determining step for mass transfer between pellets and liquid phase. Thus we must explicitly model the mass transport of monomer (index *M*) and water (index *W*) in the polymer (index *P*) lattice. This requires that we consider the spatial distribution of monomer and water within a pellet. In terms of a population balance we would need to introduce a particle characteristic, which is described by a distribution instead of a single value like particle size commonly, used in modelling of crystallization processes. Since this is not a straight forward task we have chosen a different approach.

We first derive a dynamic model for a single pellet. This model describes a pellet within a time varying environment. Such a model can be applied to the conditions in the leacher if we consider a reference frame fixed relative to this pellet. The structure of the single pellet model is shown in Fig. 13.

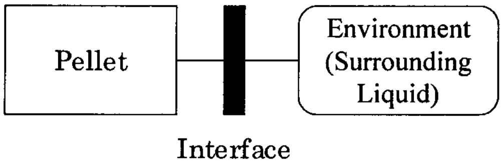


Figure 13. Model structure for single pellet model.

Due to the assumptions we have made on the flow pattern within the leacher we will be able to define a coordinate transformation to merge this partial model with the partial model of the liquid phase already derived in the previous section. That model was derived for a reference frame fixed to the apparatus.

First we set up mass balances for the different species within the pellet. We use the same methodology as for the liquid phase balances described above. This yields:

$$\frac{\partial}{\partial t}\overline{\rho_{i,s}}+\frac{\partial}{\partial r}\left(r\overline{m_{r,i,s}}\right)=0 \quad \forall(t,r,i)\in\Re\times\Re\times\{M,W,P\}:0<t\leq T,\,0<r<R^{1}. \tag{13}$$

These balances describe the dynamics of the mass fluxes inside a single pellet. No reactions are considered. The state profiles across spherical surfaces have been approximated by mean values.

We can refine the total mass fluxes into convective flux and diffusive flux:

$$\overline{m_{r,i,s}}=\overline{\rho_{i,s}v_{r,s}}+\overline{j_{r,i,s}} \quad \forall(t,r,i)\in\Re\times\Re\times\{M,W,P\}:0\leq t\leq T,\,0\leq r\leq R. \tag{14}$$

The diffusive fluxes can be described by the following equations assuming that the diffusive mass transport is due to mass fraction (which relate to concentration) gradients:

$$\overline{j_{r,M,s}}=-\rho_s\left(D_{11}\frac{\partial}{\partial r}w_{M,s}+D_{12}\frac{\partial}{\partial r}w_{W,s}\right) \quad \forall(t,r)\in\Re\times\Re:0\leq t\leq T,\,0\leq r\leq R, \tag{15}$$

$$\overline{j_{r,W,s}}=-\rho_s\left(D_{21}\frac{\partial}{\partial r}w_{M,s}+D_{22}\frac{\partial}{\partial r}w_{W,s}\right) \quad \forall(t,r)\in\Re\times\Re:0\leq t\leq T,\,0\leq r\leq R. \tag{16}$$

[1] Since we consider a pellet in the following the mass-based partial densities refer to the volume of the solid phase only.

The diffusion coefficients in these equations are assumed to be constant (see the above sections for a discussion of this assumption).

The total diffusive flux is zero. Thus we can determine the diffusive flux of polymer species from:

$$0 = \sum_{i\in\{M,W,P\}} \overline{j_{r,i,s}} \quad \forall(t,r)\in \Re\times\Re : 0 \le t \le T,\, 0 \le r \le R\,. \tag{17}$$

The mean of the convective fluxes can be related to the mean value of the partial densities by

$$\overline{\rho_{i,s}} v_{r,s} = \overline{\rho_{i,s} v_{r,s}} \quad \forall(t,r,i)\in \Re\times\Re\times\{M,W,P\}: 0 \le t \le T,\, 0 \le r \le R \tag{18}$$

introducing a mass averaged velocity.

The mass fractions needed to express the diffusive fluxes are defined by

$$\overline{\rho_{i,s}} = w_{i,s}\overline{\rho_s} \quad \forall(t,r,i)\in \Re\times\Re\times\{M,W,P\}: 0 \le t \le T,\, 0 \le r \le R \tag{19}$$

with

$$\overline{\rho_s} = \sum_{i\in\{M,W;P\}} \overline{\rho_{i,s}} \quad \forall(t,r)\in \Re\times\Re : 0 \le t \le T,\, 0 \le r \le R\,. \tag{20}$$

Since we have assumed a rigid lattice of the polymer we have to state that there is no mass flux of polymer species:

$$\overline{m_{r,P,s}} = 0 \quad \forall(t,r)\in \Re\times\Re : 0 \le t \le T,\, 0 \le r \le R\,. \tag{21}$$

Now we have derived a dynamic model for a single pellet. The model forms a PDAE system. Thus we have to define boundary conditions and initial conditions. The boundary condition at the centre of the pellet is given by the assumption that there should be no diffusive fluxes across the centre point. These fluxes would result in asymmetry with respect to the centre point of the pellet. Symmetry requires vanishing mass fraction gradients:

$$\frac{\partial}{\partial r} w_{i,s}\,|_{r=0} = 0 \quad \forall(t,i)\in \Re\times\{M,W\} : 0 \le t \le T\,. \tag{22}$$

A fully specified dynamic model will be obtained if we add equations describing initial mass fraction profiles within the pellet and the dynamic behaviour of the mass fractions at the pellet surface. The latter can be obtained from equilibrium calculations within the interface model if the state of the surrounding liquid is known. Also some physical property calculations have to be included at the system boundaries. We omit these equations at this point.

The model obtained so far can be merged with the liquid phase model if we carry out a coordinate transformation as indicated above. This transformation relies on some assumptions we have made earlier on the nature of the pellet packing. If we consider a single pellet entering the working section at the top the assumptions state that the pellet moves down the apparatus in a straight line with constant velocity. Thus we can state the following relation between place and time:

$$z = L + v_s t \,^{1}. \tag{23}$$

Using this linear relation to transform the pellet model to the spatial coordinate z we get the following formulation for the mass balances:

$$r v_s \frac{\partial}{\partial z} \overline{\rho_{i,s}} + \frac{\partial}{\partial r}\left(r \overline{m_{r,i,s}}\right) = 0 \quad \forall (z,r,i) \in \Re \times \Re \times \{M, W, P\} : 0 \le z < L, 0 < r < R. \tag{24}$$

The transforms of all other model equations can be obtained by exchanging z with t and T with L due to the linear character of the transformation.

The missing initial conditions correspond to the boundary conditions at the top of the working section. These can now be obtained from modelling the top end of the working section. From the total mass balance around the top interface we get an expression for the pellet velocity:

$$M_{top,s} = -A_s v_s \rho_{top,s} . \tag{25}$$

If we assume flat partial density profiles at the entrance to the working section we can derive the following expression from species mass balances around the top end:

$$\overline{\rho_{i,s}} \,|_{z=L} = w_{top,i,s} \rho_{top,s} \quad \forall (r,i) \in \Re \times \{M, W, P\} : 0 < r < R . \tag{26}$$

We can add a closure condition for the mass fractions at the top:

$$1 = \sum_{i \in \{M, W; P\}} w_{top,i,s} . \tag{27}$$

[1] The value of v_s is negative, because the pellets move against the direction of the spatial coordinate z.

Applying mass conservation principles to the bottom end of the working section gives expressions for the total mass flow and mass fractions of the pellet stream leaving the working section:

$$M_{bot,s} = \frac{A_s v_s}{R} \int_0^R \overline{\rho_s} \,|_{z=0}\, dr \,, \tag{28}$$

$$M_{bot,s} w_{bot,i,s} = \frac{A_s v_s}{R} \int_0^R \overline{\rho_{i,s}} \,|_{z=0}\, dr \quad \forall i \in \{M,W\}, \tag{29}$$

$$1 = \sum_{i \in \{M,W,P\}} w_{bot,i,s} \,. \tag{30}$$

5.1.5.6 Liquid-solid interface

An analysis of the model obtained so far shows that for a complete model we are missing the values of the mass fractions at the pellet surface and the mass flows between the phases. We can derive equations for these variables from statements on the liquid-solid interface.

These considerations are governed by the assumption that the mass transfer rate between liquid and solid is determined by the transport processes within the pellet. This allows us to neglect the mass transfer resistance of a boundary layer around the pellet in the liquid phase. Thus we can assume equilibrium between the liquid phase and the solid phase at the pellet surface. As already indicated in the previous section the equations describing equilibrium conditions between the pellet and the liquid form the boundary conditions at the pellet surface. We describe the equilibrium with a single equilibrium constant and the relative volatility:

$$w_{M,l} = K_c w_{M,s} \,|_{r=R} \quad \forall z \in \Re : 0 \le z \le L \,, \tag{31}$$

$$w_{W,l} = K_c \alpha_{c,W} w_{W,s} \,|_{r=R} \quad \forall z \in \Re : 0 \le z \le L \,. \tag{32}$$

Similar to the diffusion coefficients the equilibrium constant and relative volatility will be considered constant in this model (see section 0 for a discussion of this assumption). No equilibrium relation is required for the polymer because we consider the polymer insoluble in the leaching agent.

From mass conservation principles at the pellet surface follow the definitions of the mass fluxes between the liquid and solid phase:

$$\overline{j_{p,i,l}} = \varepsilon a_{pellet} \overline{m_{s,r,i}} \quad \forall(z,i) \in \Re \times \{M,W\} : 0 \le z \le L\,, \tag{33}$$

$$\overline{j_{p,l}} = \sum_{i \in \{M,W\}} \overline{j_{p,i,l}} \quad \forall z \in \Re : 0 \le z \le L\,. \tag{34}$$

These equations form the missing link to the partial model of the liquid phase. The geometric factor a_{pellet} is the specific pellet surface with respect to the pellet volume.

5.1.5.7 Physical properties

An analysis of the model equations derived so far shows that the system is not yet fully specified. The required additional equations can be obtained from physical property calculations at the solid phase boundaries. Compared with the liquid phase model these relations are required because we have explicitly included the convective velocities of the pellets and the monomer and water within the pellet model.

We assume that the polymer forms a rigid lattice, which is filled with monomer and water. Thus we neglect any swelling processes. With this assumption we can derive the following relations:

$$\rho_{top,s} = \frac{\rho_{Polymer}}{w_{top,P,s}}\,, \tag{35}$$

$$\overline{\rho_s}\,\big|_{r=0} = \frac{\rho_{Polymer}}{w_{P,s}\,\big|_{r=0}} \quad \forall z \in \Re : 0 \le z \le L\,, \tag{36}$$

$$\overline{\rho_s}\,\big|_{r=R} = \frac{\rho_{Polymer}}{w_{P,s}\,\big|_{r=R}} \quad \forall z \in \Re : 0 \le z \le L\,. \tag{37}$$

These equations assume the density of the polymer to be constant. Relaxing this assumption is not possible, because of the assumptions we made on the flow pattern within the apparatus in section 5.1.5.3.

5.1.5.8 Additional equations

Analysing the model we will find out that the model is fully specified. However the model will require the cross sectional area occupied by the liquid and solid phase as parameters. These values usually will not be known a priori. Thus we introduce two additional equations, which determine these parameters from the apparatus cross sectional area and the void fraction of the packing.

The cross section occupied by pellets can be calculated by:

$$A_s = (1-\varepsilon)A_{app}\,. \tag{38}$$

The cross section of the liquid phase is given by:

$$A_{app} = A_s + A_l\,. \tag{39}$$

Model parameters

The model derived in the previous section only contains a few parameters. This makes it very suitable for design calculations in the early stages of a process development activity and illustrates the concept of parsimony discussed in section 5.1.1.2.

The model needs several geometric parameters. These include the working section length and radius. From the latter the apparatus cross-section can be calculated. In a design scenario these parameters need to be optimised to achieve an economic process. The model also allows the study of the influence of the void fraction of the packing and the pellet surface per pellet volume. The latter can be calculated from the geometry of a pellet. The packing property can only be determined without experiments if we state additional assumptions on the arrangement of the pellets in the packing.

Crucial to the application of the model will be information on the thermodynamic parameters. These include the polymer density, the diffusion coefficient and the equilibrium parameters.

In general all these properties are functions of temperature, pressure and composition. Since we are only dealing with liquid and solid phases we can neglect the pressure dependencies. In the model we have considered all properties to be constant. This is mainly due to the lack of availability of information on the dependencies. Experience of the authors show that only the polymer density is available from the literature. All other parameters have to be obtained from experiments or estimated. The equilibrium and diffusion parameters might be obtained from the data set of experiments on a single pellet. The pellet model derived above might be used to extract the parameter values from this data set with parameter estimation techniques.

The model can be enhanced if the temperature and composition dependencies of the physical property parameters are included into the model. However due to some of the assumptions in the model this can only change the type and number of model parameters but does not add new effects described by the model.

Exchanging the parameter for the polymer density with a function of temperature and polymer chain length distribution only adds additional parameters to the model. This is due to the fact that the assumption of constant polymer density is the basis for assuming a rigid lattice formed by the polymer in the pellets. Thus from the assumption it follows that the temperature of the pellets must be constant within the whole apparatus. The chain length distribution of the polymer is assumed constant since we do not model any reactions.

If we refine the diffusion and equilibrium parameters with functions of temperature and composition we include a new effect into our model. We can exclude the assumption of constant diffusion and equilibrium parameters and have reduced the number of parameters. The extended model will also predict the effects of changing compositions on the equilibrium and diffusion process. The temperature is still a model parameter as shown above.

5.1.6 TOOL SUPPORT FOR MODELING

This section consists of two parts. In the first part we discuss the experiences obtained while carrying out the case study. The second part exemplifies how advanced tools reported in the literature can support some of the steps related to the presented design and modelling process. For a full overview on the capabilities of the different advanced modelling tools the reader is referred to other chapters of this book (see Chapter 3.1).

5.1.6.1 Experiences from the case study

Aspen Plus with Polymers Plus from Aspen Technology has been used for steady-state simulations. One limitation encountered is that detailed separation models are not available in Polymers Plus, therefore, it cannot be used for a detailed modelling of the process. To overcome this limitation, either a new model for the whole process must be developed using another tool or new Aspen model blocks representing the missing separation units on a higher level of detail need to be implemented. Both alternatives require significant efforts.

In a typical flowsheeting tool the granularity of detail of the model blocks is fixed. The user cannot adjust it and therefore, the accuracy of predictions with the model may be limited and consequently, it may not possible to make decisions based on the on the predictions from these models. For example, it is not possible to calculate third and higher moments of molecular weight distributions, which are relevant for modelling the operations of industrial polymer processes.

A custom leacher model has been implemented with the equation-oriented modelling and simulation package gPROMS. The modelling features of this package can be classified as a *general modelling language* according to Marquardt (1996). Such languages support the hierarchical decomposition of models into encapsulated sub-models. The behaviour of the sub-models is declaratively described by a set of equations and variables. The language does not contain chemical engineering concepts. Thus it is not domain specific. This property limits support in deriving the model equations. However, all advanced modelling systems, which can be classified as *process modelling languages*, *modelling expert systems* and *interactive (knowledge-based) modelling environments* by Marquardt (1996) have never been reported in the literature as being in regular *industrial* use. Thus the choice of tools can be considered to be of industrial relevance.

Major parts of modelling the leacher had to be carried out with pen and paper or in the head of the modeller due to the lack of chemical engineering concepts within gPROMS. However there are mappings of the results of the modelling steps to concepts within the modelling language. Decisions on the model structure will be reflected in UNIT, STREAM and EQUATION subsections of MODEL sections in the gPROMS input file. There is no support to document the derivation of a particular model equation such as the balance equations. The gPROMS language only represents the results of these steps in the EQUATION subsection of a MODEL section of a gPROMS input file.

5.1.6.2 Support with advanced tools

To overcome the problem that flowsheeting tools cannot always provide suitable model blocks for all applications, efforts have been undertaken to integrate flowsheeting and equation-oriented tools. The Aspen Engineering Suite allows the migration from flowsheeting to equation-oriented modelling environments but not in the other direction. Thus the numeric robustness of the Aspen Plus blocks cannot be exploited in detailed simulation studies.

A different approach is being developed in the Cape-Open (Braunschweig *et al.*, 2000) and Cheops (Wedel and Marquardt, 2000) projects, both of which aim at providing a platform to connect different tools to one central process simulator.
Currently, the case study presented above is being extended in terms of efforts to combine the Polymers Plus reactor model, the gPROMS separation model and the extruder model within the special purpose tool called MOREX that will enable an overall simulation. To this end Cheops, a component-based hierarchical process simulator is being used to integrate the different models.

The modelling goals, as stated above, were not achieved either by the flowsheeting tool or by the equation-oriented tool for the highlighted case study. In an industrial setting documentary information, such as the modelling goals, will either be kept only in the mind of the modeller or will be stated in a report on the design process. However the report is not linked electronically to the model files. Thus it might get lost easily leaving a future user without the necessary model documentation.

One approach to overcome these limitations is implemented in ModKit (Bogusch *et al.*, 2001). ModKit provides a hypertext system to document all parts of models. Each modelling object can be linked to a hypertext page, which includes informal documentation of a model. The hyperlinks do not only refer to modelling objects but also to other pages within the set of documentation pages.

Development of a suitable model structure for an equation-oriented model is a task, which requires the modeller to abstract the reality. Since a modelling tool has no representation of the reality this process can only be supported indirectly. The modelling tool is unable to make suggestions on how the model structure should look like. Instead the only help currently possible is to provide a set of well-defined canonical structural modelling objects, which can be used by the modeller to construct the model structure in a flexible manner.

In modelling the leacher we have already used a set of such modelling concepts from the methodology proposed by Marquardt (1995). However gPROMS neither forces nor supports the modeller to use such concepts. In particular there is no graphical representation of the model structure. The ModKit system provides a graphical model structure editor, which provides the canonical modelling objects on a palette. The modeller can select these objects from the palette and construct graphically the model structure. Other tools, *e.g.* ModDev (Jensen, 1998) and MODEL.LA (Bieszcad, 2000) define different sets of canonical objects, which can be employed by the user to define the model structure. However these objects are not organized in a hierarchical class structure.

In the process of deriving the model equations a successive refinement approach was used in the case study. Starting with the balance equations of the liquid phase, the model was refined by adding equations describing variables appearing in the liquid phase model. Subsequent model analysis steps (e.g. after finishing the liquid phase model) were used to determine if the model was already fully specified. During this process a model for a single pellet has been developed independently from the original problem specification. This model was then fitted into the overall model through a mathematical transformation step.

Currently there are no tools to support derivation of model equations in such a general and flexible way based on the mathematical structure of the intermediate model. Advanced tools for the derivation of unit operation models (Jensen, 1998 and Bieszczad, 2000) have chosen a different approach. These tools provide a set of dialogs, which help the modeller to specify the characteristics of a general reference model in terms of chemical engineering concepts. Bieszczad has introduced the term phenomena-based modelling for this purpose. From the user provided specifications, these tools are able to generate (derive) the model equations automatically.

The leacher example, however, shows that such an approach is not flexible enough to support modelling of non-conventional equipments. In the case study, the deviation from a general reference model was necessary in order to avoid an unnecessarily large and mathematically complex model (one employing a population balance). MODEL.LA does not allow the user to deviate from the reference models, which are encoded in "modelling logic operators". ModDev on the other hand has a subsystem ModDef that allows the user to specify new model blocks. The equations from these templates can be added to the automatically generated equations. The templates are organized in a hierarchy, which can be searched for model blocks with specific properties.

5.1.6.3 Conclusions

The case study showed that the flowsheeting as well as the equation-oriented approaches to modelling of processes are relevant to process design. However, there are no industrial tools available, which is able to provide the full flexibility needed for a seamless use of both approaches in order to perform a successful simulation of the process.

Custom modelling tools that are in industrial use today only provide very limited support in modelling of non-conventional equipments. Tools based on phenomena-based approaches, however, may be viewed as transferring the modelling approach employed by flowsheeting tools such as AspenPlus to the unit operation level. In flowsheeting tools, models are constructed from a set of predefined unit operation blocks. The phenomena-based tools construct unit operation models from a set of predefined phenomena models. This moves the modelling process away from dealing with equations and introduces chemical engineering concepts.

The case study, however, has highlighted some limitations of these advanced modelling tools with respect to flexibility and extensibility. Since there is not a single theory that may be employed to derive the necessary model equations, the equation generation process should be more customisable by an expert user. The modelling tools must provide facilities to easily define new phenomena and associated models, which may be obtained from data-driven approaches or new theories. It might be concluded that the phenomena-based tools provide support for a large new class of chemical engineering modelling problems on the unit operation level, but there is still need for a more flexible and extensible approach to model non-conventional operations as well as processes.

5.1.7 SUMMARY

This contribution has provided a case study illustrating the application of modelling in the conceptual design stage of a chemical process producing polyamide6. The case study has focused on modelling at the flowsheet level and at the unit operation level. The use of computer-aided tools during the modelling process has been discussed with respect to the case study.

The nature of the modelling problems during the early stages of a design process required that the developed models incorporated information available in the open literature, because major pilot plant experimental studies generating them would be highly unlikely so early in the design process.

The focus on the early stages of the design process is also the reason why model validation was not covered in this contribution. The reader is referred to the other chapters of this book (see all the chapters of Part III) for related topics.

5.1.8 ACKNOWLEDGEMENTS

We appreciate support for the model development described in sections 3 and 4 of this document from the Deutsche Forschungsgemeinschaft funded Collaborative Research Center 476 – IMPROVE at RWTH Aachen.

5.1.9 REFERENCES

Bieszczad, J., *A framework for the language and logic of computer-aided phenomena-based process modeling,* PhD thesis, Department of chemical engineering, Massachusetts Institute of Technology (2000)

Bogusch, R., B. Lohmann and W. Marquardt, "Computer-aided process modeling with ModKit" *Comput. Chem. Engng.* (2001)

Bogusch, R. and W. Marquardt, "A formal Representation of Process Model Equations" *Comput. Chem. Engng.,* **21**, 1105-1115 (1997)

Bokis, C.P., H. Orbey and C.-C. Chen, "Properly model polymer processes" *Chemical Engineering Progress,* **4**, 39-51 (1999)

Braunschweig, B.L., C.C. Pantelides, H.I. Britt and S. Sama, "Process modeling: The promise of open software architectures" *Chem. Eng. Progress,* **96** (9), 65-76 (2000)

Dotson, N.A., R. Galván, R. L. Laurence and M. Tirrell, *Polymerization Process Modeling,* VCH, New York (1996)

Douglas, J.M., *Conceptual Design of Chemical Processes,* McGraw-Hill, New York (1988)

Foss, B.A., B. Lohmann and W. Marquardt, "A field study of the industrial modeling process" *J. Proc. Control,* 8 (5,6), 325-338 (1998)

Gerstlauer, A., M. Hierlemann and W. Marquardt, "On the Representation of Balance Equations in a Knowledge Based Process Modeling Tool" *Proceedings of CHISA'93,* Prague, Czechoslovakia (1993)

Gesthuisen, R., S. Engell and C. Schmidt, "Simulation der Vakuum-entlactamerisierung von Polyamid-6-Schmelzen in Dünnschichtverdampfern" *Tagungsband UMSICHT Tage,* 15-16 September, 1998, Fraunhofer UMSICHT, Oberhausen, Germany (1998)

gPROMS introduction, http://www.psenterprise.com/gPROMS/index.html (Accessed 12. April 2001)

Günther, E., et. al., "Verfahren und Vorrichtung zum kontinuierlichen Polykondensieren von Lactamen" *German Patent 1495198* (1969)

Haberstroh, E, and Schlüter, M., "Moderne Technologien bei der Entwicklung von Simulationswerkzeugen" In: *Tagungshandbuch 20. Kunststofftechnisches Kolloquium des IKV,* 7, 4-7, Verlag Kunststoffinformation, Bad Homburg, Germany (2000)

Hangos, K.M., and I.T. Cameron, *Process Modelling and Model Analysis,* Academic Press, London (2001)

Hipp, A. and W.H. Ray, "A dynamic model for condensation polymerization in tubular reactors" *Chemical Engineering Science,* **51**, 2, 281-294 (1996)

Hornsby, P.R., J.F. Tung and K. Tarverdi, "Characterization of Polyamide 6 Made by Reactive Extrusion. I. Synthesis and Characterization of Properties" *Journal of Applied Polymer Science,* **53**, 891-897 (1994)

Jensen, A. K., *Generation of problem specific simulation models within an integrated computer aided system,* PhD thesis, Department of Chemical Engineering, Technical University of Denmark (1998)

Kumar, V.S., and S.K. Gupta, "Modeling of Higher Cyclic Oligomer Formation in Nylon 6 Polymerization" *Ind. Eng. Chem. Res,* **36**, 1202-1210 (1997)

Mallon, F.K., and H.W. Ray, "A Comprehensive Model for Nylon Melt Equilibria and Kinetics" *Journal of Applied Polymer Science*, **69**, 1213-1231 (1998)

Marquardt, W., "Towards a process modeling methodology" In: R. Berber (Ed.), *Model-based Process Control*, Kluwer Press (1995)

Marquardt, W., "Trends in Computer-Aided Process Modeling" *Comput. Chem. Engng.* **20** (1996)

McKelvey, J.M., and G.V. Sharps, "Fluid Transport in Thin Film Polymer Processors" *Polymer Engineering and Science*, **19**, 9, 651-659 (1979)

McKenna, T., "Design model of a wiped film evaporator. Applications to the devolatilisation of polymer melts" *Chemical Engineering Science*, **50**, 3, 453-467 (1995)

Minsky, M., "Matter, mind and models" In: W.A. Kalenich (Ed.), *Proc. IFIP Congress Information Processing*, May 1965, New York City, **1**, 45-49, Spartan Books, Washington (1965)

Polymers Plus description, http://www.aspentech.com/ap/downloads/polymersplus.pdf (Accessed 12. April 2001)

Ramkrishna, D., "The status of population balances" *Reviews Chem. Engng.* **3** (1985)

Ray, W.H., "On the Mathematical Modeling of Polymerization Reactors" *J. Macromol. Sci.-Revs. Macromol. Chem.*, C8, 1, 1-56 (1972)

Reimschüssel, H.K., "Nylon 6 Chemistry and Mechanisms" *Journal of Polymer Science: Macromolecular Reviews*, **12**, 65-139 (1977)

Reimschüssel H.K., and G.J. Dege, "On the Condensation Equilibrium in Polymerization of Caprolactam" *Journal of Polymer Science: Part A-1*, **9**, 2343-2359 (1971)

Riggert, K., and F. Terrier, "Die Entfernung niedermolekularer Anteile in Polyamid-6-Schmelzen mittels Vakuum" *CZ-Chemie-Technik*, **2**, 3, 95-99 (1973)

Romagnoli, J.A. and M.C. Sanchez, *Data Processing and Reconciliation for Chemical Process Operations*, Academic Press (1999)

Saechtling, H., *Kunststoff-Taschenbuch*, Hanser, Munich (1995)

Tai, K., Y. Arai and T. Tagawa, "The Simulation of Hydrolytic Polymerization of ε-Caprolactam in Various Reactors" *Journal of Applied Polymer Science*, **27**, 731-736 (1982)

Wedel, L. v., and W. Marquardt, "Cheops: A Case Study in Component-Based Process Simulation" In: M.F. Malone, J.A. Trainham, B. Carnahan (Eds.), *Foundations of Computer-Aided Process Design*, AIChE Symp. Ser. 323, **96**, 494-497 (2000)

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 5.2: CAPE Tools for Off-line Simulation, Design and Analysis

I. D. L. Bogle & D. Cameron

5.2.1 OFF-LINE APPLICATIONS

This section discusses the off-line use of CAPE Tools. By this we mean that the process to which the tools are to be applied either does not yet exist, or it exists and on-line data are not readily available. It should be noted that almost all the tools to be discussed are applicable throughout the product and process lifecycle. This means that it is difficult to delimit a watertight discussion on off-line simulation, design and analysis. The tools to be described here tend to overflow into later sections of the book, since, for example, dynamic simulators are valuable tools for just the types of tasks that will be discussed here. However, since we have limited space, this chapter is limited to considering the following business processes (or activities)[1]:

- Research and development.
- Conceptual design, including process synthesis.
- Detailed process design.
- Off-line analysis of collected data, from experiments or operations.

These activities are hierarchically linked, as is shown in Figure 1. The R&D activity is a precursor to conceptual design, which in turn is a precursor to detailed design. Operations then follow from detailed design.

[1] To avoid ugly expressions and confusion, a "process" will always be chemical process in this section. A business process, on the other hand, is called an "activity".

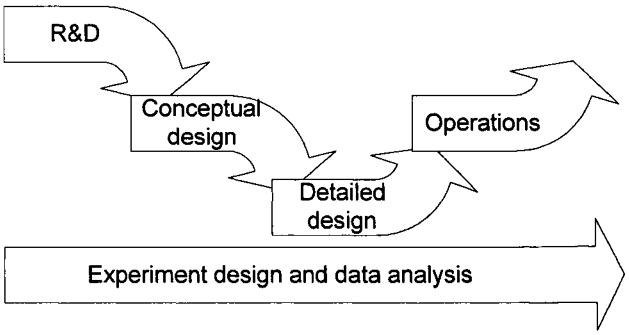


Figure 1. The off-line activities in the chemical process lifecycle.

Experimental design and data analysis is a supporting activity for the other four activities, in particular R&D and operations.

In this section, a variety of commercial tools and near-commercial academic systems will be presented. We will attempt to show how these tools support the off-line activities that lie behind the development and operation of a process. The overview will be necessarily a little superficial. Its aim is to give the reader an overview of the rich variety of commercial, semi-commercial and academic tools that are available.

The presentation is biased towards continuous production facilities. Specialised tools for batch and semi-batch processes are described in chapters 3.5 and 5.4. In addition, since tools for dynamic simulation are described in chapters 5.3 and 5.4, they are only briefly mentioned here.

5.2.2 OFF-LINE CAPE TOOLS - TOOLS FOR DECISION SUPPORT

5.2.2.1 Engineering Decisions

CAPE tools exist to provide information to engineers as they seek to find answers to questions of the following type:

1. What product shall I make?
2. How shall I make it?
3. What are the implications and side effects of making it this way?
4. Is the chosen method the best way of making the product?

The answers given to these questions, the engineering decisions made, determine the type of process facilities that are financed, built and run. Rational engineering decisions require:

1. Information about the physical behaviour of the planned or actual process. This information is provided by *experiments* and one or more *process models.*
2. Information about the economic constraints on the process.
3. A means of recording the decisions made.
4. A means of recording the rationale behind the decisions.

The methodology of process design has been widely discussed. Standard textbooks like Biegler *et al.* (1997), Seider, Seader and Lewin (2000), Turton *et al.* (1997) or Smith (1995) give a good overview of current teaching practice. The older book by Douglas (1988) is also still worth a look.

5.2.2.2 The variety of off-line CAPE tools

A wide variety of CAPE (and non-CAPE) tools are needed to support process R&D and design. Some of these are shown in Figure 2. The diagram shows a family of interrelated tools sharing calculation results and data to produce a plant design. Each of these tools will be presented in more detail later in this section.

A steady-state process simulator lies at the centre of Figure 2. A steady-state mass and energy balance is the primary tool for the design of both continuous and batch processes. They are used to calculate the design basis for the process. These are discussed further in section 5.2.3.1. An accurate steady-state model requires access to a system for calculating or estimating *physical properties* and predicting *thermodynamic equilibrium.* These are treated in section 5.2.4. Commercial steady-state simulators contain generic modules for reactors and separators. These are suitable for conceptual design and flow sheet generation, but lack predictive power. This means that they are of limited use in detailed design and operations. More detailed models of *specific unit operations* need to be built or used by the designer. These models incorporate the designer's proprietary knowledge. These models are often stand-alone Fortran programs. Some typical programs of this type are described in section 5.2.5.

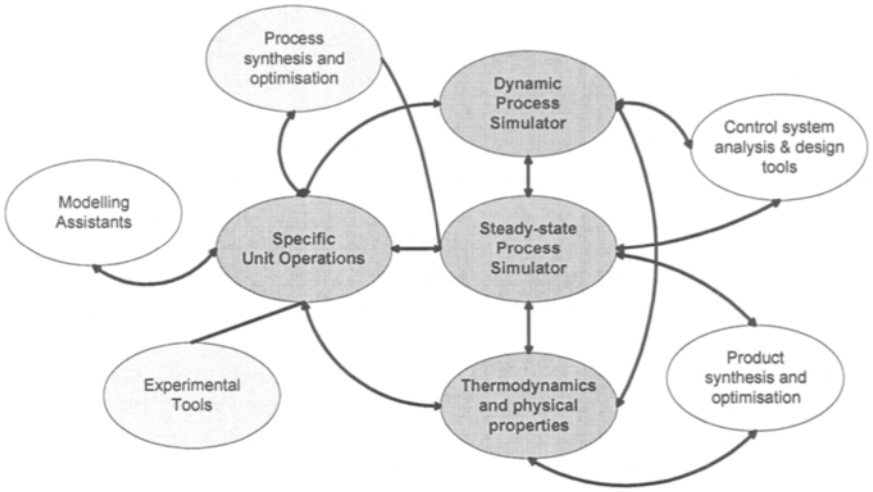

Process synthesis and optimisation
Dynamic Process Simulator
Control system analysis & design tools
Modelling Assistants
Specific Unit Operations
Steady-state Process Simulator
Product synthesis and optimisation
Experimental Tools
Thermodynamics and physical properties

Figure 2. CAPE tools in on-line applications, showing mutual relationships. Widely used and commercial tools are shown with dark shading and bold text. Less widespread, but commercially available tools are shown with lighter shading. Applications that are dominated by academic, prototype tools are shown in white.

These specific modules are often not easy to build and are based on experimental data (for example, from bench kinetic experiments, phase equilibrium tests and pilot plant runs). *Modelling assistants* are programs that attempt to automate the development of a process model. This speeds development and helps the designer in building a physically correct model. More details are given in section 5.2.5.5.

A steady-state simulator is a good tool for solving the mass and energy balance for a defined process flow sheet with a fixed structure. However, it is not, alone, capable of determining whether a given process structure yields the best possible energy use, water use, environmental impact, or operating profit. This calculation requires tools for *process synthesis and optimisation.* Process synthesis tools calculate the optimal process structure, whereas optimisation methods find the best operating conditions for a defined process structure. Both types of calculations are described in section 5.2.6.

5.2.3 STEADY-STATE GENERAL PROCESS SIMULATORS

5.2.3.1 Flowsheeting Simulators

Off-line applications were the first area where CAPE tools were applied. The first generation of flowsheeting programs sought to computerise the steady-state heat and mass balance and equipment design calculations that engineers did by hand. The method chosen was to reproduce the sequential approach of the hand calculation. Given a defined feed composition, temperature and pressure and a full set of equipment operating parameters, a procedure (or subroutine) corresponding to a unit operation is run to calculate a set of outlet streams. These outlets are then used as inlets to downstream units.

This sequential algorithm works well until a recycle occurs, or the user wishes to specify the value of a process output and manipulate an input or parameter to achieve this. Recycles are handled by *tearing*, i.e. guessing the values of one or more chosen process streams (the torn streams) and iterating through the flow sheet until these values are the same from one iteration to the next. Specified outputs are handled by *controllers*: the user inserts a block whose task is to adjust a selected input variable or parameter so that the output value achieves its specified value. Simulation controllers should not be confused with actual process controllers. They sometimes occur in the same places in the flowsheet, but a "controller arrangement" that converges a steady-state model will not necessarily work on the real, dynamic process.

General steady-state process simulators have a number of limitations. They are not efficient for solving for pressure in the system. The steady-state momentum balances that need to be solved here are so tightly coupled that an equation-oriented simulator is required (see section 5.2.3.2). Pressure is therefore specified in most steady-state design models. This means that the assumed pressure drops need to be verified at some later stage using a steady-state piping simulator or a dynamic simulator.

Furthermore, hold-ups and capacities are not represented in a steady-state model. If needed, hold-ups can be represented by fictitious streams in and out of the system. Indeed one author has suggested this as a way of performing dynamic simulations (Horwitz, 1996). Such ad hoc solutions are unwise. They are bad practice: impossible to document, difficult to understand and will only work with trivial problems and specific simulator set-ups (Newell and Cameron, 1996).

5.2.3.2 Solving Networks: Sequential-modular, Simultaneous-modular and Equation-oriented systems and their convergence.

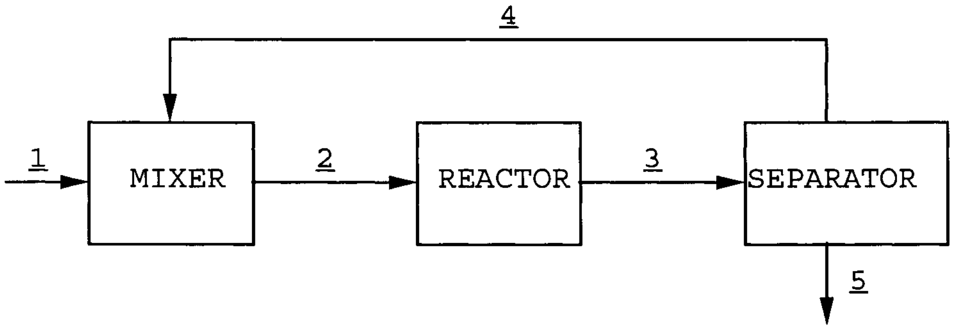


Figure 3. A simple process network.

Sequential modular systems calculate outputs of a unit from the inputs using calculation modules for each unit in the flowsheet. A sequential calculation strategy can be used to solve the whole flowsheet by solving each set of unit equations in turn, usually done as a separate module or routine for each unit. If a recycle exists the values of the variables in at least stream must be guessed. In the case of Fig 1 for example the outputs from the mixer can only be obtained if the stream vector for stream 4 is known. One method of solving this problem is to guess values for the stream vector for stream 4. Using these values one can calculate successively streams 2, 3, 5, and a new set of values for stream 4. The guessed values for stream 4 can then be modified and the flowsheet calculations redone until the calculated values for stream 4 and the guessed values are the same. When the two values are essentially the same the procedure is said to have *converged*. The guessed stream or streams are called the *torn streams*. The scheme uses an iterative procedure to solve the flowsheet equations.

If we denote all the torn variables, i.e. variables in a torn stream, by the vector **x**, a steady-state simulation can be converged by solving the non-linear equations:

$$x - f(x) = 0$$

where **f(x)** is the estimate of the torn variables at the next iteration. This is obtained by simulating through the model once.

If controllers are present there is an additional set of equations for the outputs, **y**, as a function of inputs and parameters, **z**:

$$y - g(z) = 0$$

The key to efficiently solving this model is to choose the best values of **x** and **z** at each iteration. Sequential simulators use one of several methods, including:

- Direct substitution, where $\mathbf{x}_{n+1} = f(\mathbf{x}_n)$.
- Wegstein's method, where:

$$x_{i,n+1} = x_{i,n} + \frac{f(x_{i,n}) - x_{i,n}}{1 - \dfrac{f(x_{i,n}) - f(x_{i,n-1})}{x_{i,n} - x_{i,n-1}}}$$

- Quasi-Newton techniques, where a secant approximation is made for the Jacobi matrix.
- Newton's method, where Jacobi matrix of f(x) is calculated and used.

Equation oriented flowsheeting systems set up the full system of algebraic equations and then solve them either simultaneously or by decomposing the equation set into separate blocks and solving each block independently. The information from a degrees of freedom analysis of a flowsheet, where the variables and equations are listed, can be set up in matrix form. This matrix is known as the *occurrence matrix* for the equation set. A well-posed problem can usually be obtained by reducing the columns of the matrix until the matrix is square. This is equivalent to specifying each of the variables for which a column is removed. When the matrix is square this represents a system in which the number of equations is equal to the number of variables and a solution can be sought.

The Jacobian matrix, J, of a system of equations, the matrix of derivatives, has the same *structure* as the occurrence matrix, i.e. where the occurrence matrix has a zero so too does the Jacobian and where the occurrence matrix has an entry the Jacobian has a value. If the Jacobian matrix is singular, i.e. its determinant is zero, for a particular set of parameter values then the equations are not all independent. Newton based methods are used to solve the equation set.

5.2.3.3 Commercial simulators for petroleum and chemicals processes

The market for steady-state simulation in the petroleum and chemical industries is dominated by four products: Aspen Plus from Aspen Technology Inc., Hysys (and its predecessor Hysim) from Hyprotech (a division of AEA Engineering Software), CHEMCAD, from Chemstations Inc., and PRO/II from Simulation Sciences. Aspen Plus and PRO/II are sequential modular simulators, whereas Hysys uses a simultaneous modular algorithm. Aspen Plus and PRO/II have a long history, beginning in the late 1970's and early 1980's as Fortran applications on mainframes and later minicomputers. Simulations were configured using keyword input files[2]. During the 1990's these tools migrated to personal computers, and the input files were hidden behind first DOS-based configuration tools (such as Aspen's Model Manager) and then Windows-based graphical configuration tools (newer versions of Model Manager, Simulation Sciences' ProVision, and in CHEMCAD). Hysim was developed later, and was written for personal computers. Hysys was a further development for a Windows operating system.

Table 1 Some market-leading steady-state simulators.

Name	Supplier	Web site
Aspen Plus	Aspen Technologies Inc.	http://www.aspentech.com/index.asp?menuchoice=ap5aspenplus
Hysys	Hyprotech (part of AEA Engineering Software)	http://www.hyprotech.com/process/default.asp
PRO/II	Simulation Sciences	http://www.simsci.com/products/proII.htm
CHEMCAD	Chemstations Inc.	http://www.chemstations.net

5.2.3.4 In-house Simulators

Many large chemical companies developed their own steady-state simulators during the 1970's and 1980's. For example ICI developed Flowpack, BP built Genesis, Norsk Hydro built MAHEBA and NEW*S. Exxon, DuPont, Dow, DSM, BASF, Bayer and Shell all had similar systems. During the late 1980's and 1990's most of these systems have been superseded by generic simulators. The production companies decided that software development and support was not core business and that it was not worthwhile maintaining the (expensive) skills that were needed.

There remain, however, some companies who maintain in-house simulators as a source of competitive advantage. Liquefied gas companies usually use their own simulators, with their own highly accurate thermodynamic and fractionation model, such as Linde's OPTISIM simulator (Volt, 1994). Effective design of air separation equipment requires very accurate calculation of phase equilibrium, as the relative volatility of oxygen and nitrogen is very low.

Haldor Topsøe is another company who maintains their own engineering simulator (Christiansen, 1992). Their maintenance of a specialised tool is

[2] These files were noticeably influenced by their Fortran pedigree. They were terse, dense and not very readable.

justified by their concentration upon a narrow range of processes in which proprietary reactor and catalyst technology is central.

5.2.3.5 Flowsheeting in Paper, Minerals and Metals Processing

The standard steady-state simulators are most widely used in the petroleum and chemicals industries. This has been due to both technical and economic pressures. Technically, the simulators were well suited to modelling fluids with relatively well-defined compositions and thermodynamic properties. They were less suitable for processes with poorly defined compositions and properties. This was despite the fact that Aspen was originally developed as a simulator for simulating processes with ill-defined compositions, such as coal (see, for example McIlvried, Marano and Boyd (1996)).

The petroleum and chemicals industries were also economically and culturally attractive to the vendors. Engineers understood the need for steady state simulation as a means of developing a design basis. Clear cost benefits could be demonstrated by using these new tools.

Minerals processing and metallurgy have provided niches for smaller vendors and research organisations to provide specialised steady-state simulators. For example, Mintek in South Africa, market Pyrosim, a simulator for pyro-metallurgical processes. Further description of this product is available at the web-site http://www.mintek.co.za/Pyromet/Pyrosim/Pyrosim.htm. Similar products are marketed by JKTech, the commercial arm of the Julius Kruttschitt Mineral Research Centre (http://www.jktech.com.au/software.htm). SpeedUp has been applied to electrolytic zinc processing (Barton and Perkins, 1988). A typical simulator for minerals processing is described by Houdouin and Coelho (1987). Simulators for minerals processing (*i.e.* crushing, grinding, screening and flotation) usually include data reconciliation algorithms as the predictive power of unit models are low and redundant process measurements and analyses are available.

Many pyrometallurgical and hydrometallurgical process are modelled using thermochemical software, such as Outokumpu's HSC software (http://www.outokumpu.com/hsc/) or FACTSage (http://www.factsage.com/). These tools offer limited modelling of flowsheets, but good modelling of single reactors using the minimisation of Gibbs free energy.

Pulp and paper processes are characterised by solid-liquid-gas systems, complex chemistries and poorly defined materials (such as pulp, broke and black liquor). The processes are also tightly coupled mass flow networks. This makes equation-based process simulators attractive. Again there are some niche products. Massbal (Shewchuk, 1987) is an equation-oriented simulator that has good market penetration in the paper industry. VTT in Finland markets a sequential-modular simulator called BALAS (http://www.vtt.fi/ene/balas/balas.htm).

5.2.3.6 From Steady-State Design to Dynamic Design

Dynamic simulation first grew out of the need to predict the effects of controller tuning, ensure reliable emergency procedures and to develop effective start up and shut down of plant. However a number of factors affecting plant design and operation have caused a much greater interest in plant dynamics recently. Plants are becoming more complex and more tightly integrated, both internally within individual plants and on a site-wide basis, via shared utilities, relief systems, *etc*.. Such integrated complexes often have more complex dynamic behaviour and are therefore more complex to commission and to operate. Upset dynamics is often a critical issue, potentially leading to safety & environmental problems and the propagation of faults through the site. Localised manufacturing is creating the need for much smaller plants with very similar requirements. Many of the newer high-value products (such as pharmaceuticals, fine chemicals, *etc*) are produced using batch (or other discontinuous) processes (and, of course, many plants are hybrids, *i.e.* contain both continuous and batch sections) - such processes are inherently dynamic and therefore more difficult to develop, design and operate.

The analysis and exploitation of the dynamics of a plant provides capabilities, which can help ensure that these goals are met. Great strides, particularly in the provision of dynamic simulation of continuous and to a lesser extent batch processes have helped in achieving some of the benefits. The offshore oil industry in particular has made great use of the tools and methods available.

A number of products are now available for use as design and verification tools. In many cases the tools are closely tied to steady state simulation tools. Aspen Dynamics from Aspentech is a dynamic simulation tool integrated with their steady state simulator. The system is an equation-based technology where the equations can be seen and modified. Hysys, from Hyprotech, is also tightly bound with the steady state capability allowing a very easy switch from steady state mode to dynamic mode. gPROMS from PSEnterprise Ltd (http://www.psenterprise.com) is a general purpose process modelling, simulation and optimisation environment allowing open access to the model equations. It has a more sophisticated technique for handling discrete events than the systems mentioned so far. They have also developed an object-oriented framework for the modelling and the dynamic simulation of chemical processes, Odysseo (Object-oriented Dynamic Simulation Software Environment), a reusable C++ development framework as a foundation for general or dedicated dynamic simulator design. Models are developed in equation-based form but it also supports inclusion of modules. Prosim (http://www.prosim.fr) has developed some tools for the design of batch processes which are by their nature dynamic.

5.2.3.7 From Steady-State Design to Optimisation

Most design problems are in fact optimisation problems – there are many possible solutions to the problem and the problem is to find the best choice. It is necessary that there be at least one degree of freedom and variables represented by these degrees of freedom are manipulated to find the best solution. It is important of course to define by what criterion the optimal choice is to be made i.e. to define an objective function. Investment design problems are made on the basis of a weighted balance of capital and discounted operating cost over the project lifetime such as the Net Present Value of the project. Many operational decisions are made on the basis of a measure of the current profitability of the short to medium term operating costs. For example refinery blending schedules are determined solving linear optimisation problems. Increasingly designs are being optimised on the basis of environmental performance using for example Life Cycle Analysis as a basis (Stefanis *et al.*, 1997; Azapagic and Clift, 1995) or the sustainable process index (Krotschek and Narodoslawsky, 1996).

Optimisation codes manipulate the variables until the objective function satisfies the optimality conditions, where the objective function cannot be improved locally, while still satisfying the model equations. The full optimality conditions are known as the Kuhn Tucker conditions. Most process optimisation tools use variants on the sequential quadratic programming algorithm (SQP). A full explanation of process optimisation can be found in Edgar *et al.* (2001). It should be noted that all optimisation tools currently implemented in design tools find a local minimum, which may not necessarily be the best, or global, minimum. Significant strides have been achieved recently in techniques for determining the global optimisation (see for example Floudas, 1999) but there is still some way before there are commercial codes. To attempt to obtain the global solution normal practice is to try the same problem from many different starting guesses. Aspen Plus, PRO II, Hysys and gPROMS all support an optimisation capability using SQP based algorithms. CHEMCAD does not, preferring to encourage the user to develop scenarios to explore the potential solutions.

5.2.3.8 Batch Scheduling

Tools for batch process design have been concentrated on solving the scheduling problem – deciding how best to schedule a set of recipes within a fixed set of plant resources. The tools assume that the times and capacities of each step in the recipe are known. This topic is covered in detail in chapters 3.5 and 5.4.

5.2.4 THERMODYNAMICS AND PHYSICAL PROPERTIES

Wide ranges of thermodynamic property prediction tools are available. Main simulator vendors have their own, proprietary thermodynamics system. This is also true for the most of the smaller vendors. For example, Prosim, offers a system called PhoPhy Plus. AspenTech's system, PropertiesPlus, is also available as a stand-alone product.

These calculation systems build on databases of pure-component properties and interaction factors such as

- o DIPPR (http://www.aiche.org/dippr/),
- o PPDS (http://www.ppds.co.uk/) and
- o DETHERM (http://www.dechema.de/infsys/dsd/englisch/dethermMain.htm).
- o

Figure 4 shows the typical structure for a physical properties system. The calculation program draws on one or more pure property databases, one or more mixture (interaction parameter) databases, a library of methods (equations of state, activity coefficient equations) and a library of algorithms (different types of flashes). All these components need to be in place and consistent with each other for a successful calculation.

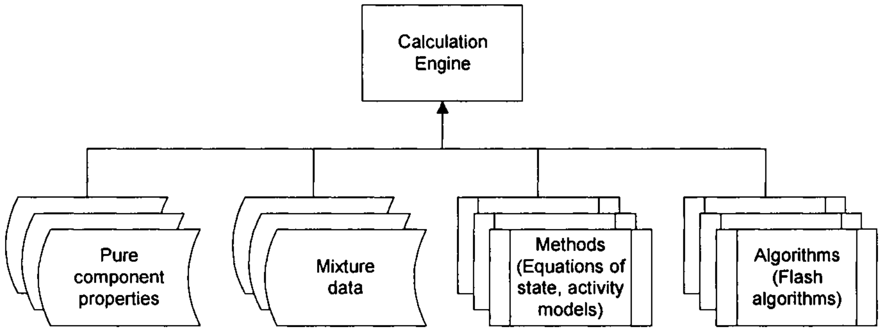

Calculation Engine
Pure component properties
Mixture data
Methods (Equations of state, activity models)
Algorithms (Flash algorithms)

Figure 4. Structure of a thermodynamic system.

A general simulator thermodynamic system must be able calculate properties for at least the following types of compounds:

- o oil fluids, including pseudo-components.
- o polar chemicals and solvents.
- o aqueous electrolytes.
- o polymers.

The large vendors' systems all have methods for representing these fluid types. Smaller vendors, such as Infochem (http://www.infochemuk.com/) and Calsep (http://www.calsep.com) have concentrated on delivering stand-alone products, with system integration features, for oil and gas systems.

The thermodynamic and physical properties packages are complicated and difficult to use. Typical problems encountered by users are:

- A large number of methods are available. These are of dubious relevance and unknown quality. As Linkson (1998) has pointed out, using an unsuitable method can give dangerously spurious answers. CAPEC at the Technical University of Denmark (http://www.capec.kt.dtu.dk) has developed, an expert assistant, called TMS. This program identifies the most appropriate thermodynamic model needed to describe the specified properties for a given mixture.
- Data for pure components or mixtures is not available, unless linked to suitable databases. All good physical property systems provide tools for regressing experimental data for pure components and mixtures. If, however, it is not feasible to conduct experiments, estimates of key parameters may be made using group contribution methods, such as UNIFAC (Fredenslund *et al.,* 1977). A graphical tool, called PROPRED uses a variety of group contribution methods to calculate pure component properties for molecules that the user interactively draws. This product was developed at CAPEC and an independent commercial version ha also been developed by IP-Sol (http://www.ip-sol.com/).

5.2.5 SPECIFIC SIMULATORS FOR UNIT OPERATIONS

As noted in the introduction, generic process simulators contain generic models for reactors and separators. These models are usually good enough for conceptual design, but are inadequate for detailed work. In addition, reactors and separators are usually the parts of the process that are proprietary and confidential. Most technology licensers possess a variety of detailed models for key processes. These are particularly useful for batch process design and development.

5.2.5.1 Reactors

Aspentech have a range of specialist reaction design systems, among others for Hydrocrackers, FCCs and Hydrotreaters. BatchReactor from Prosim is a tool for chemists, technicians and process engineers for prompt and safe design of batch chemical reactors. It permits the optimisation of the process combined with thermal stability analysis. Aspen's Batch Plus supports the development of modelling of batch recipes of reaction and separation stages.

5.2.5.2 Heat exchangers

During the mid 1990's the leading vendors of specialised software for designing and rating heat exchangers were acquired by one of the vendors of sequential modular simulators. Thus, HTFS' parent company acquired Hyprotech and integrated HTFS into Hysys. Similarly AspenTech acquired and integrated B-JAC. Around the same time, Simulation Sciences acquired the Hextran software.

These products are broadly comparable and are capable of designing and rating a wide variety of shell-and-tube, finned and compact heat exchangers. The software is available as a stand-alone system, but can be integrated with the parent company's simulator.

Smaller vendors also have niche products, for example, Prosim sells a product called ProSec, which is dedicated to brazed aluminum plate-fin heat exchangers and Ariane, which optimises power plants (steam, electricity, hot water) to determine the operating parameters that enable the lowest energy cost.

5.2.5.3 Distillation

Aspentech have a system for modelling non-equilbrium separations, both packed and trayed, called RATEFRAC, which uses a rate-based model to predict separations. For batch distillation Aspentech provide a tool called BATCHFRAC and Prosim provides Batchcolumn. Both of these models use rigorous tray-to-tray calculations based on phase equilibrium models.

Hyprotech markets a product for column sequencing and design of ternary and azeotropic distillation columns. This product, called Distil, is a commercialisation of research results from the University of Massachusetts.

5.2.5.4 Adsorption

Aspentech provide ADSIM for modelling and designing the full range of industrial gas and liquid adsorption processes. A brochure on this system can be downloaded from http://www.aspentech.com/ap/downloads/adsim_wp.pdf.

5.2.5.5 Modelling assistants

Developing models, especially those with any degree of detail, is a difficult task. In recent years there have been new tools to assist with this task. One of the first of these was MODEL.LA (http://modella.mit.edu/) (Bieszczad, 2000). Another approach to model building is presented in the ICAS system (Gani *et al.*, 1997) (http://www.capec.kt.dtu.dk). Modelling assistants are treated in more detail elsewhere in this book (see for example, chapters 3.1 and 5.1).

5.2.6 PROCESS SYNTHESIS

The problems solved by the tools outlined in section 1.2 all assume that the choice of units and interconnections between them, the structure or topology of the flowsheet, are predetermined. The aim of Process Synthesis tools is the automatic identification of the optimal choice of units and the connections between them for the production of a particular product. Algorithms for the solution of this problem have been appearing since the 1960s. There are three main approaches for this problem: properties based, optimisation based, and heuristics based.

5.2.6.1 Whole Process Synthesis

Jaksland *et al* (1995) proposed an approach to the problem, which determines which separation techniques are feasible using physical property differences between the major components to be separated. The approach is based on the idea that each unit operation is governed by a physical property driving force eg distillation is governed by the volatility difference between the components. Table 2 lists a number of alternative separation techniques along with the physical property, which governs its performance. For the method it is necessary to develop the binary ratio matrix, a matrix that contains the ratios between each pair of components of all the relevant physical properties. For the published results the properties are mostly predicted using group contribution methods.

Table 2. Separation Techniques and governing physical property

Separation technique	Property
Absorption	solubility parameter
Distillation	boiling point, heat of vaporisation
Crystallisation	melting point, heat of fusion
Microfiltration	size, molecular weight
Gas separation membranes	critical temperature, van der Waals volume
...	...

In optimisation based approaches an optimisation problem is formulated which encompass all possible solutions of the problem. It is necessary first to develop a mathematical description of the problem, which encompasses all solutions, often known as a superstructure, and to define an objective function, which characterises the performance of the proposed design. This may be economic or environmental or characterising any other objective which can be quantified. This optimisation problem is then solved using computational optimisation methods either using continuous optimisation methods such as sequential quadratic optimisation methods (Papoulias and Grossmann, 1983b) or using integer search methods (Fraga and McKinnon). This approach is often used for subproblems of the overall synthesis problem *e.g.* heat exchanger network problems (Papoulias and Grossmann, 1983a). Fraga (1998) has developed a system called Jacaranda, which uses integer optimisation solution method and coarsely discretises the continuous variable space.

Douglas (1988) developed a heuristic or rule based procedure for designing process plants. His procedure begins with defining the economic potential of the design as the difference between the product value and the raw material costs. The procedure then follows a series of decisions trees to decide the following questions:

- Operate in batch or continuous mode?
- What is the flowsheet input output structure?
- What is the recycle structure?
- What is the separation system?
- What process integration is possible?

The procedure involves continued evaluation of the effects of each choice on the economic potential. They have implemented the approach in a system called PIP. The use of thermodynamic heuristics in the design of heat exchanger systems has been a great success for the process industries and this is discussed in more detail in the following section.

GHN mbH market a synthesis tool called PROSYN, which uses an expert system principally using heuristics for process synthesis (http://www.ghn.de/). The system works through a number of domain expert programs for the following problems: the selection and design of reactors for a given reaction path (READPERT); the generation of distillation sequences using simple and complex columns (REKPERT); the development of energy integrated distillation sequences (HEATPERT); the development of distillation sequences for the separation of azeotropic or close-boiling mixtures (TEAGPERT); and the determination of the method of producing crystalline products and the selection and design of the appropriate crystalliser (KRISPERT).

5.2.6.2 Heat Exchanger and Water Network Synthesis

The area with the most concerted activity is the synthesis of heat exchanger systems motivated by the need for energy efficiency. Aspentech market a tool called Aspen Pinch. The tool implements the ideas of pinch technology for predicting minimum energy and capital requirements for a set of hot and cold streams with target temperatures.

Pressure on performance targets for the effective use of process utilities has encouraged the development of a number of tools for synthesising systems for the optimal use of process water. Aspen Water from Aspentech is a tool for optimising process water utilisation. Aspentech's Aspen Utilities focuses on the optimisation of the purchase, supply and usage of fuel, steam and power within environmental constraints.

AspenTech is used only as an example here. Similar products are available from other vendors and consortia, such as:

- Veritech (http://www.veritech-energy.com/).
- Linnhoff March (http://www.linnhoffmarch.co.uk/).
- UMIST (http://www.cpi.umist.ac.uk/).
- Hyprotech (http://www.hyprotech.com/hx-net/default.asp).
- NEL (http://www.ppds.co.uk/products/heatnet.asp).

5.2.7 DATA RECONCILIATION

Developing models off-line directly from plant data is an important application. While many off-line tools can be used with generic models where possible using realistic design parameters, many operational problems require better quality prediction from the models especially when used in conjunction with any on-line applications. This is done using available data, as much as can be obtained, together with a data reconciliation package which obtains the model parameters which best fit the data. See also chapter 3.4 for a detailed discussion on the data-reconciliation framework.

VALI from BELSIM (http://www.belsim.com) is an equation based data reconciliation software tool. It uses information from redundancy and conservation laws to correct measurements and convert them into models. VALI detects faulty sensors and pinpoints degradation of equipment performance (such as heat rate and compressor efficiency.).

Data Reconciliation is more efficiently done using an equation-oriented simulator than a sequential modular simulator. Thus, Simulation Sciences markets their data reconciliation product, DATACON[3] as a parallel product to the PRO/II simulator. Use of the Massbal simulator for data reconciliation has been described by Cameron, Morton and Paterson (1991).

[3] http://www.simsci.com/pdf/DATACON.pdf

5.2.8 PRODUCT SYNTHESIS

A new frontier for off line tools is in designing appropriate products for specific well-defined uses. There are few examples of generic tools in this are.

However one successful example is the Computer Aided Molecular Design (CAMD) technique, the computerised generation and identification of compounds with specific properties. ProCAMD (http://www.capec.kt.dtu.dk, http://www.ip-sol.com/) implements this using group contribution property prediction tools to build molecules with specific properties and has been used to identify/design potential new solvents, process fluids and refrigerants (Harper and Gani, 2000).

Westerberg and Subrahmanian (2000) have also developed some generic ideas but the topic remains in its infancy.

5.2.9 INTEGRATING THE DESIGN PROCESS

One of the weaknesses of design is that it is done in a sequential fashion choosing units, then interconnections, then capacities and flows and thermodynamic states, then control system to keep operations at the chosen conditions etc.. However these decisions are rarely distinct and much iteration is required to ensure that technical specification are met as well as safety and environmental limits and of course cost optimality or as close as can be obtained. The integration of the design process remains an elusive target, not helped by the diversity of tools used in the various stages of design. Even the first stage of integrating the design process, that of tools integration eludes us although some prototype tools such as Epee (Costello et al. 1996) and ICAS (Gani *et al.* 1997) exist. Also a prototype for tracking the rationale of design decisions, KBDS (Banares Alcantara (1995a, 1995b), has also been developed. True concurrent engineering in the design process is a challenging topic and progress in this topic is covered in detail in chapters 6.2 and 7.1.

5.2.10 REFERENCES

Azapagic A. and Clift R., (1995), "Life Cycle Assessment and Linear Programming - Environmental Optimisation of Product Stream", *Comput. Chem Eng.*, **19**(Suppl.), S229-S234

Banares Alcantara, R., (1995a), "Design support systems for process engineering. i. Requirements and proposed solutions for design process representation.", *Comput. Chem. Engng*, **19**(3), 267-277.

Banares Alcantara, R., and Lababidi, H.M.S., (1995b), "Design support systems for process engineering. ii. KBDS: and experimental prototype", *Comput. Chem. Engng*, **19**(3), 279-301.

Barton, G.W., and Perkins, J.D., (1988), "Experiences with SpeedUp in the Mineral Processing Industries", *Chem. Eng. Res. Des.*, **66**(5), 408-418.
Barton, P.I., (2000), "Modeling, Simulation and Sensitivity Analysis of Hybrid Systems", *Proceedings of the 2000 IEEE International Symposium on Computer-Aided Control System Design.*
Bieszczad, J, (2000), "A Framework for the Language and Logic of Computer-Aided Phenomena-Based Process Modeling", Ph.D. Thesis, MIT, available from http://modella.mit.edu/bieszczadthesis/index.html.
Biegler, L.T., Grossmann, I.E., Siirola, J.J., and Westerberg, A.W., (1997), "Systematic Methods of Chemical Process Design", Prentice Hall.
Cameron, D.B., Morton, W., and Paterson, W.R., (1991), "Framework for the modelling of metallurgical processes", *Trans. Instn Min. Metall. (Sect. C: Mineral Process. Extr. Metall.)*, **100**, C11-C20.
Christiansen, L.J., (1992), "A Computer-Aided Engineering System For Development Of Catalytic Processes", *Comput. Chem. Engng,***16**(Suppl.), May, S55-S68.
Costello D., Fraga E.S., Skilling N., Ballinger G.H., Banares-Alcantara R., J Krabbe J., Laing D.M., McKinnel R.C., Ponton J.W., & Spenceley M.W. (1996) épée: A Support Environment for Process Engineering Software, *Comput. Chem. Engng*, **20**(12), 1399-1412.
Douglas J.M., (1988), "Conceptual design of chemical processes", McGraw Hill, New York.
Edgar T.F. and Himmelblau D.M. and Lasdon L.S., (2001), "Optimization of Chemical Processes", 2nd Ed., McGraw Hill, New York.
Floudas, C., (1999), "Recent advances in global optimisation for process synthesis design and control: Enclosure of all solutions", *Comput. Chem. Engng*, **23**(Suppl.) S963-S973
Fraga E. S. and McKinnon K.I.M., (1994), "CHiPS: A process synthesis package", *Chem Eng Res. Des.*, **72**, 389-394.
Fraga E.S., (1998), "The generation and use of partial solutions in process synthesis", *Trans IchemE*, **76A**, 45-54.
Fredenslund, A., Gmehling J., Rasmussen, P., (1977), "Vapour-Liquid Equilibria Using UNIFAC", Elsevier, Amsterdam, The Netherlands.
Gani R., Hytoft G., Jaksland C., and Jensen A.K., (1997), "An integrated computer aided system for integrated design of chemical processes", *Comput Chem Engng*, **21**(10), 1135-1146.
Harper, P.M., and Gani, R., (2000), "A multi-step and multi-level approach for computer aided molecular design", *Comput. Chem Engng*, **24**(2-7), 677 – 683.
Horwitz, B.A., (1996), "Avoid nausea when solving dynamic problems", Chem. Eng. Prog., March, 44-51.
Hodouin, D. and Coelho, S.V., (1987), "Mass Balance Calculations Around Mineral Processing Units Using Composition Analyses Within Particle-Size Classes", *Int. Jnl. Mineral Processing*, **21**, 65-82
Jaksland C., Gani R., and Lien K. (1995) 'Separation process design and synthesis based on thermodynamic insights' Chem Eng Sci. 50/3 511-530

Krotschek C. and Narodoslawsky M. (1996) The sustainable process index. A new dimension in ecological evaluation. Ecological Engineering, 6(4) 241.
Laing D.M. and Fraga E.S (1997) A case study on synthesis in preliminary design. Comput chem Engng **21** Suppl. PpS53-S58
Linkson, P.B., (1998), "Can you trust your aqueous system simulations?", *Chem. Eng. Prog.*, November, 39-47.
McIlvried, H.G., Marano, J.J., and Boyd, J.H., (1996), "Development of an ASPEN model of a direct coal liquefaction facility", http://www.fetc.doe.gov/publications/proceedings/96/96jpfs/jpfs_pdf/aspen.pdf
Newell R.B and Cameron I.T., (1996), "Issues for future dynamic process simulators", *Chemical Engineering in Australia,* ChE21(2), 11-14.
Papoulias, S., and Grossmann, I., (1983a), 'A structural optimisation approach to process synthesis - II Heat recovery networks', *Comput Chem. Eng.*, **7**, 707
Papoulias, S., and Grossmann, I., (1983b), 'A structural optimisation approach to process synthesis - III Total Processing Systems', *Comput Chem. Eng.*, **7**, 723
Seider, W.D., Seader, J.D., and Lewin W.R., (1998), "Process Design Principles : Synthesis, Analysis, and Evaluation", Wiley, New York.
Shewchuk, C.F., (1987), "Massbal Mk II: New Process Simulation System", *Pulp Paper Canada,* **88**(5), T161-T167.
Smith, R., (1995), "Chemical Process Design", McGraw-Hill, New York.
Stefanis S.K., Livingstone A.G. and Pistikopoulos E.N. (1997) Environmental impact considerations in the optimal design and scheducling of batch processes. Comput Chem Engng. 21/10 1073-1094.
Tolsma, J.E., Clabaugh, J., and Barton, P.I., (1999), "ABACUSS II: Advanced Modelling Environment and Embedded Simulator", http://yoric.mit.edu/abacuss2/abacuss2.html
Turton, R., Bailie, R.C., Whiting, W.C., and Shaeiwitz, J., (1997), "Analysis, Synthesis, and Design of Chemical Processes", Prentice Hall.
Volt, J., (1994), "The production of computer-based plant optimization systems with the OPTISIM® process simulator. Experience with a Linde air separation plant", *Linde Reports on Science and Technology,* **54**, 19-25.
Westerberg, A.W. and Subrahmanian, E., (2000), "Product Design", *Comput. Chem. Engng,* **24**(2-7), 959 – 966.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 5.3: Dynamic Simulators for Operator Training

D. Cameron, C. Clausen & W. Morton

5.3.1 INTRODUCTION

This chapter describes current practice for the use of dynamic simulators in the process industries. Over the last two decades, dynamic simulation has matured as a tool for training operators of capital-intensive and safety-critical equipment (such as oil platforms, power stations and nuclear reactors). Advances in computer speed, programming methods and user interface facilities have made dynamic simulation a robust, effective and relatively inexpensive way of training operators. As this has occurred, dynamic simulators for training have been adopted (often in a simplified form) in other industries.

Furthermore, the technical quality of simulation models has improved with computer speed. This means that dynamic simulators can also be gainfully used for engineering design and operations support. These trends will be described and explored later in the chapter.

The discussion begins with a discussion of dynamic process modelling. This sets the context for later, more detailed discussions. This is followed by a brief historical overview of dynamic simulation in the process industries and related areas, such as energy production. This is followed by a brief discussion of how and where dynamic simulators can be used in the process lifecycle. This section provides a road-map for the rest of the chapter.

Dynamic simulators (together with on-line systems) are perhaps the area of Computer Aided Process Engineering where user requirements are most stringent. A presentation of the requirements of a specific, but typical industrial operation is given in section 5.3.6.

Once this background has been given, it remains to describe the tools that are currently available. This is done in two steps. Section 5.3.7 describes the two methods that are used to solve dynamic models of processes. Section 5.3.8 lists and describes typical products that can be used for dynamic simulation.

Sections 5.3.9 and 5.3.10 revisit the discussion of lifecycle dynamic modelling in more detail. Section 5.3.9 presents how dynamic simulators are used in operator training and section 5.3.10 gives an example of how a dynamic model can also be used to provide on-line operator support.

The chapter concludes with a few thoughts about how dynamic simulators will develop over the next decade.

5.3.2 WHAT IS A DYNAMIC PROCESS MODEL?

5.3.2.1 Models are sets of differential and algebraic equations

The theory of process dynamics is part of standard undergraduate Chemical Engineering theory, and is well covered by standard process control textbooks such as Luyben (1990) or Bequette (1998). The dynamic behaviour of a process is governed by the laws of conservation of mass, momentum and energy. These, for an example of a lumped control volume (see Figure 1) with in and outflow and *N* chemical species present, give rise to a system of *N+2* differential equations:

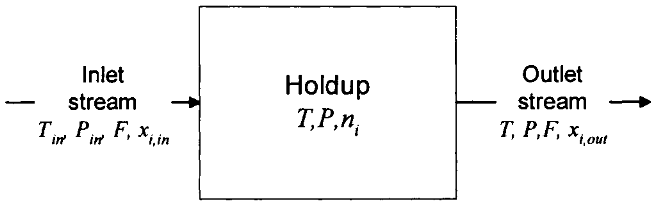


Figure 1. A simple, lumped control volume

Conservation of mass for each chemical species

$$\frac{dn_i}{dt} = F_{in}x_{i,in} - F_{out}x_{i,out} + \varepsilon_i \text{ for } i = 1, N$$

Conservation of energy (enthalpy in a flow system)

$$\frac{dH}{dt} = F_{in}h_{in} - F_{out}h_{out} + \Delta H + q$$

Conservation of momentum and equation of state (with constant volume)

$$\frac{dP}{dt} = f\left(V, \frac{dT}{dt}, \frac{d\sum n_i}{dt}\right)$$

These differential equations are accompanied by a set of algebraic equations. These algebraic equations represent (a) definitions of variables in the model, (b) calculations of physical properties, (c) quasi-steady-state relationships for phenomena with fast dynamics and (d) process inputs.

Thus, any dynamic model is a system of differential and algebraic equations (DAEs). A dynamic model of a chemical process would typically include hundreds of differential equations and thousands of algebraic equations. Efficient solution of these models is not trivial. Methods for solving such systems are reviewed in section 5.3.7

.

5.3.2.2 Dynamic models include more than just the bare process

The dynamic behaviour of a chemical process depends on much more than the dynamics of the actual processing units. To be useful a process model needs to take account of the dynamic behaviour of the "ironmongery" in the process system. Thus, a dynamic process model will need to include dynamic models of:

- sensors and transmitters (*e.g.* thermocouples and flow sensors).
- analogue-digital conversion and sampling.
- signal processing and transmission.
- logic and sequences.
- control algorithms.
- actuator dynamics (*e.g.* valve positioners, actuators and variable-speed drives).
- final control elements (*e.g.* valve stem travel, hysteresis and sticking).

These additional models introduce further DAEs, and will usually introduce discontinuous, discrete events.

This has two implications:

(1) Dynamic process models require much more information than the corresponding steady-state model. Steady-state models are largely indifferent to equipment size and shape[1], controller tuning, sampling time, signal noise, valve hysteresis or shut-down sequences. An adequate dynamic model is, however, dependent on the correct modelling of these features. Indeed, the dynamic behaviour of many processes is dominated by the dynamics of the control system.

(2) Dynamic process models are mathematically challenging. The modeller always needs balance physical fidelity and efficient solution.

[1] Although a predictive steady-state model of a reactor will need to take account of the reactor size and flow patterns.

5.3.3 REVIEW OF APPLICATIONS

This review will, given the backgrounds of the authors, give most attention to the oil and gas industries. However, the use of dynamic simulators in this industry is in many ways typical for other industries. As a rule, training simulators have been widely adopted in industries where capital investment is high, processes are complex and the consequences of plant or operator failure are serious. Industries of this type are the offshore oil and gas industry and the power and energy industry (both nuclear and conventional). Adoption has been slower, but still significant, in oil refining, chemicals and pulp and paper. The presentation of these sectors will be brief, and will point out areas where practice differs from the oil and gas industry.

5.3.3.1 Oil and Gas Production

High fidelity dynamic process simulators have been used extensively by all major companies within the oil and gas industry for more than two decades. Basically within process design studies, detailed engineering studies, process de-bottlenecking, control system verification, but also within operator training. Constantly changing process conditions, slugging problems from pipelines, changing feed stocks and complicated control systems are the reality in the day to day operation of most plants. Only well trained operators, thoroughly tested operational procedures and optimised control schemes will guarantee maximum throughput and safe operation.

This comprehensive use of dynamic simulators is based on the fact that steady state simulations do not give any answers about transient behaviour of the process. They provide an instant picture of a particular design case, and do not include feedback from the process itself or from the control system. A steady state simulator also neglect the effect of equipment hold-ups and integrating elements in the process. A dynamic model solves the necessary equations of the process system and calculates all plant model variables as functions of time. This means that the dynamic response of the process to changing operating conditions can be thoroughly examined and evaluated with a dynamic simulator.

Normally top side dynamic simulators can be split into two main types: engineering simulators and training simulators.

Engineering simulators

The main objective of the engineering simulators was, and still is, to undertake detailed operational and design studies. An engineering simulator is developed to be an accurate and detailed model of the main processes of a plant (or plant area) using either generic or emulated process controllers.

A generic controller is a standard PID algorithm, as supplied by the simulator vendor, whereas an emulated controller is the real control system functionality recreated within the simulator supplier's software.

Process engineers would normally focus their attention to a relatively small area of the plant and perform a series of tests and studies enabling them to suggest solutions to a given problem. This need for detailed engineering studies demanded a thorough treatment of physical behaviour, which in turn demanded fast computer processing. The lack of processing capacity a decade ago resulted in slow simulations, often below one fifth of real time. This was a nuisance, but was not considered a great problem.

Most significant fields or sub sea wellhead/pipelines in the North Sea have been modelled either as a separate engineering model or added onto an existing engineering model. Multiphase pipelines have normally been modelled using OLGA, whilst topside processes have been modelled using one of the different modelling tools listed in Section 5.3.8.3.

So far dynamic simulators has proven to be a powerful and valuable analytical tool in investigating the time varying behaviour of a process system.

Training simulators

The main objective of a training simulator is to educate skilled and competent control room operators. To be able to meet this objective the training simulator should represent the process plant as close as possible. The dynamic response of the training simulator should not differ significantly from the behaviour of the real plant. Thus the simulator needed to run in real time. Normally a training simulator would contain more models (although somewhat simplified) of the process utility areas than an engineering model. As an example, a training simulator of an offshore oil and gas platform would include models of the emergency and process shutdown system, fuel gas system, flare system, cooling and heating systems, regeneration systems, and water treatment systems. In addition necessary sequences to perform start-up of turbines, compressors and pumps would be modelled. This modelling approach was chosen to ensure that the operators received adequate training in operating all main areas of the process. Training simulators was normally based on either an emulated or a stimulated control system. These are explained further in section 5.3.8 of this chapter. Stimulated simulators gave the possibility of using and testing the real control system prior to upgrades, but the capital hardware costs were high. Emulation was cheaper, but any change in process or control system configuration needed a change in emulated software by the simulator supplier. This was a time consuming process and the choice of concept was basically a question of philosophy.

This first generation of training simulators was considered very successful. Using Statoil as an example, training simulators have been built for the Gullfaks, Sleipner, Heidrun, Norne, Troll and Veslefrikk offshore fields. In addition, simulators have been developed for the Kårstø NLG plant, Kollsnes Gas Treatment plant and the Tjeldbergodden methanol plant. A smaller batch training simulator also exists at the Mongstad refinery. These operator training simulators together cover the whole range from computer-based trainers to full replica operator simulators. Most of them are still in daily use.

5.3.3.2 Nuclear Industry, Power and Energy

Operator training and dynamic simulation was recognised as a key enabling technology for safe operations of nuclear plants (Nuclear Energy Agency, 1998). In many jurisdictions, simulator-based operator training is required by law. In some countries (including Finland, the USA and Canada), trainees are examined using the simulators for licensing purposes.

In a survey of operating companies and training institutions in OECD countries (Nuclear Energy Agency, 1998) it was found that 100% of operators in responding companies (in Belgium, Canada, Finland, France, Germany, Japan, South Korea, Spain, Sweden, Switzerland and the USA) trained using a process simulator that represented the control room and all key process operations. 80% of operators trained on a simulator that was a replica of the plant on which they worked.

Specialised simulators of nuclear reactors are regularly benchmarked and compared. There are ranges of simulators used, and it appears that most national research organisations and power utilities maintain a core simulator and a power plant simulator. A benchmarking of different simulators on simulating a break in the main steam line of a Pressurised Water Reactor is presented in Nuclear Energy Agency (2000).

A representative system is APROS, which is produced and marketed by VTT, the Technical Research Centre of Finland (Puska, 1999). This simulator is capable of simulating both the reactor core and the power system, whereas an older system uses different software for these components, and an interface was required to the model of the core.

Historically, developments in simulation in the Nuclear industry have catalysed and enabled applications in other industries. For example, the Norwegian nuclear research institute, IFE, was involved in the pioneering training simulators in the North Sea (Endrestøl *et al,* 1989) and remains active both in its own right, and as a collaborator with Kongsberg Simrad (Ek, 2000). Other vendors, such as GSE Systems and VTT have taken the same path.

Simulator systems that were developed for nuclear applications can naturally be extended to model coal-fired or gas-fired thermal power plants.

5.3.3.3 Chemicals Industry and Refining

Training simulators in the chemical industry and oil refining use the same technology as all other training simulators. The main qualitative difference is that chemical reactors are key elements in the process, and hence in the process model. The reactor is a challenge to the process model and the integrator. The reactor is often the most complicated unit to model: dynamic models of chemical reactors usually require the solution of a system of partial differential equations. Each reactor has a unique geometry, flow arrangement and reaction scheme – this means that reactor models often need to be tailor-made for a given process. Furthermore, reactors are usually proprietary, licensed technologies. Technology licensers can be reluctant to disclose detailed information about process chemistry and reactor design to the supplier of a dynamic process simulator. In this situation the modeller has three choices:

1. A simplified representation of the reactor can be built, that represents how the vendor believes that the model works.
2. A "black box" model of the reactor, obtained from the technology licensor, can be integrated into the simulation model. This may require substantial work in interfacing the model and ensuring that it interacts properly (*i.e.* in a numerically stable way) with the rest of the simulation.

5.3.4 LIFECYCLE DYNAMIC SIMULATION

5.3.4.1 Evolution of the training simulator

In the recent past, and still to some extent, the engineering simulator and the training simulators for a platform were purchased separately, sometimes even from two different suppliers. Accurate and stable engineering studies demanded very detailed modelling and a high execution rate of the calculations and sequential network solvers (*i.e.* short time steps). Engineering simulators therefore ran slowly. Operator training simulators, on the other hand, must be able to run in real time, rather than being 100% accurate under all transient situations. Real-time execution was not possible with complicated thermodynamic calculations and fast execution rate using the prevailing processing power and speed. A training simulator therefore often compromise on modelling rigour and accepted longer time steps to achieve real time simulation. Filters and lags then compensated for instability and ensured plausible and stable behaviour in transient mode.

The development of modern computer technology has opened several new possibilities. Steadily increasing processing speed, multiprocessing computers and the use of standard application software and platforms makes it possible to develop simulators that are faster, better and more detailed than ever before. Now, even large simulators will be able to run in real time even with small time steps. New technology also makes it possible to build simulators through a graphical online builder, to generate a dynamic model from a steady state model, to import sequences or cause-and-effect diagrams, to use vendor soft controllers and to keep track of modelling data. Today fully integrated steady state/dynamic simulators are commercially available. We are very close now to the life cycle simulator concept where the same simulator model is used throughout the lifetime of the plant. The vision is one basic starting model, where the simulator's functionality increases in scope and fidelity with time and use.

Around the early to mid 1980's simulators began to be applied in the chemical industry by vendors with backgrounds in the nuclear and energy industries. Thus Womack (1986) can give a review of dynamic simulation in Mobil. At that stage Mobil had used ACSL for model building and also purchased simulators from Bechtel and Lummus/CE Simcon (now ABB). Training simulation started in the early 1970s, with an analog computing simulator used for start-up training of oil refineries at Joliet and Wilhelmshaven.

Around the same time, ICI was using ACSL to develop a training model of an evaporator plant (Aldren, 1986). This solution, using a standard solver for modelling, was contrasted with vendor-supplied training simulators.

During the late 1980's and early 1990's implementation in chemical processes passed from being pioneering work into common practice. The introduction of reasonably-priced minicomputers, notably the VAX, reduced the cost and technical risk of training simulators. Most of the simulators that are commercially important today began to be widely used in this period. Thus, SACDA's "Trainer" simulator was used to develop training models of pulp and paper plants. It was later acquired by Honeywell, and formed the basis for their training simulator products.

Jones and Brook (1990) describe a typical training simulator delivery using Simcon's GPURS software. The simulator was installed on an ethylene plant run by Shell. The control system, a Honeywell TDC3000, was emulated (see section 5.3.9). This highlights a further change that occurred in the late 1980's. Second-generation control systems (such as the TDC3000) began to appear. These were more complicated than the panel or first-generation systems that they replaced. This increased the need for training.

5.3.4.2 Applications through the process lifecycle

As we have seen above, dynamic simulation is routinely used in the design and start-up phase of processes, as shown in Figure 2. In addition, one simulator is increasingly used for both engineering and training. It is therefore attractive to consider using this simulator to support and document *all* phases of the process lifecycle.

The simulator can be used as a vehicle for transmitting and retaining knowledge from conceptual design, through design and commissioning to operations. It can also be used as a means of transferring experience (and data) from operations back into the design of future plants and the retro-fitting of existing plants.

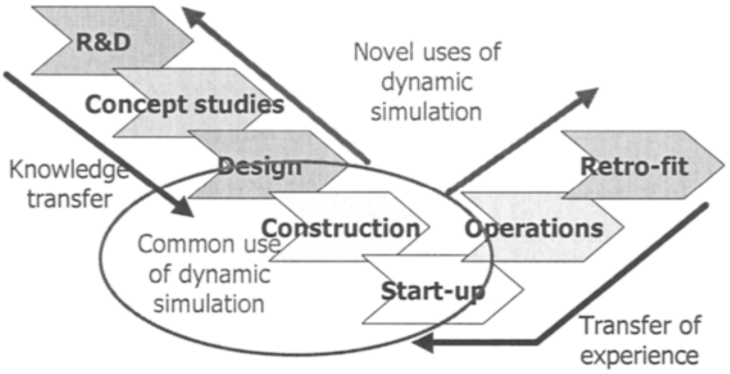


Figure 2. Dynamic simulation in the process lifecycle.

This is what we call the Life Cycle Simulator concept.

A life-cycle simulator model in the oil industry should be used for:

- Conceptual studies in the pre-design phase.
- Engineering (what-if) studies during detailed design (de-bottlenecking, process verification).
- Control system verification studies.
- Testing of new applications.
- Perform start-up preparations.
- Validation of operational procedures.
- Optimisation and troubleshooting.
- Training and evaluation of operators.
- Operator support and monitoring systems.
- Safety training and crisis handling.
- Modification and tail production[2] studies.

[2] By tail production we mean the period in an oil field's history where oil production is declining. The drop in oil production rate results in a relative increase in water and gas production. Undesirable multiphase flow phenomena, such

5.3.5 INDUSTRIAL REQUIREMENTS

A process simulator should be robust and reliable, accurate and repeatable, easy to maintain, use and update, be based on high fidelity models with stable network solvers and first principle techniques. In addition a modern process simulator should be based on standard Windows technology and be fully compatible with standard windows office applications (such as spreadsheets).

The model shall be stable, realistic and repeatable when operated within the operating domain of the process plant: from ambient start-up conditions through steady state operation, shutdown situations and operation through malfunctions.

A dynamic process model should be based on the following requirements:

- Standard PC computers. Developed and running on Windows software. Hardware should have significant spare computer power, spare storage capacity and extension capabilities (printers, CD-ROMs, internal/external net).
- Object oriented, online configurable graphic interface (WYSIWYG-philosophy).
- High fidelity mathematical models based on first-principles engineering practise and sound mathematical methods.
- Three-phase flash capabilities (water-oil-gas) and multiphase flow capabilities.
- Rigorous thermodynamics or continuously updated local property correlations. All calculations should be based on equation of state algorithms. The results should be presented in engineering units (This means SI in Europe, and otherwise in the USA).
- Well proven and stable simultaneous pressure/flow network solvers within fast transients.
- Acceptable integration methods to ensure fast and stable calculations.
- Robust, fast and reliable simulation through discontinuities (vessel overflow/dry-up, phase changes, compressor surge and stonewall regions. etc.)
- Conversion tools to and from steady state/dynamic state.
- Proven DCS interface technology. DCS to be based on suppliers soft controllers.
- Easy on-line parameterisation/validation of process equipment and control modules.
- The graphical user interface should, like a CAD program, support "layers". Thus, the engineer building a model can place all control schemes on one layer, all logic on another, and instructor facilities on yet another layer. It is then simple to display or hide these features as required.

as slugging, may become more common. Pressure in the processing system may also need to be reduced. All these effects require careful optimisation of operating conditions and re-training of the process operators.

- Easy graphical update functionality for ESD, PSD, sequences and logic.
- Flexible naming convention when copying equipment or plant areas.
- Animation or HMI symbols to show the status of process equipment (*e.g.* running/stand-by/off-line, levels and valve position).
- Standard instructor and engineering facilities to ensure optimal use. (run/freeze, malfunctions, operator evaluation, printable trending, data export/import)

The requirements of engineering and training simulators may still differ somewhat due to the irreducible demand of real time execution of an operator training simulator. This might cause some simplification in the models and in accuracy in transient areas, but with multiple processors this should be considered rather a challenge than an obstacle out of reach.

The need for instructor facilities and operator monitoring systems on a training simulator are also a considerable difference between the two types of simulators. Again, today's technology should, by now, be able to overcome these challenges by adding/removing/loading specific application software and ultimately bring together the best from both simulator concepts.

5.3.5.1 Industrial needs

Traditionally process design was based on steady state simulation tools such as PRO II. Operability and control issues were addressed after the process had been designed, and normally tested on a separately developed dynamic simulator. Today this could be the same simulator tool.

What the oil and gas industry needs is a high fidelity process simulator that contains the same core model for both engineering and training purposes. A simulator model that can evolve naturally from an early steady state model in the concept design phase, be extended into a dynamic simulator with detailed modelling for engineering studies and finally be moved into a rigorous dynamic model with soft controllers to be used for detailed control/modification studies and operator training. This is the life-cycle simulator described in section 5.3.8.

In addition to the process model it is necessary to have:

- Well-skilled engineers and instructors to enable correct and optimal use of the process simulator.
- A training room where the training sessions can be performed in familiar surroundings.
- Well-documented training courses based on operational procedures.
- A variety of training scenarios for normal and emergency operation.
- Objective monitoring system for operator evaluation.

Finally, but not least importantly, a management decision that the process simulator should be considered as a constant investment rather than an annual expense.

5.3.5.2 Challenges to be met

Process simulators are considered by the majority of oil and gas industry to be a very valuable tool in both engineering studies and in operator training. But they are also considered to be a tool for specialist. The key of future success is to ensure that the simulation tool is moved towards a more general accepted and user friendly tool. This means that dynamic simulation tools must be constantly developed and improved to withstand the increasing demands to model fidelity and usability.

Co-operation between simulator suppliers and company engineers in performing/ordering the studies and further develop the models.

Today's process simulators are based on either generic or emulated controllers. Future process simulators must be based on supplier's soft code of the real controllers. This will enable the process simulator to perform engineering studies on the real control system without having a complete traditionally stimulated system.

The possibility to run the process simulators on the internet/intranet will enable process engineers to be able to solve process problems both onshore and offshore.

The possibility to extend existing models to include the physical behaviour of various internals in drums and other equipment. This will enable detailed studying of limited process areas where more detailed modelling is necessary.

An integration of the dynamic process simulation system and transient multiphase pipeline models must be handled within the same network solver. This will make it possible to study the dynamics of slugging interaction between pipeline and topside equipment.

Future simulators must be based on the same methods for fluid modelling and thermodynamic flash calculations for both engineering and training simulators.

Companies must participate in developing Best Practice methods for standard simulation studies to ensure consistent approach.

Another future development might be to connect the topside model to a reservoir simulation model to fully integrate the dynamic behaviour of the reservoir and flowing wells to pipeline and top side models.

5.3.5.3 Requirements

The simulator model shall be based on the piping and instrumentation diagrams (PIDs), system control diagrams (SCDs), cause and effect diagrams and isometric drawings of the process. Marked up plant piping and instrumentation diagrams shall clearly identify, by suitable colour coding, the equipment, pipes or instruments which shall be included in the process model. All equipment modelling shall be based on suppliers data sheet unless otherwise agreed by Company.

The minimum level of fidelity required for specific types of equipment, instrumentation and unit operations is defined below:

Vessels (*e.g.* separators, scrubbers)

The modelling of these items of equipment must be based on internal geometry and separation devices. The modelling of vessels shall be based on rigorous heat and mass balances with resulting vapour / liquid equilibrium of incoming streams. Heat loss to surroundings shall be correctly modelled for both phases.

Heat Exchangers

Exchanger performance shall be calculated based on upon the flowing stream's temperature, flow rates and composition, the exchanger geometrical arrangement and the thermal properties of the materials of construction. The model shall account for heat loss to the surroundings with appropriate effects of heat exchanger materials of construction and fluid hold up incorporated.

Compressors (Centrifugal and axial)

Compressors shall be simulated based on vendor supplied nozzle to nozzle flow versus head and polytropic efficiency curves with variable compressor speed (normally min. 4). Effects due to variable gas properties (P, T, MW, H) shall be included. Effects of surge and stonewall are to be represented in suitable detail. Run down times upon compressor trip must be based upon a power balance between load and supply. Fuel gas consumption shall be correctly calculated from turbine speed and efficiency.

Pumps

Pumps shall be based on vendor supplied performance curves. Where variation in flowing medium conditions *e.g.* temperature, composition have significant effects on observable behaviour they shall be simulated. Both forward/reverse flow through a stopped pump shall be correctly modelled.

Drivers (rotating equipment)

For large or variable speed machines the rotating speed shall be calculated based on a power balance between drivers and loads, including appropriate friction losses and total of inertia of all equipment to the drive shaft. For smaller speed machines simplified start up and shutdown dynamic responses shall be permitted.

Control valves

Control valves shall be based on a standard or vendor-supplied flow correlation with the correct valve characteristic. In vapour service the effects of choked flow are to be simulated. Where available, valve stem stroke times are to be used. In the event of loss of instrument power (air/hydraulic) all valves are to travel to their fail safe position.

Isolation valves

Isolation valves shall be simulated with stroke times from equipment data sheets if available. Limit switches, where applicable, are to be realistically positioned on valve stems. The restriction to flow shall be variable as a function of valve opening and representative of the valve type.

Pressure relief valves

Pressure relief valves shall be simulated to reflect the opening/closing nature of these items of equipment. Multiple relief valves with identical set pressures can be modelled as a single device with equivalent capacity.

Piping network

The flow distribution within piping networks must be accurately represented with adequate regard taken for equipment configuration and operational pressures. Dynamic responses within the network due to equipment operation must be realistically represented.

Plant logic

The emergency and process shut down system shall preferably be based on easy-to-modify cause and effect diagrams to enable quick and easy change of logic.

Simulator Model Accuracy

The steady state accuracy of the model when compared to operating data from the actual process shall be within +/- 2%. The requirements for accuracy of transient behaviour must be better than +/- 10% for transient greater than 1 minute.

5.3.6 ALGORITHMS

As mentioned above, a dynamic process model is described by a system of differential and algebraic equations (DAEs). Process simulators solve these equations in one of two ways:

- *sequentially*. Algorithms of this type are described in section 5.3.6.1.
- *all-at once*. Algorithms of this type are described in section 5.3.6.2.

5.3.6.1 Sequential Simulators

Block-oriented simulation

A sequential process simulator divides the process into small blocks. A block usually corresponds to a piece of equipment (*e.g.* a vessel, pipe, pump, motor or valve). These blocks are connected together by data structures that represent streams and signals. Each block is a procedure that, given values for inputs (at the last and new time step), calculates values for the state variables and output variables at the next time step. Blocks are usually written in a third-generation programming language (Fortran, C, or, increasingly, C++), and each block is responsible for implementing its own integration algorithm and equation solving method. This means that the equations solved in a block are "hidden" from the simulator executive program and the user.

In other words at a certain time, a block contains values for the inputs at time step *j*, the state variables at time step *j* and the outputs at time step *j*. The block immediately upstream executes and passes the inputs at time *j+1* to the block. The block procedure uses all this information to calculate the state variables and outputs at time step *j+1*. The output values are then passed to the downstream blocks.

This algorithm works well as long as there are no recycles of material or information in the system. However, the method can break down if such recycles occur. Luckily, most process and information recycles normally involve time delays longer than the control sampling interval, so that a "once-through" plant models (with the recycle broken) may be adequate.

Network solvers and flow pressure networks

However, the sequential modular algorithm has difficulty dealing with upstream information flow, which notably occurs in "flow-pressure networks". Consider the simple system shown in Figure 3, which consists of a network of pipes and pumps.

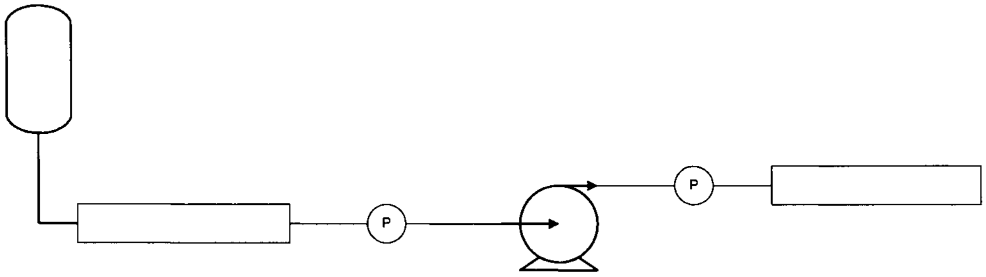


Figure 3. A simple flow-pressure network. Pipe segments are shown as rectangles and pressure nodes are shown as circles.

The pressure in the vessel depends on the flow rate in the pipe outlet. However this requires knowledge of the pressure at the outlet of the pipeline. This is available at the previous time step, so a calculation can be made – but it can easily be inaccurate and unstable. Flow-pressure interactions are fast, especially in liquid systems, which mean that very small integration steps are needed to avoid this instability.

Successful commercial dynamic simulators solve this problem by setting up and solving a system of simultaneous equations for flow and pressure in the flow pressure network. Blocks in a flow pressure network are of one of two types:

- *Flow blocks*, such as pipe segments, pumps and valves, calculate flow given inlet and outlet pressure.
- *Pressure blocks*, such as vessels and branches, calculate pressure given all inlets and outlet flows.

The system of equations for flow and pressure is structured and sparse. Thus, commercial simulators use some variant of Newton-Raphson equation solving and a sparse linear algebra solver. Algorithmic finesses like line search and relaxation are used to increase accuracy and robustness. A well-designed and well-implemented network solver can be very efficient. Network solver-based simulators have been able to accurately solve process models with hundreds of pressure nodes in real-time.

However, where there is close coupling between flows, pressures and other process variables (such as temperatures and compositions), unphysical interactions may arise between the variables in the network and these other variables. In some situations this requires careful implementation of the model if spurious results are to be avoided.

Sequential dynamic simulators are widely used, and dominate the market for training simulators. Commercial products of this type are presented in section 5.3.8.3.

5.3.6.2 Differential-Algebraic Simulators

It may be better to use an efficient, standard DAE solver (on a fast machine) despite the computational cost this involves. DAE solvers work by reducing the system of DAEs to a system of algebraic equations through the application of a time discretisation. This can be demonstrated simply through the use of *velocity variables* (Smith, 1985, Smith and Morton, 1988).

If the DAE system is written in the following form:

$$\mathbf{f}\left(\frac{d\mathbf{x}}{dt}, \mathbf{x}, \mathbf{c}\right) = 0$$
$$\mathbf{g}(\mathbf{x}, \mathbf{c}) = 0$$

where **x** is the vector of dynamic variables and **c** is the vector of algebraic variables, we can convert this system into a system of algebraic equations by introducing the velocity variables, **s**, which are the current values of **dx**/d*t*. If this is done the following non-linear equation set can be solved for a full time step, *d*, from time t_k to t_{k+1}:

$$\mathbf{f}(\mathbf{s}, \mathbf{x}, \mathbf{c}) = 0$$
$$\mathbf{g}(\mathbf{x}, \mathbf{c}) = 0$$
$$\Delta(\mathbf{s}, \mathbf{x}, d, t)$$
$$t - d = t_k$$
$$d = d_{specified}$$

where Δ is a vector if functions obtained from the chosen integration algorithm. Higher order methods, such as Runge-Kutta solve this system of equations at intermediate points in the time step.

Widely used DAE solvers are capable of solving *stiff* systems of equations effectively through error control and variation of step size.

A stiff system of differential equations is dominated by the time response of one or two dynamic variables with short time constants (e.g. mass transfer on a distillation column tray). The fast dynamics of these variables requires short time steps for stability and accuracy. However, once these fast transients have died out, the same short time step would be inefficient and inaccurate (due to the accumulation of round-off errors) for solving the rest of the dynamic response.

Thus, the Massbal dynamic simulator uses an adaptive Runge-Kutta algorithm, first published by Prokopakis and Seider (1981). Other simulators, such as SpeedUp, use an implicit Gear-type (BDF) algorithm. One advantage of using a Runge-Kutta type algorithm is that step length is an explicit part of the algebraic equation system. It is therefore relatively straight-forward to handle discontinuities and timed events. This is more difficult with Gear-based algorithms. Recent simulators, such as gProms, offer a choice of algorithms.

DAE simulators have been widely used for the detailed, accurate simulation of smaller plant units. They have used computationally-intensive algorithms and, until recently, lacked a graphical user interface. However, these simulators are maturing and may allow a common engineering and training model of the plant to be maintained if the latter is detailed enough. Whether this is yet feasible in the majority of applications is open to debate. There may, for example, be a need to buy a supercomputer or a network of workstations (*e.g.* one processor for each unit or group of units) to achieve real-time dynamic simulation. In this case the handling of communications between processors needs to be addressed.

The products that use this type of algorithm are presented in section 5.3.8.1.

5.3.7 SOME TYPICAL PRODUCTS

5.3.7.1 General Differential and Differential-Algebraic Simulators

Much work on dynamic simulation of processes and control systems, especially by students, is done using general equation-solvers and software. One of the oldest of these is ACSL (Breitenecker and Lingl, 1998). This program can uses a graphical configuration tool to build models in a way that resembles analogue computing (with gain, integrator and summer blocks) (Gauthier, 1999). ACSL was widely used in the chemical industry in the 1980's (Aldren, 1986, Womack 1986). SimuSolv was a proprietary tool, developed by the Dow Chemical Company as a system for simulating chemical systems, especially reactors. It is still used academia as a teaching tool. Both of the above tools were valuable in their time has since been superseded by the advent of equation-oriented process simulators. They now are useful in legacy applications, teaching and for low-budget, one-off modelling of processes.

Probably the most prevalent general tools in use today, particularly in academia, is Matlab, with its graphical modelling tool, Simulink. This program provides a mathematical programming language that makes it a good tool for rapidly constructing and solving systems of equations. Over the years, the base system has been supplemented by commercial and academic toolboxes covering areas such as signal processing, optimisation and control. Particularly within the process control community, Matlab has become a means of publishing and sharing methods and algorithms.

A valuable guide and comparison of general tools is given by the Arbeitsgemeinschaft Simulation News at their web site (http://www.argesim.org/).

Finally, a number of simulators developed in the milieu around the University of Lund, Sweden, have proven to be powerful and effective simulators for mechanical and energy systems. Object-oriented prototype systems, such as Omola and Dymola (Cellier, 1991) have provided a basis for a consensus modelling language – which can be supported by any number of systems (such as Dymola) – called Modelica (Modelica, 2000). These tools are more used for simulating mechanical systems than process systems, but both Omola and Dymola have been applied to chemical and energy systems. An application of Dymola to an Akzo Nobel ethoxylation plant is described by Askaner (1999).

5.3.7.2 Differential-Algebraic (Equation-oriented) Process Simulators

The primary disadvantage of the above tools for large-scale dynamic modelling is that they lack the *infrastructure* that is needed to support process simulation. As general tools, they lack a natural unit-operations orientation (with the exception of the object-oriented systems, such as Dymola), they lack data structures that can represent process streams or material hold-ups, and they lack well-structured access to standard methods of calculating thermophysical properties and equilbria. Specialised process DAE simulators have these features, and thereby provide greater productivity.

This group of process simulators is widely used for engineering simulations, but are less widely used for training simulators. All these simulators express the process model explicitly as a system of DAEs, which are then solved using one of the algorithms described in Section 5.3.6.2.

Pioneer simulators of this type were SpeedUp from Imperial College London, Quasilin from Cambridge University (Smith and Morton, 1988) and Ascend from Carnegie Mellon University (Perry and Allan, 1996). SpeedUp was commercialised by Aspen Technologies and is now called Aspen Custom Modeller. The academic experience obtained in building SpeedUp was then embodied in the gProms simulator (Barton and Pantelides, 1994), which was designed to support discrete events and distributed-parameter models (*i.e.* partial differential equations). In a similar way the experience of Quasilin was implemented in the Massbal simulator (Shewchuk and Morton, 1990), which is now marketed by Hyprotech.

Both SpeedUp and gProms allow the user to build models by declaratively writing the variables and equations that represent a unit operation. A properly defined unit operation could then be used (repeatedly, if needed) in building a model of a flowsheet. Ascend had a similar approach. Models of common unit operations and processes could be built using a library of ready-made unit operations. Massbal, on the other hand, provides a library of finished unit operations in which the equations are hidden from the user. In addition, it is possible to build additional unit operations declaratively, but this feature is less elegant than in SpeedUp or gProms. On the whole, a program like Massbal is easier to use than SpeedUp, but is less flexible. My understanding of Aspen Custom Modeller is that it has taken this approach to increasing usability.

The greatest challenge in solving a large equation-oriented dynamic model is giving a consistent set of specifications. An effective equation analyser and degrees-of-freedom counter are essential if this is to be done effectively.

5.3.7.3 Engineering and Training Simulators

The below listing of process simulator suppliers is by no means complete, but reflects the knowledge of the authors involved. It is recommended to use the Internet or other sources to supplement the list. The listing gives the name of the supplier followed by location and the name of the simulation tool. All suppliers have varied experience within both engineering and training simulators, and also within various processes *i.e.* Distillation towers, LNG-plants, absorption towers, amine/glycol regeneration, CO_2- injection, and compact heat exchangers. Again a comprehensive listing of supplier's deliverables and model references will give a valuable insight into possible choices. Especially within specialised processes i.e. where catalytic processes are involved and often are considered proprietary information.

Supplier's name	**Location**	**Name of tool**	**Reference / Website**
Kongsberg Simrad	Norway	ASSETT	http://www.kongsberg-simrad.com
Fantoft Process AS	Norway/UK/US	D-SPICE	http://www.fantoft.com
Honeywell	US/Canada/UK	Shadow Plant / Trainer	http://www.iac.honeywell.com
Hyprotech (AEA)	Canada/UK/US	Hysys/Plant	http://www.hyprotech.com
RSI (IFP)	France	INDISS	http://www.rsi-france.com
Aspentech Ltd;	US/UK	OTISS	http://www.aspentech.com

Supplier's name	Location	Name of tool	Reference / Website
ABB Simcon	US/UK	*ABB SIMCONx*	http://www.abb.com
GSE (S3/Singer)	US	SimSuite Pro	http://www.gses.com/
CAE	Canada	ROSE	http://www.cae.com/
Esscor (Invensys)	US	Ascend	http://www.esscor.com

5.3.8 TRAINING SIMULATORS

5.3.8.1 What is a training simulator?

Operator training is necessary for both new and experienced operators. New operators need training in the layout of the control system, normal operations, start-up and shutdown and general plant behaviour. More experienced operators need refresher courses and advanced training in how to deal with unexpected problems, hazards and incidents.

In addition, engineers and process managers require training in how the process works and how operating decisions will affect plant performance.

Training simulators allow training to occur in a structured, well-ordered way, without disturbing normal plant operations. Operators can use the simulator to tackle situations that only occur rarely in practice. This experience can then be used to avoid loss of production time and equipment if these situations do occur.

5.3.8.2 Types of training simulators

Overview

Training simulators may be classified both by application type and technology. We identify three applications for training simulators:

(1) Operator training.
(2) Process training.
(3) Discipline training.

There are three system technologies that are used for training system development:

(1) Stimulated systems.

(2) Emulated systems.
(3) Quasi-stimulated systems.

All these terms are defined below.

Operator training

The most common, and most comprehensive, type of training simulator is used for operator training. Process operators are trained and drilled in operating their process using a simulated copy of the plant's control system. These systems are relatively expensive, since they require that the operator trainee interact with an HMI that that is the same as (or at least resembles strongly) the actual plant control system.

Process training

Operators, engineers and maintenance staff require a good knowledge about the plant on which they are working. The fundamentals of this knowledge are obtained from process training courses. A simulation of the plant can be useful tool in these courses. The instructor can use the model as a "living flow sheet" of the process during the course. The model can be run in plenum to demonstrate key procedures and features of the process. Furthermore, engineers and maintenance staff can be given a better understanding of the operator's tasks through the demonstration of start-up and shut-down procedures. Finally, trainees can also use the simulator itself to gain familiarity with the process and its peccadilloes.

Process simulators are usually less expensive than full-featured operator trainers. No connection with a control system is needed, and, often the simulator's native user interface is sufficient for use in training.

Discipline training

Finally, dynamic models of generic processes or process segments can be valuable aids in general process engineering education. High fidelity process models of realistic processes can be useful tools for gaining hands-on experience and insight in process control. For example, a lecture or course on controller tuning can conclude with a practical session where a trainee can tune a variety of control loops. This hands-on experience is invaluable in building the necessary feel for the dynamics of processes and their control systems.

5.3.8.3 Types of training system

Overview

Operators communicate with their process through a control system (either a (Distributed Control System) DCS, or a SCADA (Supervisory Control and Data Acquisition) system (where the process controllers are implemented in a number of PLCs). They need training in how they relate to the process through the DCS or SCADA human-machine interface. This means that a training simulator must provide the same "look and feel" as the control system HMI.

This similarity can be obtained in three ways:

- by using the simulator to communicate with a physical copy of the entire control system. This is called a *stimulated* system.
- by using the simulator to communicate with a physical copy of either the DCS operator stations or the SCADA part of a SCADA/PLC control system. This is called a *quasi-stimulated* system.
- by building a software replica of the control system and its user interface and communicating with this. This is called an *emulated system.*

Stimulated Systems

In a stimulated system the simulator implements process behaviour only, and communicates with a physical copy of the control system. The process simulator provides the control system with values for its digital and analogue inputs (equipment status, measurements). The control system, in turn, provides the simulator with values for digital and analogue outputs (trip signals, controller outputs). Conceptually, the simulator is responsible for representing system behaviour up to the control system's I/O cards. Indeed, in older training systems, the simulator communicated with the control system through the actual I/O connections. This is less common now, as control systems now offer one or more application programming interfaces (APIs) that can be used for software communication with the relevant I/O tags in the control system. Indeed, the OPC standard makes it possible to use a generic interface to several types of control systems.

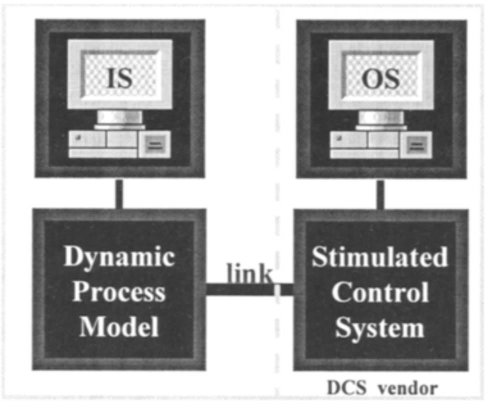


Figure 4. Stimulated Training System

Emulated Systems

An extra copy of the control system hardware can be expensive, in addition proprietary, unpublished communications protocols may make stimulation of a control system infeasible. In such circumstances, it is attractive to simulate the control system and its user interfaces alongside the process. It is then the responsibility of the simulator vendor to copy the appearance and behaviour of the control system's user interface. This is labour consuming, even with older systems, which had static, full-screen displays. This task becomes even more complicated with windows-based user interfaces, since the layout of screens can be variable. The emulated operator station runs on cheap, non-proprietary, hardware, such as a PC.

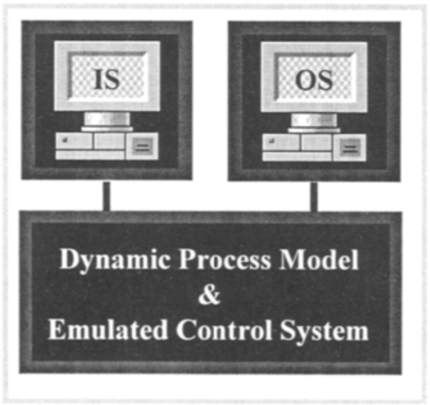


Figure 5. Emulated Training System

Quasi-stimulated Systems

If a given control system user interface is difficult to emulate, it may be possible to use a model of the process and its control system to provide data and receive values from an actual copy of the control system's user interface hardware (*i.e.* the operator stations).

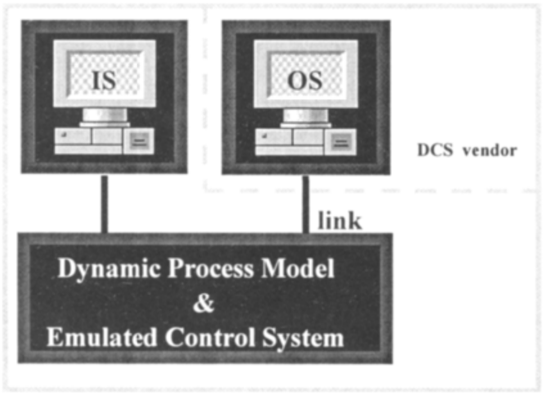


Figure 6. Quasi-stimulated Training System

A disadvantage of this arrangement is that the communication link has high bandwidth (data transfer rate) compared with a stimulated system. However, this bandwidth can be minimised by exception reporting.

5.3.8.4 Control system check-out and maintenance

A stimulated training simulator is also a useful tool for checking, verifying and pre-tuning the control system. Since the simulator provides the control system with a valid set of input signals and gives realistic responses to outlet signals, it can be used to find errors and problems in the control system before it is commissioned. This saves labour and speeds up production start.

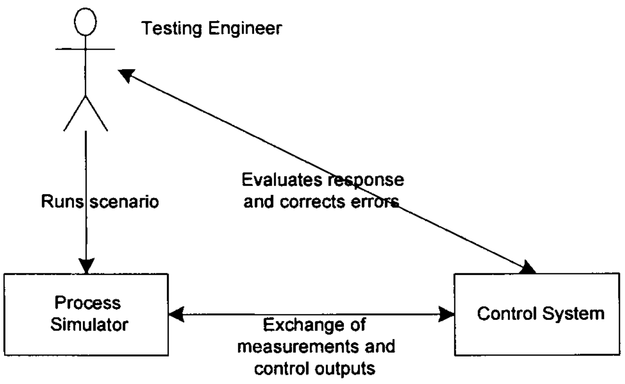

Testing Engineer
Runs scenario
Evaluates response and corrects errors
Process Simulator
Control System
Exchange of measurements and control outputs

Figure 7. Control system checkout

A check out simulator consists of four components (as shown in the figure below):

- The actual *control system* hardware and software (prior to shipping) or a duplicate thereof (if this is to be used later for a training simulator).
- The *process simulator*, which simulates the behaviour of the process system. This model responds to changes in analogue and digital outputs received from the control system.
- A *link between the simulator and the control system*. This is software for transferring data between the simulator and the control system. The data simulator sends values for all relevant analogue and digital inputs. The control system sends values for all relevant analogue and digital outputs in return.
- A *human-machine interface*. The testing engineer will use this interface to:
    - Configure and run scenarios.
    - Examine the response of the control system and its effects on the process.
    - Configure, monitor and run the link software.

The responses of the PLC can be logged and examined using this interface. All aspects of the behaviour of the model in response to the actions of the PLC can also be inspected using the HMI.

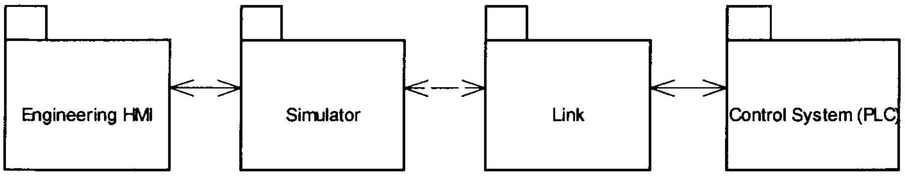


Figure 8. Components in the check-out simulator

5.3.8.5 Training Practice

Well-trained and confident operators are the best guarantee for safe and optimal operation throughout the lifetime of the plant. This sounds like an obvious fact, and it is a fact, but it is not obvious how we can achieve this goal during our training practice. A training simulator model, as good as it might be, will never be successful if used as a stand-alone model. Simulator training must be seen in a wider context. We need to combine skilled instructors, well prepared training sessions, with a good simulator model containing the necessary instructor features, to achieve our training goals.

The operator training therefore must be based on four pillars.

- a high fidelity simulator model. (See section 5.3.7.3).
- detailed training courses.
- simulator instructor facilities according to requirements.
- good and skilled instructors.

All operator training should be based on this fact and training sessions and the training simulator should be developed according to these needs.

Simulator training courses

Many major oil companies have developed their training courses after the following pattern:

Simulator Course I Duration: 1 week.	Introduction to the control system, ESD/PSD and Fire and Gas system. Familiarisation with the man machine interface (MMI). Process familiarisation. Simple MMI operations.
Simulator Course II Duration: 1 week.	Normal process start-up of wells and separation system. Normal process start-up of compressor system with gas export and/or gas injection. Process shut down.
Simulator Course III Duration: 1 week.	Emergency training. Handling of critical situations from process changeover, local shutdown to complete shutdown and depressurisation of the plant.
Simulator Course IV Duration: 2-3 days.	Annual refresher courses with predefined scenarios. Training in normal/abnormal situations. Emergency situations.

Some oil companies still continue the practice of holding a one day refresher/update after each four week leave period.

All training courses should be carried out in the same manner and be based on predefined scenarios to ensure objective and consistent evaluation of the operators.

Instructor Station

The training sessions are controlled from the Instructor Station. The instructor Station is the graphic interface from which the instructor prepares the training sessions, supervises the participants and monitors operator actions for later debriefing. Normally the Instructor Station would be situated in a separate room. The instructor interface shall be realised using the latest display technology to achieve the best overall performance. It shall be easy to configure and modify for different applications.

As a minimum, the following instructor functions shall be available:

- Run/Freeze
- Speed (altering the speed of execution).
- Load/Save Initial Conditions (IC).
- Snapshots, Backtrack and Replay
- Change of Operating Conditions
- Field Operator Functions (FOD), *e.g.* opening or closing a manual valve at a specified rate.
- Malfunctions
- Scenarios (event or time controlled).
- Trainees Performance Monitoring System (operator evaluation system).
- Monitoring, trending and Reporting Facilities.
- Online help facilities.
- Comprehensive self explaining error log.

Instructor skills

The instructor is responsible for preparing the initial process conditions for training, running predefined scenarios, creating equipment malfunctions and controlling the operator monitoring system. The instructor plans and carries out the training sessions in compliance with educational techniques.

The instructor also acts as the field operator and implements any field actions that are required by procedures or the operators.

To be able to take care of all the instructor roles the instructor must have a comprehensive knowledge of the process and its operational procedures. The instructor to be able to act correctly as field operator and plant personnel during training sessions.

The instructor must know the Instructor Station functionality's and have the pedagogic knowledge to be able to run the training sessions professionally.

The instructor must be aware of standard teaching techniques, and prepare all training sessions accordingly. Debriefing should be held after all emergency/decision training courses.

The instructor must be positive and supportive when correcting operator errors to ensure the training aim: *better-skilled and more confident operators.*

The Training Centre

Several major oil companies (and refineries) have built their own training centres and included large-scale replica training simulators of the process control rooms. Especially from the early 1980's many large-scale training simulators were built. The largest training simulators, such as Gulfaks, Veslefrikk, Sleipner (Statoil), Oseberg/Brage (Norsk Hydro), Draugen (Shell) and Snorre (Saga) are all examples of what were basically full replica training simulators. Several other training simulators (*e.g.* by ESSO, BP, Philips and Elf) have been built but not to the same extent. Full replica training simulators included correct size of rooms, colours, floors, panels and operator stations as well as a separate instructor room with instructor stations. Even radios, plant monitors and telephones were correctly installed in many cases. The main reason for these large-scale full replica training simulators was to create the feeling that this was the real control room and the real plant. The approach is similar approach to that employed within the nuclear power industry.

Today's training simulators are no longer 100% replicas of the control room. Many panels, monitors and communication are simplified, but the training simulators are still expected to serve the overall need for training. Still a training simulator is the only tool we have.

All the same, it is still considered important that the training room should have an outlook fairly similar to the control room to avoid the feeling of "running a computer game". At least the Training Centre should consist of a control room area with operator stations (and panels), an instructor room with instructor station and a classroom area for theoretical tutoring and debriefing.

Training Tasks

The ultimate goal of any training is to give the participants the necessary skill, knowledge and confidence to be able to handle any known process operation and be able to analyse and correct any unforeseen event. Dynamic real-time simulations are excellent training tools because the can reveal complex interactions between process components and subsystems.

Critical situations can be demonstrated without the risk of damage to life and equipment. Emergency situations and operational procedures can be visualised and rehearsed throughout training sessions.

Shift personnel can be trained together in emergency situations. (Co-operation effects)

Operators and instrument engineers can use the simulator to optimise control schemes.

Control room operators can be taught to take correct action to process disturbances.

The operators will learn what to do, and equally important, what not to do in critical situations.

Basic training tasks:

- Familiarisation with the control system
- Training on start-up and shutdown procedures.
- Operator training (normal start, stop operation of wells, oil-train, gas train, utility systems)
- Team training (shift training within normal operation)
- Decision training (what to do in abnormal situations, malfunction of equipment)
- Procedure training (change over, well testing, compressor load change etc.)
- Emergency training. (critical situations,)
- Plant start-up (black start up of plant,
- Control system operation (change of parameters, set points of controllers)

Simulator advantages

Within large technically complex installations human errors are still the most frequent cause of shut down and emergencies. The use of training simulators is by far the most efficient tool to develop skilled and confident operators.

Skilled and well-trained operators will lead to:

- Better understanding of process interactions
- Fewer shutdowns.
- Quicker start-up.
- Improved safety.
- Optimised throughput.

5.3.9 REAL-TIME DYNAMIC SIMULATORS

5.3.9.1 Overview

As noted above, engineering models, simulator-based operator training and control system checkout are now routinely-used tools in the oil industry. Oil companies invest large amounts of intellectual (and real!) capital in models of their processes.

Real-time simulation applications allow this "corporate knowledge" to be made available to operators as they run a process. A dynamic model (or partial model) of the process is set up to run in parallel to the process and to access and use measured data from the plant's control system. The model then uses this information to:

- Ensure that it is tracking (or shadowing) current process conditions. A discrepancy between the model's predictions and process conditions may indicate a fault. The operator should be informed about this.
- Calculate estimates of unmeasured (indeed, in some cases, non-measurable) variables that are of use to the operators, plant engineers and plant management.

To provide the above functions, an on-line model requires the following components:

- a database for storing process data.
- an interface for obtaining process data from the control system.
- a process model which is capable of following process behaviour and performing useful calculation.
- a module for checking process measurements, reconciling measured data, estimating unmeasured variables and tuning the process model.
- an interface for reporting results to the operator.
- a real-time executive system.

This structure is shown is shown in Figure 9.

A model that is tracking the process can provide a validated starting point (a snapshot) for a range of other calculations:

- A *predictive*, or look-ahead model that runs certain fixed scenarios ahead of the process and flags potential problems.
- A *planning* model, where the engineer is free to specify a scenario, or set of scenarios, to be evaluated.

- A *hindsight* model, that can be used to re-run plant history so that mistakes that were made can be avoided in future.
- Various control and optimisation calculations.

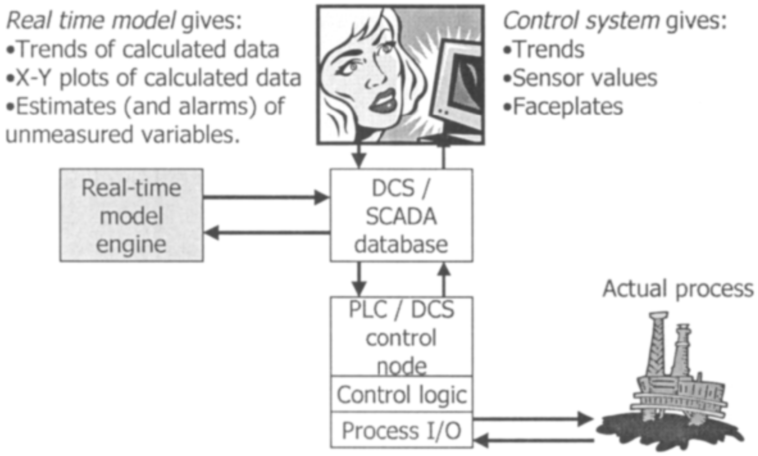


Figure 9. Structure of a real-time system.

Real-time systems have begun to be widely implemented in downstream operations, such as refining and ethylene production. Here the typical application consists of a model-based process controller for key units (such as catalytic crackers) or a steady-state on-line model, which has the role of calculating optimal set points for the process control system. This last type of application is called on-line optimisation.

Real-time systems in oil and gas production are more seldom. Most modern compressor anti-surge systems are actually small real-time systems. A model of the compressor performance is used to place the surge and stonewall lines and determine appropriate control actions.

5.3.9.2 Systems requirements

The process model runs on a *Real-time Model Server* computer that is connected to the same network as the control system (SCADA) server. The model(s) on the model server will exchange values directly with the SCADA server's database.

The simulation system in the model server reads real measured process values from the database in the SCADA server and uses them for model tuning. The model is tuned slowly, using selected parameters, so that it closely follows normal plant operation. Estimates of unmeasured variables can then be passed back to SCADA server for trending, alarm handling and display.

Model maintenance and tuning will be done through an *Engineering Station* with a graphical HMI.

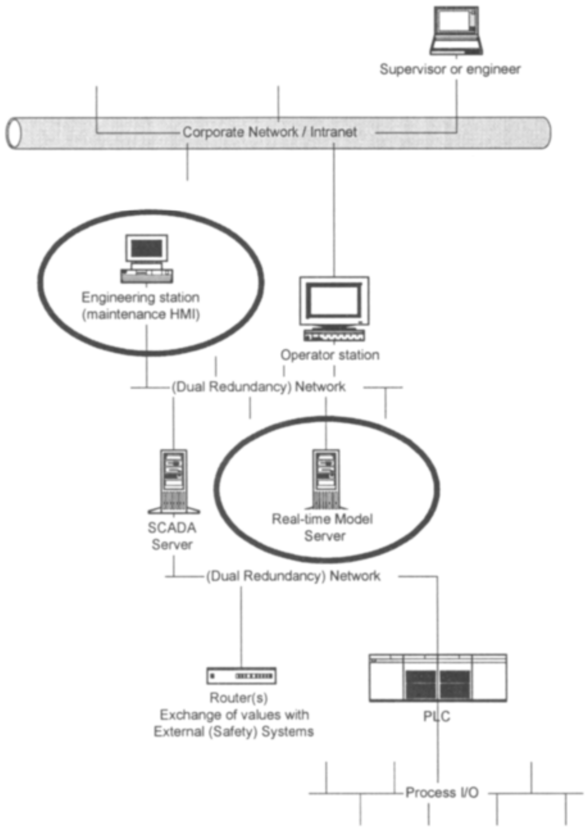

Supervisor or engineer
Corporate Network / Intranet
Engineering station (maintenance HMI)
Operator station
(Dual Redundancy) Network
SCADA Server
Real-time Model Server
(Dual Redundancy) Network
Router(s) Exchange of values with External (Safety) Systems
PLC
Process I/O

Figure 10. Systems for a Real-time model

5.3.9.3 An example: Monitoring of pipelines

Statement of Problem

A pipeline monitoring system uses a high-fidelity model of a hydrocarbon transport network to provide operational information to the system's operators. The key requirement for a PMS is that it will raise an alarm when a leak occurs in the system. This is called leak detection. In addition, the system then uses the model to predict where the leak is most likely to have occurred. This leak location is important information, as the pipelines monitored are often several hundred kilometres long and run through inhospitable terrain.

Scrapers, or pigs, are routinely sent through piping systems to clean out deposits and reduce pressure drop. Smart scrapers can also be used to collect information for preventative maintenance. The position of a scraper in the pipeline depends on the fluid flow, composition and topography. This can be readily calculated by a model, but is difficult to predict using rules-of-thumb. An on-line *scraper-tracking* tool is used to warn operators of an approaching scraper and to raise an alarm if a scraper is stuck in the pipe. Related, *batch tracking* calculations are relevant for multi-product pipelines, where it is important that the interface between different products (such as diesel, petrol and avgas) is tracked along the pipeline, so that operators can minimise the loss of off-spec product at the receiving station. Long pipelines contain appreciable amounts of fluid. They are therefore used as storage buffers to balance supply with demand. In gas pipelines, pressure can be increased in anticipation of demand peaks. The calculation of *line pack*, which is the amount of fluid in a pipeline, is therefore a vital tool for optimising profit and customer satisfaction. Finally, the line pack can be used to calculate and report a *survival time* for the pipeline. This is the time during which normal supply can be maintained to customers after a stoppage in supply to the pipeline.

A PMS uses a detailed model of a pipeline to calculate and report these control parameters, thereby providing a "window into the piping network" for the operator. Operators thus gain access to calculated results that are not available from instrumentation alone. Safer, more efficient operations result from this.

The PMS System

The operation of the PMS system is shown in Figure 11. Measurements of pressure and flow from the pipeline system are read from the control system. In addition valve positions (which can analog or digital signals) and equipment status is read. This data is checked and validated so that it does not cause spurious results when entered into the model.

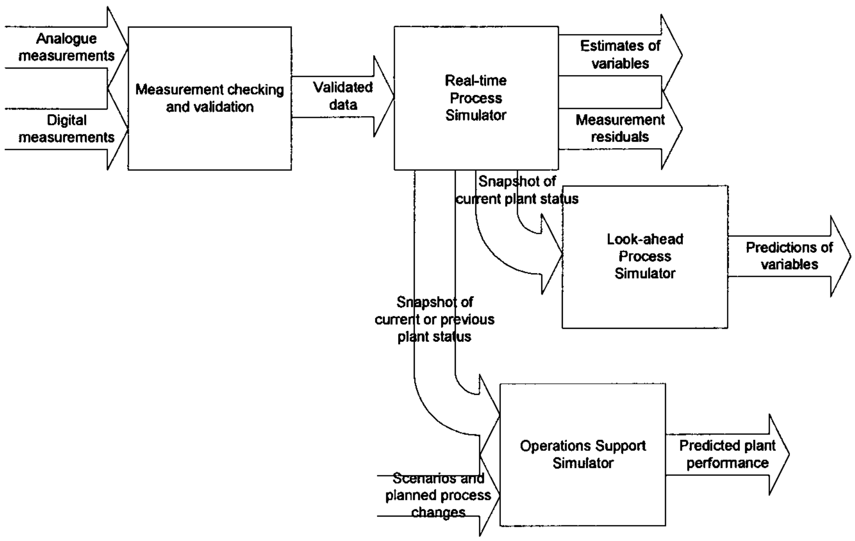

Analogue measurements
Digital measurements
Measurement checking and validation
Validated data
Real-time Process Simulator
Estimates of variables
Measurement residuals
Snapshot of current plant status
Look-ahead Process Simulator
Predictions of variables
Snapshot of current or previous plant status
Operations Support Simulator
Predicted plant performance
Scenarios and planned process changes

Figure 11. Data flow in an online simulator.

A selection of this data is used to drive a dynamic model of the process. Some of this data is used as direct specifications of boundary conditions, whilst other data is used to tune the model over a long time horizon. The rest of the data can be used to generate measurement residuals, which can be used to detect leaks.

At any given time, a snapshot of process conditions can be taken and used by the look-ahead or the operations-support (planning) simulator.

Technical Challenges

The technical challenges involved in real-time applications lie in the following areas:

- Maintaining robust operation in a critical process environment.
- Delivering software that meets real-time, safety-critical quality requirements.
- Properly handling inadequacies in the process data supplied to the real-time system.

An on-line model must be able to run for long periods with predictable and low maintenance requirements. It must be good-quality software. Errors, such as memory leaks, which are not usually critical in off-line models, are not permissible in an on-line application. For this reason, the modelling calculations and user interfaces are usually supported by separate, independent programs. These programs communicate using a common protocol, such as TCP/IP. This structure is robust, as the modelling program is freed from the (error-prone) complexities of user interaction. System availability is thereby improved.

Robust on-line models require robust numerical methods. Pipeline models are complicated PDE modules with time-dependent two-point boundary conditions (Matko *et al.* 2000). Work needs to be been devoted to developing stable, accurate interfaces between these specialised models and the simulation platform's algorithms for integration and solving flow-pressure networks.

Finally, a mix of heuristics and statistical methods is needed to ensure that the data used to drive the on-line model is correct. Furthermore, effective fault detection requires that measurement residuals that are generated by a fault are differentiated from a model error. This decision often requires process knowledge, as statistical methods, by themselves cannot differentiate between model mismatch and a fault. Making on-line systems robust to measurement and modelling errors remains the main challenge for implementation.

5.3.10 FUTURE PERSPECTIVES

5.3.10.1 Faster, better and friendlier – but complex!

Commercial dynamic simulators have evolved into powerful, relatively user-friendly tools over the last decade. As computers continue to decrease in price and increase in power, we can expect models to be built of ever larger and more complex process systems. We can also expect that the application of web technology, object-oriented development and virtual reality will make process models easier to configure and simpler to use.

5.3.10.2 Expansion through the lifecycle

Applications of dynamic simulation are currently concentrated in the detailed design and start-up phases of the process lifecycle.

5.3.10.3 Better integration with the design process

Much of the engineering time involved in developing and implementing a dynamic model of a process lies in entering and maintaining the parameters in the model. These parameters (such as vessel diameter, weir height, valve discharge coefficient or valve type) are results from the process design and a documented on equipment data sheets. However, these parameters may need to be revised as the result of dynamic simulations. For example, a valve may be found to be too large or to have the wrong characteristic for given control task.

For existing plants, this design data lies in a paper-based or, increasingly, a digital archive. However, the digital archive often only contains scanned copies of the former paper-based archive. Thus, diameter of the vessel still has to be read manually off the scanned datasheet. Some design organisations are using process design databases (such as Zyqad or AXSYS) to store design sheets *and their information* for electronic retrieval. Such databases make it possible to exchange design data with the dynamic simulator. However, this requires proprietary solutions, and a simulator vendor needs to support multiple interfaces.

Perhaps the answer to this problem lies in the use of a neutral data transfer format. Initiatives, such as pDXML and PI-STEP, which are described Chapters 4.2 and 4.3 of this book, could meet this need.

5.3.10.4 Combination with virtual reality

Dynamic simulators have been already integrated with virtual reality (VR) systems in the nuclear industry. For example, a VR control room, called HAMMLAB-2000 is being built at the OECD facility in Halden, Norway. This system will incorporate training simulators of a variety of reactors (Nuclear Energy Agency, 1997).

Older training laboratories for training in the oil industry were based on physical replicas of the control room. However, such installations are now viewed as being expensive and unnecessary. However, similarity to the operator's actual working conditions is desirable, VR facilities can be used to deliver this similarity at lower cost.

This above application is very prosaic. We believe that with imagination, VR tools can be applied to make dynamic simulation more accessible and more compelling to for engineers and decision makers.

5.3.11 NOMENCLATURE

F	Molar flow, kmol/s.
ε	Molar production of a component, kmol/s.
h	Specific enthalpy kJ/kmol.
H	Enthalpy holdup, kJ.
ΔH	Internal production of enthalpy, kW.
n	Molar holdup of a component, kmol.
P	Pressure, kPa.
q	Flow of heat to a control volume, kW.
T	Temperature, K.
V	Volume of a control volume, m^3.
x	Mole fraction of a component.

5.3.12 REFERENCES

Aldren, D.B., (1986), "Development of a Low Cost Dynamic Training Simulator for a Chemical Plant", *Proc. 2nd IEE Intl. Conf. on Simulators,* Warwick, UK, 7-10 September.
Askaner, M., (1999), "Simulerad process lättare att utveckla (Simulated process easier to develop)", in Swedish, *Kemisk Tidskrift / Kemivärlden,* October, 16-18.
Barton, P.I., and Pantelides, C.C., (1994), "Modelling of combined discrete / continuous processes", *AIChE Jl.*, **40**(6), June, 966-979.
Bequette, B.W, (1998), "Process Dynamics: Modeling, Analysis and Simulation", Prentice Hall.
Breitenecker, F., and Lingl, M., (1998), "Comparison 7 –ACSL Hybrid Modelling Approach", *Simulation News Europe,* **22**, March, 42.
Gauthier, J.S., (1999), "Comparison 1 – ACSL Graphical Modeller Numerical Approach", *Simulation News Europe,* **26**, 35.
Cellier, F.E., (1991), "Continuous System Modeling", Springer Verlag, New York.
Ek, Arild, (2000), "Feltsimulator – et nytt verktøy for petroleumsindustrien", *Energien,* **2000**(2), 16-17.
Endrestøl, G., Sira, T., Østenstad, M., Malik, T., Meeg, M., and Thrane, J., (1989), "Simultaneous Computation within a Sequential Process Simulation Tool", *Modeling, Identification and Control,* **10**(4), October, 203-211.
Jones, D.R., Brook, J., (1990), "High fidelity real time simulation and its application to olefins plant operator training", in Bussemaker, H.Th., Iedema, P.D, (eds.), *Computer Applications in Chemical Engineering,* Elsevier, Amsterdam, 1990, 217-222.
Luyben, W.L., (1990), "Process Modeling, Simulation and Control for Chemical Engineers", 2nd edition, Mc-Graw Hill, New York.
Matko, D., Geiger, G., Gregoritza, W., (2000), "Pipeline simulation techniques", *Mathematics and Computers in Simulation,* **52**, 211-230.
Modelica (2000), "Modelica Specification 1.4", available from http://www.modelica.org. December.
Nuclear Energy Agency, (1997), "Meeting Summary of the Second CSNI Specialist Meeting of Simulators and Plant Analyzers. Current Issues in Nuclear Power Plant Simulation", NEA/CSNI/R(97)36, available from: http://www.nea.fr/html/nsd/docs/1997/csni-r1997-36.pdf
Nuclear Energy Agency, (1998), "Principal Working Group No. 1 – Extended Task Force on Human Factors. Task 5: Role of Simuators in Operator Training", NEA/CSNI/R(97)13, 29th June, Paris.
Nuclear Energy Agency, (2000), "Pressurised Water Reactor Main Steam Line Break Benchmark Workshop. Summary of the Fourth Workshop. 24th and 25th January 2000, Château de la Muette, Paris", NEA/NSC/DOC(2000)2, available from: http://www.nea.fr/html/science/docs/2000/nsc-doc2000-2.pdf

Perry, J.L. and Allan, B.A., (1996), "Design and Use of Dynamic Modeling in ASCEND IV", Carnegie Mellon University, EDRC Technical Report 06-224-96, http://www.ndim.edrc.cmu.edu/~ascend/pdffiles/AscendIVP.pdf.
Puska, E.K., (1999), "Nuclear reactor core modelling in multifunctional simulators", VTT Publications 376, Espoo, available from http://www.inf.vtt.fi/pdf/,
Shewchuk, C.F., and Morton, W., (1990), "A New Dynamic Simulator for On-line Applications", in *Proc. 1990 TAPPI Engineering Conf., Atlanta,* 595-603.
Smith, G.J., (1985), "Dynamic Simulation of Chemical Processes", Ph.D. Dissertation, University of Cambridge.
Smith, G.J., and Morton, W., (1988), "Dynamic Simulation using an Equation-Orientated Flowsheeting Package", *Comput. Chem. Engng,* **12**(5), 469-473.
Womack, J.W, (1986), "Dynamic Simulation Gives 20-20 Foresight", *Oil Gas Jl,* April 7th, 66-76.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 5.4: Computer tools for Discrete/Hybrid Production Systems

L. Puigjaner, M. Graells & G.V. Reklaitis

5.4.1 INTRODUCTION

Hybrid production systems participate of both, continuous and batch/discrete process characteristics. Supporting computer tools for such systems should therefore comprise software for steady state process simulation and analysis, as well as support tools for the handling of batch process dynamics and discrete decisions involved.

The software tools for batch processes may be classified in two broad classes. The first class involves modelling, simulation, and analysis of the physico-chemical processes that take place during batch operations. These tools can be associated with the process simulation methodology that constitutes the core of the first Cape Open project. The second class is designed to support the decision processes at the different managerial and operational levels of the plant operational hierarchy, which are the topics included in the scope of the follow-up, Global Cape Open Project. This class of tools may be further classified on the basis of the planning, scheduling, monitoring, and control tasks, which they support, specialized to the batch processing case. In this chapter we review a representative set of tools from these two classes that are available commercially. While there is much innovative research and development in both academic and industrial groups in this domain, we have excluded experimental software and proto-types from our discussion.

5.4.2 PPROCESS SIMULATION & ANALYSIS

5.4.2.1 Continuous process simulation

Process simulation systems were initially conceived to allow modelling of continuous steady-state process flowsheets, as represented by classical petrochemical processes such as the hydrodealkylation of benzene or the production of ethylbenzene. The process flowsheet model generally consisted of models of the unit operations present in the flowsheet that were linked through the process streams which constituted the external input and output variable sets for these models. The solution of the flowsheet model was complicated by the presence of recycle streams, which created linkages between the unit models. To solve the resulting large scale coupled algebraic equation systems a natural decomposition strategy called the sequential modular approach was widely adopted. Under this approach unit models were executed in a serial fashion in the direction of the principal process streams with the recycle stream, or more precisely, tear streams serving as iteration variables. AspenPlus (Aspen Technology.), HYSIM-HYSYS (Hyprotech/AEA) and PRO/II (Simulation Sciences) remain the most significant examples of this type of simulation technology (Fig. 1). These tools gradually evolved to contain substantial libraries of unit operations modules, including some which involved models consisting of differential equations, as well as extensive physical property estimation capabilities and supporting properties constant databases. Because of the desirability of allowing the easy addition of new process specific unit operations models or modifications of existing models, the need soon arose to allowing linkage or insertion of user-added FORTRAN subroutines. In recent years, this in part served to stimulate the idea of COSE (Cape Open Simulation Executive). In due course, with advances in efficient methods for the solution of large-scale algebraic systems, these methods were adopted to solve the entire set of flowsheet model equations simultaneously, resulting in the so-called equation oriented flowsheet simulation architecture.

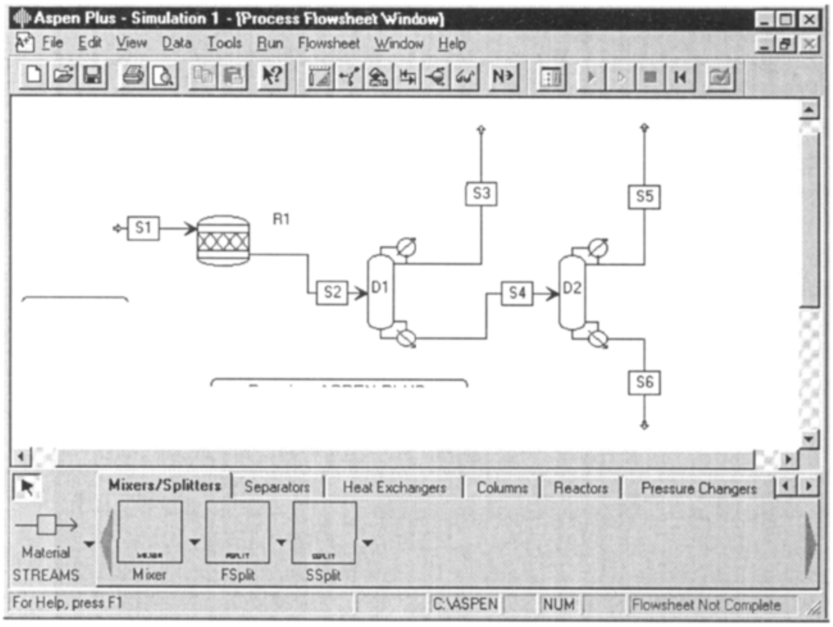


Figure 1. Continuous process simulation. The AspenPlus Environment.

A natural next step for flowsheet simulation was to provide the capability to model the dynamics of the process. Again the solution of such dynamic simulation models was initially approached using sequential modular strategies but contemporary systems have emphasized simultaneous solution methods. Tools such as DYNSIM (Simulation Sciences), HYSYS (Hyprotech/AEA) and AspenDynamics (Aspen Technology) embodied these architectures (Fig. 2). Ideally such tools offer a consistent set of steady state and dynamic models of the same unit operation, allowing the user the convenience of ready transition from steady state to dynamic simulation of a given process. The DynaPLUS system marketed by Aspen Technology is intended to achieve precisely that aim.

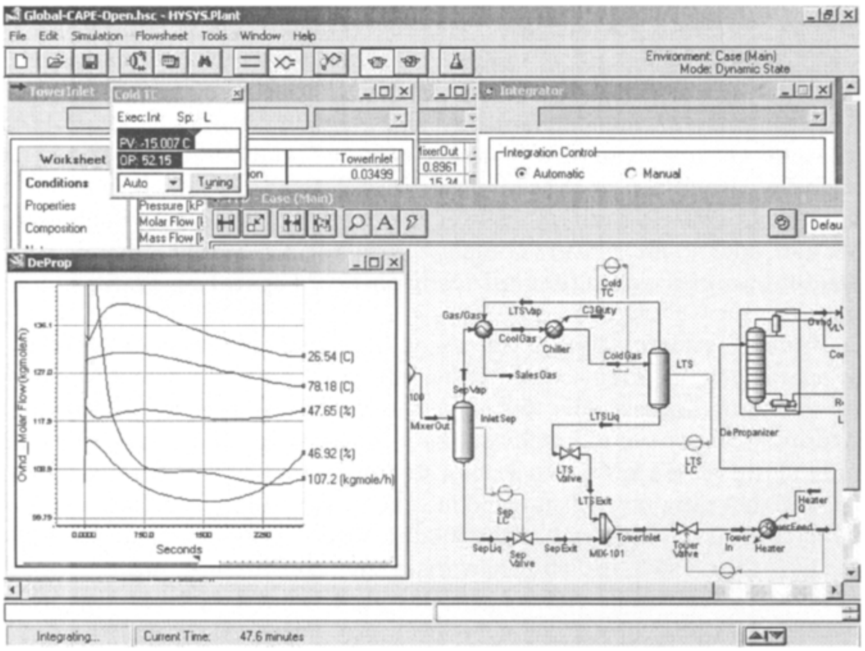


Figure 2. Dynamic continuous process simulation. The HYSYS environment.

5.4.2.2 Batch process simulation tools

Since batch operations are inherently dynamic, one would not expect steady state flowsheet methodology to be applicable to batch process simulation. On the other hand, one might well assume that dynamic simulation systems would be extensible to handle batch processes. While this is the case for individual operations, it is not when one seeks to model the entire network of batch operations typical in batch chemical processing. The complicating factors include the need to handle the discontinuities inherent in the start and stop of the tasks which comprise a batch process and the fact that with batch processes the description of the set of chemical–physical tasks which must be executed to manufacture a given product (the recipe) is distinct from the set of equipment which are used to perform these tasks. Since the equipment items are generally multipurpose, the definition of the flowsheet for a batch process requires a series of task to equipment assignment decisions which may be governed by equipment availability, the availability of resources such as feedstock's, catalysts, and hold tanks, product priorities, and other state dependent or even economic factors. Thus, in the batch case, the flowsheet is in effect defined dynamically as the recipe is executed. These additional numerical and decision aspects are not accommodated by conventional dynamic flowsheet simulation systems.

Of course, simulations of individual batch operations, such as batch reaction and batch distillation, can be modelled and solved within the framework of dynamic simulation methodology. These simulations are most commonly structured as stand-alone modules or programs, which either share the physical properties estimation features of an associated process simulation system or employ an independent physical properties package. BATCHFRAC and BatchCAD are examples of these types of commercial systems. Other examples of programs of this kind include Batch Colonne, Batch Réacteur (ProSim) and BDIST-SimOpt (Batch Process Technologies). Such systems may offer additional features such as operating profile optimization or parameter optimisation to fit empirically observed operating profiles.

To facilitate the simulation of processes involving a mix of continuous and batch operations, as might arise in single product plants in which some of the operations must necessarily be of a batch nature, several of the process simulators do allow use of batch reactor and batch distillation modules, of the type noted above, within a continuous steady-state model of the process. As is the case with a plug flow reactor model, the integration of the batch operation model is carried out internal to the module and averaged output streams are computed for use by the downstream continuous unit operation. Conceptually, this can be viewed as following the batch operation with one or more implicit holding tanks, which at the termination of the batch effectively provide the averaged output stream or streams, which feed the succeeding continuous unit. This type of linkage of steady state and dynamic model types can readily be accommodated in the sequential modular architecture since all unit operations models are treated as closed procedures. It can also be handled within equation-oriented systems provided such system also accommodates closed procedures. Although the AspenTech and SimSci steady-state products thus do permit interfacing of batch and continuous operations in this fashion, the entire process model remains effectively steady state.

As noted earlier, the effective simulation of batch processes requires representation of the dynamics of the individual batch operations, the decision logic associated with the start and stop of operations, as well as the decisions associated with the assignment of equipment and other resources to specific operations as defined through the product recipe. Some conventional dynamic simulators (*e.g.*, HYSYS) do offer tools for programming the decision logic associated with a series of events to be executed at specific points during the execution of a simulation run. In this way it is possible to simulate certain classes of batch processes. However, this type of adaptation of the dynamic simulation executive only address part of the requirements for the simulation of typical multiproduct batch processes. The more advanced capabilities of a combined continuous-discrete simulation architecture are required to accomplish this in a general fashion. The BATCHES (Batch Process Technologies) simulation tool does accommodate the above mentioned batch process features and uses advances in combined discrete-continuous dynamic simulation methodology to follow the progress of the batch plant over time.

A BATCHES simulation model consists of three main building blocks: a recipe network, an equipment network and a set of processing directives. The equipment network defines the equipment specifications and the connectivity or transfer limitations between the equipment. The recipe for the manufacture of a product is modelled as a network of tasks and each task as a sequence of subtasks. A task consists of all of the operations performed in a single item of equipment: a subtask consists of a model of one of these operations. Tasks have associated with them requirements of specific types of equipment and selection priorities. BATCHES provide a library of models of various types of operations (heating, cooling, decanting, batch reaction, etc.). Subtasks may have associated with them additional requirements for resources types and levels such as operator types and utilities as well as definition of conditions under which the subtask is to be terminated. These may be state dependent (a specific temperature or composition level is achieved) or directly specified (completion time or duration). Processing directives consist of information that drives the execution of the process over time. These include information such as the amounts and sequences in which the various products are made, the due date and amount for finished product delivery, or the amounts and frequency of raw materials deliveries or other resource releases.

A sample equipment network shown in Fig.3 will indicate symbols of the equipment items, their input and output ports, and the connectivity of the equipment items. The recipe network, illustrated in Fig.4, indicates the subtasks involved in each task, the subtasks between which material transfer occurs, as well as the order in which tasks are to be executed. In this example, the recipe for product A involves nine tasks but of these only three involve more than one subtask. The results of a simulation typically consists of extended Gantt charts which indicate the dynamic status of each item of equipment over time, as shown in Fig.5, as well as plots of any dynamic variable such as tank levels, materials utilization profiles, and other resource level plots. The equipment network diagram can also be used to animate the simulation, that is, to playback the dynamic progression of events arising during the course of the simulation. For instance in Fig. 3, the green lines indicate the active transfer of materials between vessels, the vessel coloring can show the material level in the vessel, and gauges and bar plots can show current utilization levels of key shared resources.

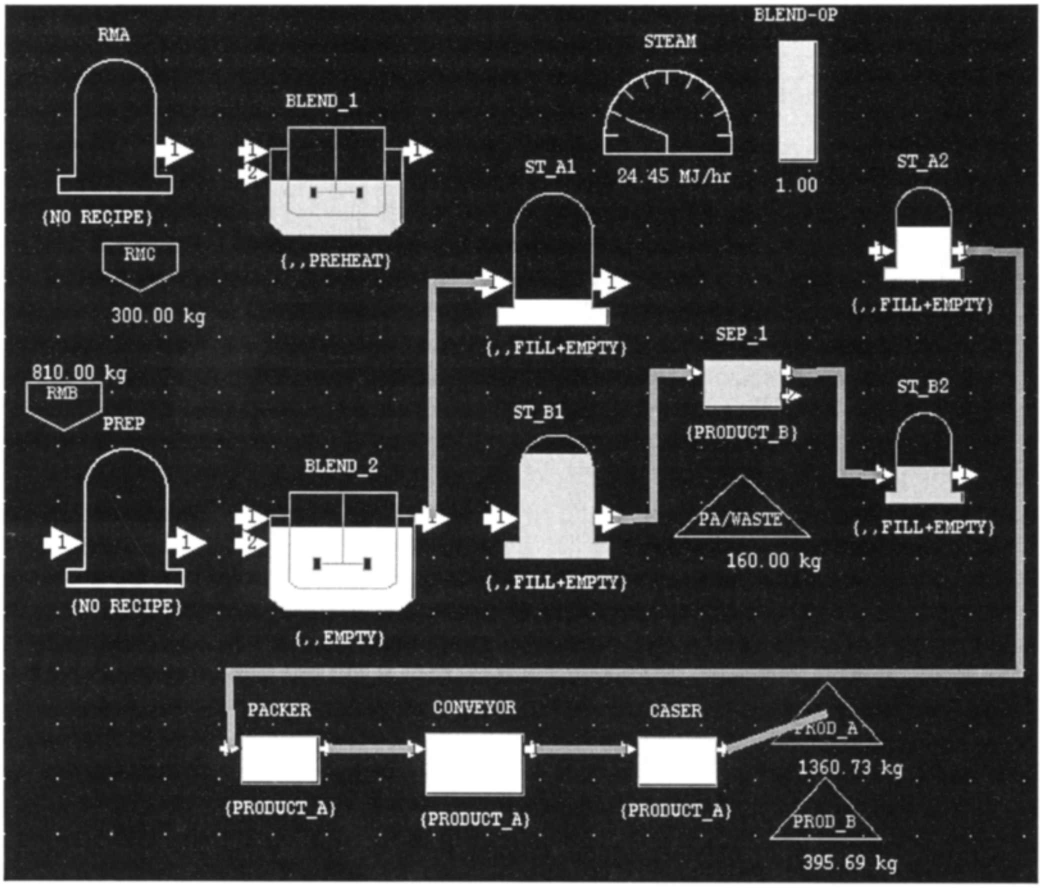


Figure 3: Example BATCHES equipment network

BATCHES uses a dynamic solution strategy under which the dynamic models associated with all of the subtasks that are active in a given time period are solved simultaneously using a DAE solver. As the solver advances through time, the occurrence of subtask termination or start events is tested at each solver time step. As events are identified and occur, the set of active subtask models is reconfigured and the solution process continued. This computational approach is effectively a decomposition strategy as only the models of the subtasks active at a given point in time are actually included in the integration step executed.

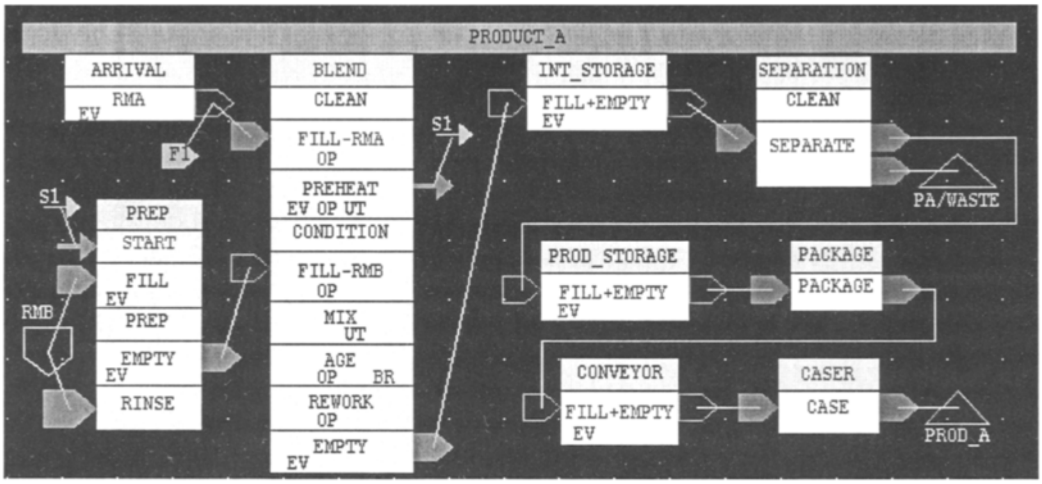


Figure 4. Example BATCHES Recipe Network

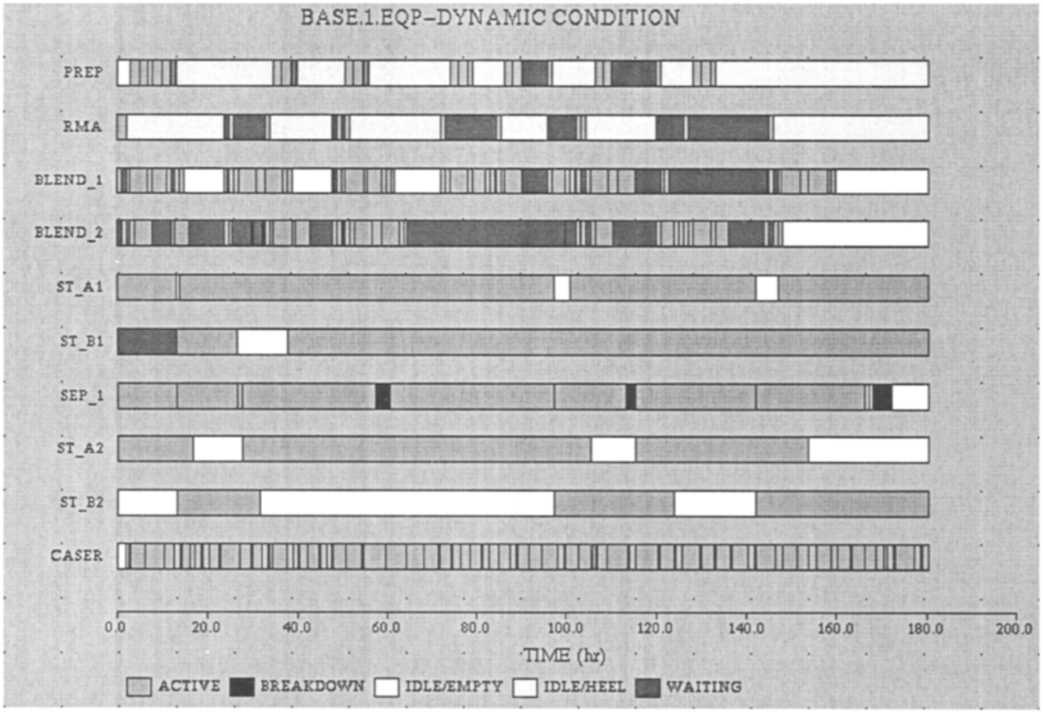


Fig.5: Example BATCHES Equipment dynamic status plot

To accommodate stochastic parameters, BATCHES allows Monte Carlo sampling of simulation parameters from a library of distributions using established techniques from the discrete event simulation literature. It also provides linkages to physical properties estimation capabilities. More complex decision processes, such as solution of an assignment or scheduling model, can be accommodated by defining event conditions under which the simulation is interrupted, the information necessary to execute the decision model is assembled, the decision model solved, and the simulation of resulting actions transferred back to the simulation executive. In this fashion, the simulation can be used to study the effects of various dispatching or scheduling methods under deterministic or stochastic conditions.

5.4.2.3 General purpose modelling languages

As in the steady state simulation case, the library of subtask models and decision logic options provided with the simulation system is sufficient to represent most applications. Occasionally a specific feature of the application may require a subtask model not available in the simulator library. In BATCHES non-standard subtask models as well as unusual decision or event logic can be accommodated using the vehicle of user-supplied models written in conventional programming languages. This is a limitation, which the tool shares with conventional steady state and dynamic simulators that do not provide a higher level modelling language. That limitation is mitigated by general-purpose process modelling and simulation software such as Speedup, gPROMS and Abbacus. gPROMS is a successor to Speedup and is in turn superceded in part by the academic package, Abbacus. gPROMS does accommodate the full range of model types, from purely batch to purely continuous. It allows model developers to write models of the most complex processes and their operating procedures – from the detailed mathematical equations for individual components, to the structure and operation of large complex systems composed of many such components – using a sophisticated natural language. The complexity of the processing of the resulting equations and the solution methodology are handled through system utilities and thus can largely be hidden from the user. gPROMS offers extensive facilities for linking to external software across a range of hardware and software platforms. It also provides advanced features such as dynamic optimisation of continuous dynamic models thus allowing simultaneous optimisation of the parameters of equipment and operating procedures. Of course, since it is a modelling system, it leaves to the user the definition and formulation of the models and particular decision processes of batch operations. Moreover, in practice, users are generally reluctant to build mathematical models and prefer to assemble models using the blocks provided through a model library.

5.4.2.4 Batch process development and information management

An alternative approach to supporting the development and operation of batch processes is to offer a software package that provides the capabilities of organizing and managing recipe information together with a suite of tools for operating on that information, including a rudimentary recipe simulation capability. Linkage to more detailed tools such as stand-alone batch reactor and batch distillation packages can also be provided. This is the approach that has been used in two of the packages described in this section. The additional functionalities not provided by these tools, namely the generation of operating procedures and the execution of batch process hazards analysis, are available from IPS. We describe these developments in the third part of this section.

Batch Plus (Aspen Technologies) is a general-purpose system designed to model the complex, recipe-based processes found in the batch process industries. The key design concept is the representation of a batch process using the multilevel "process recipe" metaphor. Based on the recipe description of the process, software allows design, engineering, and scale-up to be performed. Among features provided is the AspenTech electronic Batch Record System (AeBRS) which is designed to address manufacturers' needs in the areas of work orders management, procedure management and documentation management.

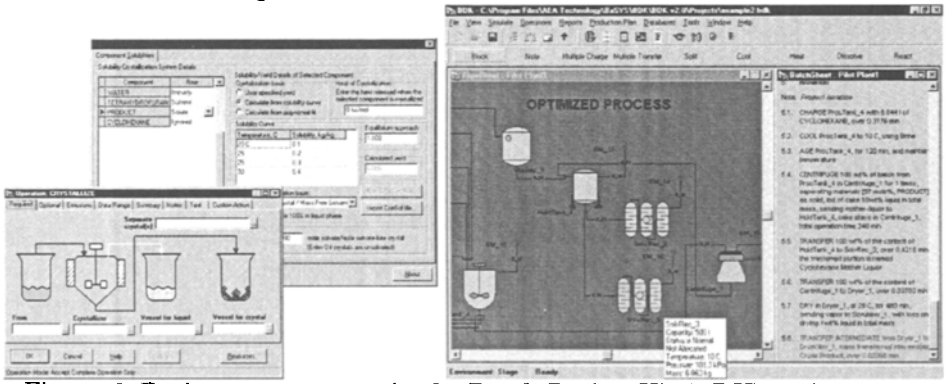


Figure 6. Recipe management in the Batch Design Kit (BDK) environment.

The BaSYS system of software is designed to help batch processing companies improve communication between chemists and engineers, perform faster process scale-ups and more efficiently allocate existing equipment. BDK, the core of BaSYS, is an integrated batch process development and design environment that helps companies accelerate many aspects of batch process development, from route synthesis to implementation. BDK offers tools for enabling the rapid selection from among alternative synthesis route for manufacturing, improving waste processing and facility utilization, and providing a "knowledge warehouse" that documents the development process (Fig. 6). Neither of these systems are intended to provide rigorous process simulation capability but could in principle be interfaced to a simulator, given that much of the recipe and other data required for a simulation is contained in the information base that these systems maintain.

Two critical tasks in the development cycle of a batch process are the synthesis of operating procedures given basic recipe information and the analysis of the operating procedure to identify, evaluate and mitigate potential operating hazards. These two functionalities are not provided by the systems described above. However, the generation and validation of operating procedures for new processes constitute time-consuming and error prone activities that lend themselves to computer support. Moreover, with the increasing complexity of operating procedures and governmental and social pressure to reduce safety and environmental incidents, there is strong incentive to conduct process hazards analysis and mitigation for all new and existing processes. Since considerable portions of this activity are also repetitive and very labour intensive, there is an opportunity for intelligent tools to support process hazards analysis. PHASuite, a system initially developed at Purdue University to support this task, is now available from IPS. PHASuite consists of two closely integrated components, iTOPs and Batch HAZOPexpert. The former component serves to synthesize the detailed sequence of instructions an operator in a chemical plant needs to follow in order to manage a process safely and optimally. BHE serves to systematically analyse the process in question to identify, assess, and mitigate the possible hazards that can occur.

PHASuite follows ISA S88 batch standards for modelling batch process information. In iTOPS, Graf chart-based concepts are used to represent the batch process and a hierarchical planning technique is employed together with information about the process materials, equipment, and chemistry to synthesize procedures at the phase level. An example of the Process Sequence Diagram (PSD) that is generated by the iTOPS component of PHASuite is shown in Fig.7. The left hand side of the figure displays the menu for inputting material, chemistry, equipment and recipe information. From this the PSD and then the detailed operating procedure are automatically generated. After this, BHE generates the HAZOP results. This is shown in Fig.8. The HAZOP results table summarizes the deviations, abnormal causes, adverse consequences, safeguards and recommendations for mitigating the hazards for a given task in a given equipment in a process. BHE uses a logical separation of process information into specific and generic components, qualitative causal digraph models and a two-layered Petri-net based model of the process to systematically identify possible hazardous situations that can arise in a chemical process. The two PHASuite components are fully integrated, including the creation of the various representations of the process recipe and logic. The user inputs batch process information – the materials, the chemistry, and the equipment – through a top-level interface. PHASuite returns a complete batch record documents, including safety instructions and a list of potential hazards classified by severity. The systems has been tested extensively in pharmaceutical applications, resulting in documented substantial savings in engineer time, and considerable improvements in accuracy of the resulting operating procedures and completeness in hazards identification.

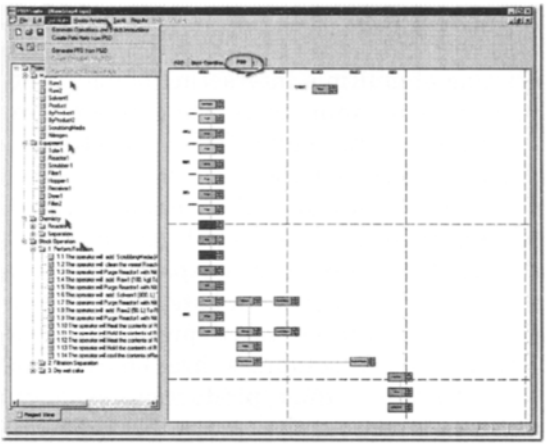
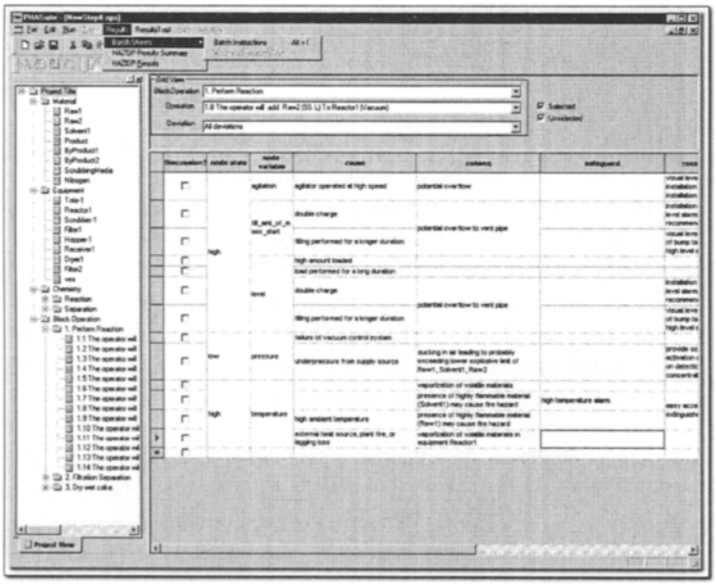


Figure 7. An example of Process Sequence Diagram generated by iTOPS



Block Operation: 1. Perform Reaction

Operation: 1.8 The operator will add Raw2 (50. L) To Reactor1 (Vacuum)

Deviation: All deviations

☑ Selected
☑ Unselected

Discussion?	node state	node variable	cause	conseq	safeguard	[illegible]
☐	high	agitation	agitator operated at high speed	potential overflow		visual leve installation installation
☐		fill_amt_of_mass_start	double charge	potential overflow to vent pipe		installation level alarm recommen
☐			filling performed for a longer duration			visual leve of bump la high level c
☐			high amount loaded			
☐			load performed for a long duration			
☐		level	double charge	potential overflow to vent pipe		installation level alarm recommen
☐			filling performed for a longer duration			visual leve of bump la high level c
☐	low	pressure	failure of vacuum control system			
☐			underpressure from supply source	sucking in air leading to probably exceeding lower explosive limit of Raw1, Solvent1, Raw2		provide ac activation c on detectic concentrat
☐	high	temperature		vaporization of volatile materials		
☐				presence of highly flammable material (Solvent1) may cause fire hazard	high temperature alarm	easy acce extinguishe
☐			high ambient temperature	presence of highly flammable material (Raw1) may cause fire hazard		
☐			external heat source, plant fire, or lagging loss	vaporization of volatile materials in equipment Reactor1		
☐						

Figure 8. HAZOP results from BHE

5.4.3 PPOCESS PLANNING & SCHEDULING

5.4.3.1 Planning

Production planning for individual batch plants and planning for entire supply chains consisting of multiple interacting batch plants can in principle be performed using generic planning tools. Thus, generic linear programming (LP) and mixed integer programming (MILP) packages such as CPLEX (ILOG) can be used providing that the user is prepared to develop the case specific formulation and provide the appropriate data interfaces. Alternatively, if manufacturing recipe details are aggregated and thus the plant treated as a black box, then various capacity planning tools offered by ERP vendors can be applied. However, in batch manufacturing applications, the details of the batch operations often prove to be important because equipment and other limited resources are shared among the various products under consideration, thus production capacity is constrained. Unfortunately, at the present time there are no commercial tools, which can accommodate this level of detail without explicit user developed models. Such a suite of tools is available for petroleum refinery planning and supplies chain applications in the form of Aspen Technology's PIMS and Ref-Sked packages, which are based on LP methodology.

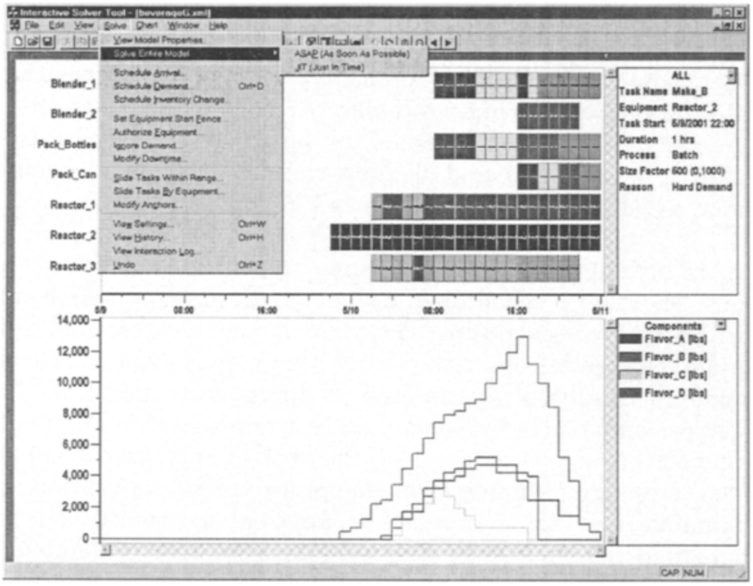
5.4.3.2 Scheduling

As at the planning level, scheduling applications for batch operations can be developed using general purpose scheduling toolboxes such as Scheduler provided by ILOG. Aspen Technology's MIMI system also falls into this category. However, the use of these toolboxes to batch processing problems requires knowledge of the strengths and limitations of the individual tools and experience in scheduling application formulation. This is a level of expertise beyond that of plant engineers. However, several tools do exist that have been developed specifically for batch processing applications, three of which, gBBS, Virtecs and BOLD are described in this section.

Available through Process Systems Enterprise (PSE) Ltd, gBBS is the outcome of extensive research conducted at the Centre for Process Systems Engineering at Imperial College, London. It is a scheduling tool designed for multi-purpose process production, from purely batch to purely continuous. Process specific issues such as cleaning, recycling and intermediate materials that are also final products can all be treated. A complete gBSS application is composed of data such as product demands and inventory, product recipes, plant resources, staff and maintenance schedules and the current status of plant equipment. The user's data is checked for consistency and converted automatically into one of three MILP (mixed integer linear programming) formulations. Specialised MILP algorithms requiring little or no user intervention are used to find the solution that is guaranteed to be optimal if allowed to run to convergence and the results are then processed and presented in an engineering form. When solving especially large problems, gBSS breaks them down into several smaller ones and combines the results into a final solution. Recipes and processes are modelled using the State-Task Network representation and the modelling language is designed to express complex plants in a simple and flexible way. Established models can be accessed from a 3rd-party front-end such as MS Excel.

gBSS can be configured to solve three types of scheduling problem formulations: Short-term scheduling, Campaign planning, and Scheduling for Design. In short-term scheduling, the plant layout, processes and products are known and change infrequently. Other data may change with each run of gBSS - demand, deadlines, availability of equipment and staff. Transport costs and times can be included so that multi-site production and distribution can in principle be accommodated, at the cost of increased computational burden. The campaign planning form is appropriate for applications in which product demands are stable or may be forecast accurately thus allowing longer production horizons to be divided into a number of campaigns. The Scheduling for Design option is used to find the optimum plant resources given a fixed set of demands and their deadlines. gBSS can take account of both capital and running costs and find the ideal plant for a minimum initial cost or a minimum lifetime cost. It is also suitable for designing extensions to existing plants. This facility is available for plants operated in either campaign or short-term scheduling mode.

Advanced Process Combinatorics, Inc. (APC) has developed Virtecs for process scheduling and planning. Virtecs is based on Mixed Integer Linear Programming (MILP) technology and offers the ability to model processes at a high level of detail including constraints on limited material shelf life, process vessel storage, shared storage, labour, utilities, minimum/maximum inventory levels, complex bill of materials, piping connectivity, equipment downtime, multiple stages of production, parallel equipment, and process changeovers. The MILP based scheduling tool can provide solutions in fully automatic mode and a user can readily make process changes. APC has recently released Virtecs v5.0 that includes support for Internet based use and publication of schedules, developing schedules from previous schedules and efficient human override of automated scheduling capability so that the tool can be used anywhere from a fully manual to fully automatic mode. The menu in Fig.9 illustrates the choices available to the user in the interactive or manual mode. These interactive tools can be used to shape the schedule to the user's satisfaction, and, with each user preference carried out by the mathematical program, schedule feasibility is guaranteed.



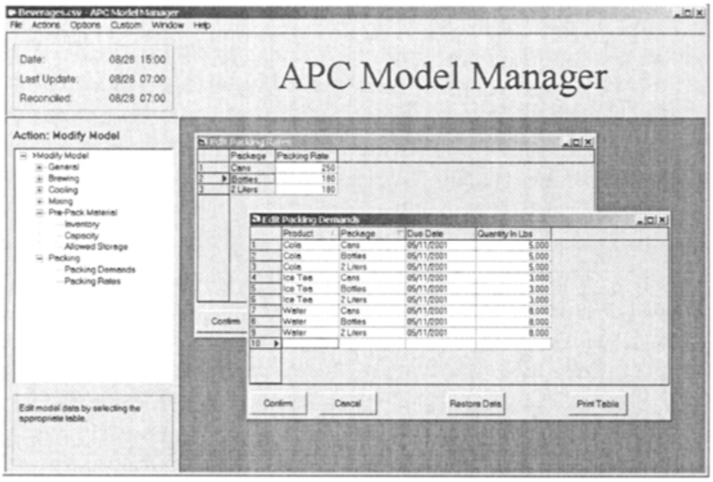
ALL

Task Name Make_B
Equipment Reactor_2
Task Start 5/9/2001 22:00
Duration 1 hrs
Process Batch
Size Factor 500 (0,1000)
Reason Hard Demand

Figure 9. Screen shot from Virtecs – options available

Previous versions of Virtecs have been successfully used in the pharmaceutical, specialty chemical, food and beverage, consumer products, lubricant, and retail industries. The MILP based solver behind Virtecs is highly customized and routinely solves scheduling problems involving hundreds or thousands of tasks in one or two minutes on a desktop personal computer. While MILP scheduling technology is in its infancy and will continue to grow in capability, the existing advantages include explanations of why demands cannot be met on time, lower consulting costs for installing and supporting tool use, ability to support engineering applications such as expansion studies or debottlenecking, online applications that can be readily extended by the user to support new products or process equipment changes. Fig.10 shows what is involved in making a process change. When a model modification is required, the work process for "model building" is described with the familiar "Windows" file structure. From there, the user selects those components (demand and process times are shown) requiring change, and initiates the change. With these actions the revised problem is ready for solution.

Because of reliable and accurate automated solution capability, Virtecs can be used in a distributed fashion to integrate and mediate detailed coordination between multiple facilities. Because of the versatility afforded by their solution technology, APC has also applied Virtecs technology to warehouse management applications and integrates them with upstream production and downstream distribution activities. APC also uses their MILP solver underneath a tool that selects and schedules projects in research and development pipelines. The main components of the Virtecs tool are a natural modelling language for describing processes, reporting system, database for managing multiple scenarios, graphical user interface with readily extensible Gantt Chart, and a highly engineered MILP solver based on implicit formulation generation. Because of the extensibility of APC's MILP approach, their tools can be used in conjunction with simulation capability to analyse the impact of uncertainty on the performance of solutions and manage risk in high-level applications such as pricing studies, mergers and acquisitions, and supply chain design/operations. Because of close ties with Purdue University, APC has world-class research and development capability and has demonstrated significant functionality gains with each new product release.



Date:	08/28 15:00
Last Update:	08/28 07:00
Reconciled:	08/28 07:00

	Package	Packing Rate
1	Cans	250
2	Bottles	180
3	2 Liters	180

	Product	Package	Due Date	Quantity In Lbs
1	Cola	Cans	05/11/2001	5,000
2	Cola	Bottles	05/11/2001	5,000
3	Cola	2 Liters	05/11/2001	5,000
4	Ice Tea	Cans	05/11/2001	3,000
5	Ice Tea	Bottles	05/11/2001	3,000
6	Ice Tea	2 Liters	05/11/2001	3,000
7	Water	Cans	05/11/2001	8,000
8	Water	Bottles	05/11/2001	8,000
9	Water	2 Liters	05/11/2001	8,000
10				

Edit model data by selecting the appropriate table.

Figure 10. Alternatives evaluation with respect to process change

A different approach is contemplated in BOLD (available through CIMADE, SA). Built upon an academic prototype developed by the Universidad Politécnica de Catalunya (MOPP), BOLD follows ISA S88 batch standards in the organization of batch process information. Complex recipes and processing structures are modelled using the hierarchical Process-Stage-Operation representation, while the Event-Operation Network representation model describes the appropriate timing of process operations and associated resources. As in the other two software applications, specific external models can be accessed via standard MS Excel spreadsheet. Which is relevant in this application its ability to handle "flexible recipes" instead of traditional fixed batch recipes. The Flexible Recipe Model is regarded as a constraint on quality requirements and production constraints. In this way, critical process variables are adjusted at the beginning of the batch resulting in an integrated decision framework that links the Master recipe with the initialised control recipe under a performance criterion that may vary from batch to batch and may contain economic as well as process variables information. The solution approach uses the Sgraph algorithm (an accelerated Branch & Bound algorithm) integrated in the flexible recipe model. Actually, BOLD constitutes a suite of tools comprising human resources management (BOLD Work Planner), production planning (BOLD APS), financial aspects (BOLD CFU) and project management (BOLD PMO).

5.4.4 SCHEDULING-CONTROL INTERFACE

Once a scheduling solution is developed using one of the tools described above, it must be implemented in the production environment and executed through the batch control system. At the level of process detail represented in scheduling applications, operational details such as valve opening and closing are normally not taken into account. Moreover, it is usually inefficient to rerun the scheduler with every modest process variation or delay that is encountered. Thus, there is a practical need for either manual or automatic conversion, if necessary, readjustment, and execution of the scheduling solution. The *SUPERBATCH* package, offered by PSE Inc and prototyped at Imperial College, offers the ability to work on-line, in conjunction with standard batch control systems, and to automatically update schedules as small delays and process variations occur (Fig.11). The *TotalPlant* Batch system offered by Honeywell provides the interface for the user to execute batches manually following a schedule created off-line and to manually readjust timing as needed, with all of the recipe and equipment details, communications with the control system, and handling of alarms and messages handled automatically. In this section we briefly describe the functionalities of these two systems. The reader should note that these functionalities are to varying degrees provided by other commercial batch automation systems.

Designed for embedding within manufacturing execution systems (MES) or linkage to a scheduling product, SUPERBATCH provides the capabilities for static off-line short-term schedule readjustment as well as for on line schedule correction, with changes broadcast once a minute to screens in departments throughout the factory.

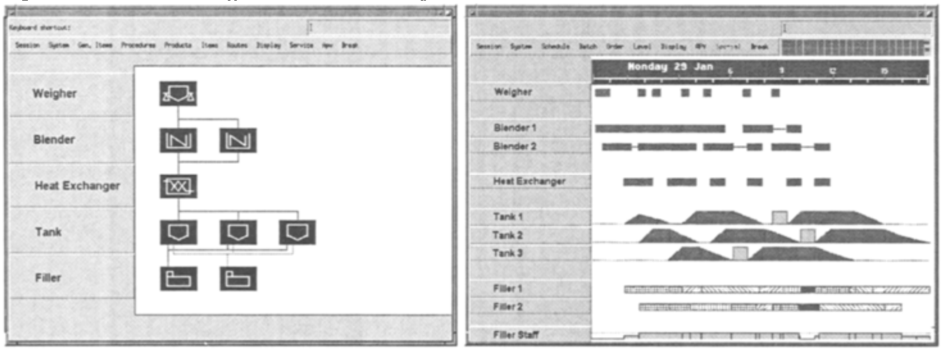


Figure 11. The SUPERBATCH scheduling and control system

It uses a modelling language, which conforms to ISA S88.01 to describe the plant, the materials, recipes and ancillary procedures (such as changeovers and cleaning suitable for hygienic industries) as well as the production batches themselves. For off-line scheduling applications, once embedded within a graphical user-interface to present the schedule, SUPERBATCH will find the earliest possible time for each batch, subject to the constraints of the model. Equipment allocations may be pre-defined or picked by the user from the feasible set which SUPERBATCH offers. SUPERBATCH also delivers the profiles of each plant item as needed to draw a graphical schedule. For on-line schedule adjustments, SUPERBATCH provides an on-line monitor, which executes the schedule, and a versatile interface, which accesses control systems (and simulators) usually across a network. Once a minute, the current status of the plant is read from the control system and the schedule is updated to match. In this fashion, delays and stoppages can be accommodated automatically. SUPERBATCH then initiates execution of any operations due by sending the appropriate commands and parameters to the control system. The system also issues alert messages, which provide advance warning of impending events requiring operator attention. SUPERBATCH is written in object-oriented C++ adhering to the ANSI draft standard, using rpc for it's networking, and is portable to a wide variety of environments and control systems.

The TotalPlant Batch systems provided by Honeywell is an open, object-oriented software application for modular batch automation. Developed around ISA S88.01 it provides batch recipe and equipment management, batch management and execution, an integrated operator interface, easy to use graphical configuration tools, and rudimentary batch simulation capability. The assignment of units can be done dynamically at run time or batch creation time. Implemented to run under Windows NT, it is designed to operate with Honeywell's TotalPlant and the PlantScape control systems. The major components of TotalPlant Batch and their functionalities are the following:

- *Batch View* allows the user to create batches, execute them, review batch related information and respond to alarms and messages.
- *Batch Server* monitors and controls execution of batch procedures and displays batch execution information
- *Batch Data Server* communicates and between the Batch Server and the phase logic sequences in the control system
- *Batch Recipe Editor* allows the user to specify recipe parameters and to graphically construct the recipe sequence using sequential function charts and tables
- *Batch Equipment Editor* allows the user to configure and maintain the physical plant model used by the other components

As might be expected, all of the user interaction with the system is through graphical means. For instance, Fig 12 shows the typical menu based approach to adding a new batch while Fig.13 shows the graphical structures used to configure the sequential control actions followed in batch execution. The system also provides customisable event archiving and batches report generation utilities.

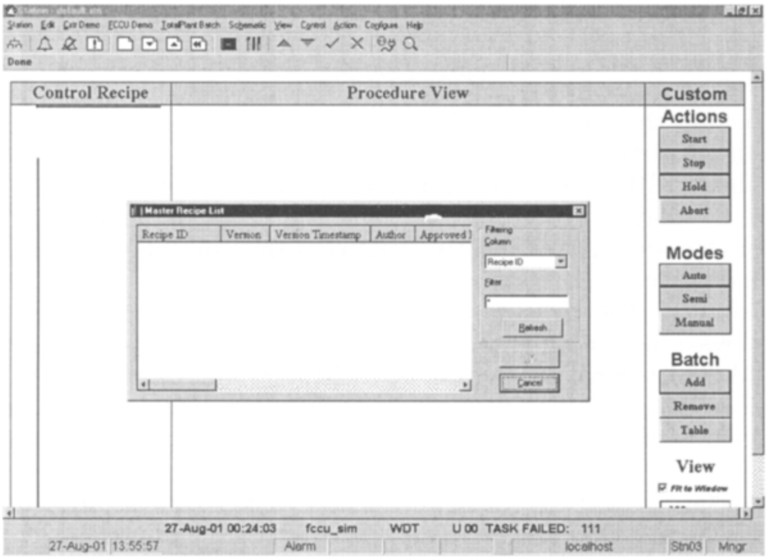


Figure 12. Menu based approach to adding a new batch

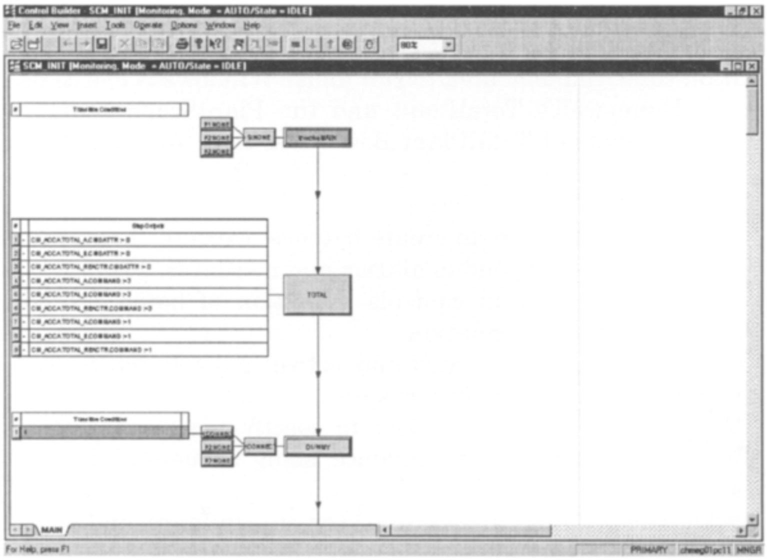


Figure 13. Graphical structures used to configure the sequential control actions followed in batch execution

5.4.5 CONCLUDING REMARKS

As should be evident, the suite of functionalities provided by existing software tools for batch process systems engineering is quite broad but as yet far from complete. While there is much exploratory research conducted within the academic community, commercial developments seem to be lagging. For instance, with the exception of BATCHES there is limited recognition in the existing software of the highly stochastic nature of the operation of most batch processes. Issues of robustness and risk are simply not addressed. Also missing is the capability to perform batch optimisation through the optimal selection of recipe parameters such as conversions, batch times, separation fractions, a capability that is generally available with steady state flowsheet simulation systems. Systematic tools for batch monitoring and fault diagnosis are notably absent: a combination of trend analysis and predictive dynamic models would appear to be required.

Integration of the existing tools is as yet a distant dream. For instance, the seamless linkage of planning and scheduling tools or of scheduling and simulation tools is not available. Furthermore, at present there clearly are no standards that would facilitate the integration of these tools through suitable common date structures. The S88 standard for representing batch recipes and equipment could well serve as the heart of such a common data structure. Indeed a number of the tools have used the recipe structure and naming conventions promulgated under that standard in defining their input data structures. However, this structure must be further elaborated to encompass all of the information associated batch product and process design and batch plant operations.

The batch operations domain evidently offers great opportunities both for methodology research and for commercial software development.

5.4.6 CONTACT INFORMATION

The following list provides contact information for the products and tools cited in this chapter. The reader is invited to pursue the latest developments through these electronic sources.

Advanced Process Combinatorics, Inc.
    Products: Virtecs
    Information: www.combination.com
AEA Technology Engineering Software
    Products: BaSYS, BDK, HYSIM, HYSYS
    Information: www.hyprotech.com info@hyprotech.com
Aspen Technology, Inc.

Products: AeBRS, AspenDynamics, AspenPlus, BATCHFRAC, Batch Plus, DynaPlus, Mimi, PIMS, Re-Sked
Information: www.aspentech.com info@aspentech.com
BatchCAD Ltd
Products: Batch CAD
Information: sales@batchcad.com
Batch Process Technologies, Inc
Products: BATCHES, BDIST-SimOpt
Information: www.bptech.com sales@bptech.com
ChemEng Software and Services
Products: BATCHDIST
Information: ChemEng@BTinternet.com
CIMADE, SA
Products: BOLD, BOLD Work Planner, BOLD APS, BOLD CFU, BOLD PMU.
Information: www.cimade.com, bold@cimade.com
Honeywell, Inc.
Products: TotalPlant Batch
Information: www.iac.honeywell.com
ILOG, Inc
Product: CPLEX, Solver, Scheduler
Information: www.ilog.com info@ilog.com
Integrated Process Solutions, Inc.
Products: BatchHAZOPexpert, iTOPS, PHASuite
Information: www.ipsol.com
Process Systems Enterprise. Ltd.
Products: gBBS, gPROMS, SUPERBATCH
Information: www.psenterprise.com
ProSim SA
Products: ProSimPlus, BatchColumn, BatchReactor
Information: www.prosim.net
Simulation Sciences, Inc.
Products: DYNSIM, PRO/II. PRO/II Batch Module
Information: www.scimsci.com
The Instrumentation, Systems and Automation Society

AKNOWLEDGEMENTS

The financial support from the European Community (project Global-Cape-Open-IMS 26691) and the Generalitat de Catalunya (through CeRTAP - Centre de Referéncia en Tecnologies de Producció") are thankfully appreciated.

Part VI: New Frontiers

Part VI shows where we are going, through new technologies and new potential applications areas that make use of the advances in CAPE and software technologies: e.g. agents, web-based systems, integrated approaches, process monitoring and fault diagnosis systems. Some of the developments in this part are already available but not widely used yet, some still involve significant research before being considered for applications. Specialists at the forefront of CAPE research have contributed to the chapters of this part.

Chapter 6.1 by Batres and Braunschweig. looks at agent-based systems in process engineering. Although these might look a bit futuristic, knowing current architectures and usages of CAPE tools, we decided to include a chapter on multiagents in this book. A number of research prototypes based on the multiagent approach have been developed, and there seems to be a strong trend towards agents in IT as a whole, with e-commerce and negotiation agents leading the way. The chapter shows what software agents can do for you through a set of illustrative examples in process monitoring, distributed design and simulation, process workflow and supply chain management. We hope that you will enjoy it

Application integration is the subject of numerous developments in computer-aided process engineering, but at the moment application integration is generally more a wish than a reality, with few real-world examples of lifecycle integration from process design to operation and control. Chapter 6.2 by Fraga et al. presents a few attempts in integrated approaches, starting with the Epée system by University of Edinburgh, which was further developed into the "Process Web Objects" system, which transposes concepts of Epée in the context of the World-Wide Web. The third example is the ICAS system proposed by the CAPEC group from the

Technical University of Denmark. The rest of the chapter relates these tools to current projects and looks at future trends for integrated approaches, such as grid computing, data management and web-enabled systems.

The following chapter 6.3 by Cameron et al. gives a broad view of web-based process engineering systems. Web technologies are invading the whole spectrum of IT developments, even in scientific applications - although the main trends are defined by the needs of e-business and of global information management systems. The chapter mentions process engineering portals, collaboration infrastructures, remote computing, virtual laboratories and web-based education systems. As Cameron writes: "...the world wide web will change the way we do business at the educational level and at the business level. If you are not actively planning your web-based systems then opportunities may well pass you by".

Chapter 6.4 by Leung and Romagnoli brings us back to equations and algorithms. It is the only chapter on process monitoring and diagnosis, and gives a summary of a number of techniques, model-based or not, used in this domain. Although sophisticated models are built for process design, operation and (sometimes) control activities, the use of such models in the area of fault diagnosis and monitoring is still in its infancy, compared with the use of simpler but computationally efficient modelling techniques. The chapter presents current approaches to building diagnosis systems, alarm filtering systems, and other monitoring tools, using statistical, causal or qualitative representations; with the increase of computer processing power available at the factory, we expect that more and more detailed models will be used for the same purposes.

Finally, chapter 6.5 by Köller et al. analyses the technical results of CAPE-OPEN and develops some of the business models that can be supported by such an interoperability framework. After giving a brief historical view on the use of IT in process industries, the chapter presents the expected benefits of Application Service Provision (ASP), of software components marketplaces, and defines a number of technical requirements for these business models to take place e.g. public catalogues, labelling facilities, brokering services.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 6.1: Software Agents

R. Batres & B. Braunschweig

6.1.1 INTRODUCTION

In the previous parts of this book, we looked at the current technologies for CAPE, based on modern software development approaches: object-oriented design and programming, component architectures, middleware infrastructures, XML for data exchange, and the like.

All these architectures, even though they allow distributed computing on heterogeneous hardware platforms, share the same paradigm for control and co-ordination: a central piece of software controls and co-ordinates execution of all software modules and components that together constitute the model and the solving mechanism of a system. One example is the central piece of software that is usually called *simulation executive*, or COSE in CAPE-OPEN architectures. Its tasks are numerous: it communicates with the user; it stores and retrieves data from files and databases; it manages simulation cases; it helps building the flowsheet and checks model topology; it attaches physical properties and thermodynamic systems to parts of the flowsheet; it manages the solving and optimisation algorithms; it launches and runs simulations; *etc.* All other modules (*e.g.* unit operations, thermodynamic calculation routines, solvers and optimisers, data reconciliation algorithms, chemical kinetics, unit conversion systems *etc.*) are under control of the simulation executive and communicate with it in a hierarchical manner, as disciplined soldiers execute their assignments and report to their superiors (Figure 1).

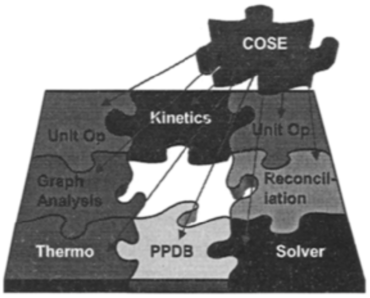


Figure 1. A COSE-centric distributed architecture

In this chapter, we look at examples of distributed architectures based on multi-agents technologies where control and co-ordination are decentralised. In these technologies, each piece of software, each module, each component, generically called *agent*, lives its own life with the ability to negotiate and co-ordinate with other components in order to solve problems such as process design, fault diagnosis, or supply chain management (Figure 2).

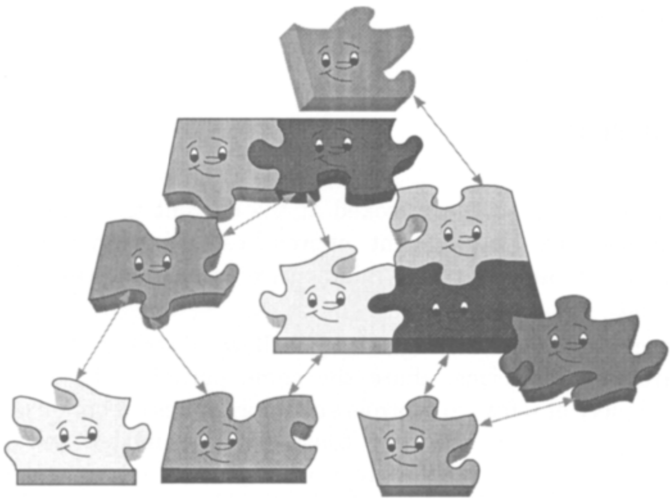


Figure 2. A multi-agent system (based on [1])

This might look futuristic and adventurous for many readers, since, at the time of publishing this book there is not even one commercial implementation of multi-agent systems for CAPE applications. But most of us might have used individual software agents in office applications or on the web.

A familiar software agent is the companion present in MS-Office products, among others. This invisible or iconised component, always active in the background, monitors what the user types, sometimes correcting spelling mistakes, sometimes suggesting better use of the word processing software, sometimes saving work without telling the user. Companions for process simulation software already exist in proprietary environments. These tools help to enter data, to select models, to check the solvability of a flowsheet, to determine the number of unknowns, etc.

Other virtual friends monitor web pages for us; they search for books, auto parts, CDs, *etc.* on our behalf. They crawl the Internet looking for information of interest to us. These are single-agent applications in which the user configures a software agent so that it executes the boring task of querying the web in search for data. At least, we can clearly see the benefits of using such *avatar agents*[1] for doing repetitive tasks, thus saving our precious time for doing more interesting activities.

[1] Avatar: a software agent that represents its human owner; avatars can be simple agents or human-like 3D representations living in virtual worlds, e-marketplaces, etc.

Multi-agent systems extend the single agent paradigm and add co-ordination between several agents, thus letting collective behaviour emerge from the combination of much individual behaviour.

The MIT Encyclopaedia of Cognitive Sciences [2] gives the following definition: "Multi-agent systems are distributed computer systems in which the designers ascribe to component modules autonomy, mental state, and other characteristics of agency. Software developers have applied multi-agent systems to solve problems in power management, transportation scheduling, and a variety of other tasks. With the growth of the Internet and networked information systems generally, separately designed and constructed programs increasingly need to interact substantively; such complexes also constitute multi-agent systems."

Now, the average reader will say: "what's in it for me?" Is there any chance that multi-agents systems will have an impact on future CAPE applications? There are at least two good reasons for giving a positive answer to this question.

First, agent technology can be seen as the future of software development. Object technology and component technology are moving towards agent-based representations. In addition to the standard facilities gained using an object-oriented or a component approach (encapsulation of behaviour, inheritance mechanisms, notion of *service provider, etc.*), the agent-based approach adds consciousness of the collaboration between objects and components; multi-agent technology adds collaborative interaction frameworks within a population of software components. For example, the RoboCup simulation league, an internal competition between soccer simulation programs, uses a CORBA standard interface for the communication between the simulated players, referee, ball, and field. With such an interface, a simulation of a soccer game is seen as 22 playing agents (plus the rest) interacting in order to win a game. Similarly, it could be possible do develop a whole simulation as a set of collaborating agents, each of them being responsible for its own data, calculation, persistence, *etc.*

The second reason is that current technology allows us to do it. For example, the CAPE-OPEN framework provides the communication infrastructure that allows CAPE components to interoperate. Process modelling components directories, such as the one developed by the CO-LaN, give ways to locate these components on the web.

Standard agents platforms such as FIPA, and agent languages such as KQML (Section 6.1.2), can help to transform process-modelling components into autonomous communicating agents (Section 6.1.3.4). These technologies can have an important impact on the development of fault-diagnosis systems and operating support systems in general where co-operation between software components generate results faster as opposed to stand-alone applications (Section 6.1.3.1). Existing interoperation technologies have been applied to demonstrate how an agency can support the workflow during the construction of a simulation model as discussed in Section 6.1.3.3. Agent technologies combined with application-independent information representation mechanisms such as XML and EXPRESS have already been applied in modelling supply chains over the Internet (Section 6.1.3.5).

The next sections of this chapter will detail these and other technologies as well as current and future applications of multi-agents systems in CAPE.

6.1.2 COMMUNICATION ASPECTS OF MULTI-AGENT SYSTEMS

In engineering, data are used in different ways by a number of tools and people as information evolves through negotiation and trade-offs. Increasingly, engineering activities demand communication approaches in which data is shared and exchanged easily between product, process and plant life cycles, and between any of their stages. Communication is possible when data becomes information.

Data becomes information when it is given a meaning that is useful within a certain context. In other words, data must be timely, it has to get to the right person, and it has to be relevant to the user in a form that is easy to use. In multi-agent systems, agents communicate by exchanging messages over the network either synchronously or asynchronously. Messages contain knowledge and data represented in different formats and languages. Communication is kept even when one agent is off-line, as information requester either finds another agent that provides similar services or waits until the agent becomes on-line (perhaps on a different computer).

Communication in multi-agent systems is possible with the combination of a number of layers (Figure 3).

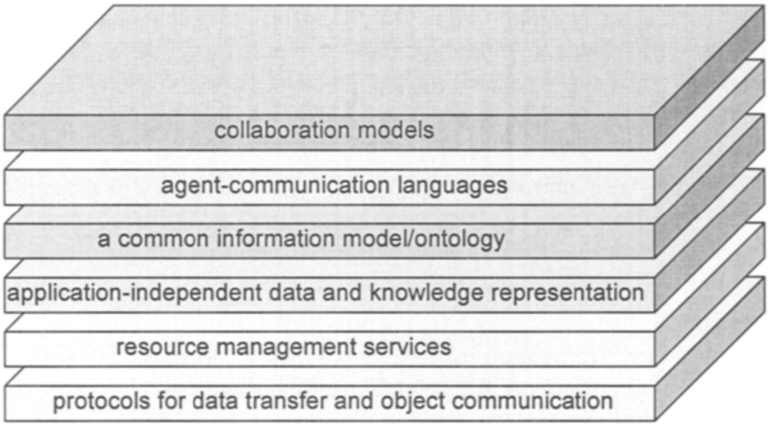

collaboration models
agent-communication languages
a common information model/ontology
application-independent data and knowledge representation
resource management services
protocols for data transfer and object communication

Figure 3. Inter-operability layers in multi-agent communication

In the lowest layer, we have common protocols for data transfer and object communication that include object interaction such as the one defined with CORBA Interface Definition Language (IDL) and COM or DCOM. In this layer of communication a software component can access the services of a remote software component by calling object methods. For example, an IDL interface allows a software component to call the methods of a remote software component [3].

CORBA and COM/DCOM alone can be used to integrate distributed components. However, using distributed object-oriented programming alone, interactions among objects remain fixed through code in each application [4].

Then we have resource management services that are the mechanisms that allow software components to find each other without knowing the exact location of the participants. For example, CORBA Object Request Broker provides this kind of services [5].

The agent name server (ANS) of some agent environments is a particular example of the resource management layer. ANS is a component that permits agents to locate and address other agents by name (rather than IP addresses and ports).

In the next layer we have approaches for application-independent data and knowledge representation, including Part 21 of the Standard for the Exchange of Product Model Data (STEP ISO-10303), the knowledge interchange format (KIF) [6] and XML [7]. STEP Part 21 defines the mechanism for exchanging instances of the data defined in an EXPRESS information model (see below). EXPRESS is a language defined in part-11 of STEP that is human and machine-readable. In STEP, data are stored in a neutral format that can be exchanged among applications that provide translation between Part 21 format and the internal data structure of the application. Likewise, knowledge interchange format (KIF) can be used to share data and knowledge. Extensible Markup Language (XML) is a standardised syntactic structure of data, so that data is exchangeable between heterogeneous software components. XML is particularly tailored to support Internet applications.

In collaborating teams and tools, interdisciplinary parties very often have diverse but overlapped views that need to be harmonised. Common information models and ontologies are specifications of a worldview that allows parties to understand the meaning of data. The lack of agreement on the meaning of data undermines consistent flow of information or knowledge.

Ongoing developments of standard information models include the ISO initiative STEP [8] [9]. Information models in STEP describe the structure of data so that plant schematic representation and simulation data can be shared and exchanged between different applications. Information models are composed of entities, attributes, and relationships among the objects.

Although similar to information models, an ontology is a formal and explicit representation of definitions of objects, concepts and the relationships that hold them [10]. The role of an ontology is to give information a precise meaning using a formal representation (often in the form of first-order logic) so as to enable consistent interpretation between humans as well as between software tools. The main difference between the two approaches is that ontologies make a commitment to an unambiguous representation of the terms of a specific domain of discourse, while information models makes a commitment to an efficient data structure. Ongoing ontology development for the process industries is reported in [11].

The components of an ontology include, classes of objects, their taxonomy, relations, and axioms. The definitions of objects can be represented using first order logic expressions in a format that is readable by both knowledge-based systems and humans. Axioms ensure that applications pass legal data for semantic consistency according to the definitions in ontologies. This capability can be exploited to express queries at a higher level of abstraction in a more convenient way than using existing text-based query mechanisms [12]. Efforts exist that extend current XML capabilities to support ontologies, including the DARPA Agent Markup Language and the Ontology Interchange Language [13].

Agent-communication languages (ACLs) provide the means by which agents represent intentions on the use of data, and find resources. With an agent communication language it is possible to convert a statement such as (open valve-1) into a question (is valve-1 open?) or into an assertion (valve-1 is open). A number of agent communication languages (ACLs) have been developed that provide such functionality. KQML (Knowledge Query and Manipulation Language), which is based on the speech act theory, is the de facto standard for agent communication languages. KQML is a protocol language developed by the ARPA supported Knowledge Sharing Effort that provides a standard interface that is independent of the implementation and platform [14]. A KQML message consists of a performative, contents and a number of parameters (keywords). An example of a KQML message is shown in Figure 4.

```
(insert
      :contents (assert (edge (node-from "pump-1")
                  (node-to "valve-1")))
      :language CLIPS
      :ontology equipment-network
      :receiver agent2
      :reply-with nil
      :sender flowsheet-agent
      :kqml-msg-id 5579+perseus+1201)
```

Figure 4. An insert performative is used to add topological information

A performative describes the intention and attitudes of an agent towards the information that is to be communicated. Some KQML performatives are listed in Table 1.

Table 1. KQML Performatives

Category	Reserved performative names
Basic informational performatives	tell, untell, deny
Basic query performatives	evaluate, reply, ask-if, ask-one, ask-all, sorry
Factual performatives	insert, uninsert, delete-one, delete-all, undelete
Multi-response query performatives	stream-about, stream-all, generator
Basic effector performatives	achieve, unachieve
Intervention performatives	next, ready, eos, standby, rest, discard

Capability definition performatives	advertise
Notification performatives	subscribe, monitor
Networking performatives	register, unregister, forward, broadcast, pipe, break
Facilitation performatives	broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all

Contents is the section of the message in which the transmitted data or knowledge is stored. The language used to represent the contents is specified using the :language parameter.

Parameters in a KQML message are used to specify information about the sender, receiver, information model or ontology, data or knowledge representation language, along with the message identifier.

Due to its extensible nature, KQML has neither fixed specifications, nor interoperable implementations, nor agreed-upon semantics. To solve these problems, FIPA (Foundation for Intelligent Physical Agents) currently is developing common specifications of an ACL with a message structure similar to KQML [15].

Figure 5 illustrates the use of messages in a brokering transaction that is initiated by

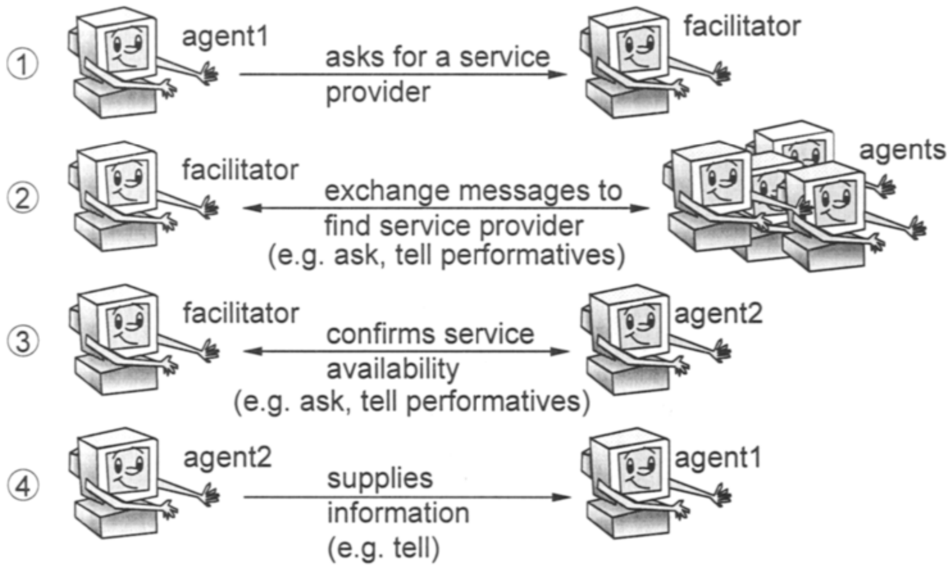

①
agent1
asks for a service provider
facilitator
②
facilitator
exchange messages to find service provider (e.g. ask, tell performatives)
agents
③
facilitator
confirms service availability (e.g. ask, tell performatives)
agent2
④
agent2
supplies information (e.g. tell)
agent1

Figure 5. Brokering (the case in which an agent wants to find a suitable service provider)

agent1 who wants to find a suitable agent for a required service. Examples of KQML performatives are shown in parentheses. **Agent1** first sends a message to a **facilitator**

agent who uses ontology, contents language, and query information along with legal and other constraints to find a suitable agent that can provide the required service. Facilitators perform tasks such as routing messages to the adequate agents, locating agents, and providing cross-platform interfaces. Once the service provider is found (**agent2**) communication is established between the service requester (**agent1**) and service provider (**agent2**). Several other communication alternatives are also possible.

Agents need a collaboration model that determines how information is to be used and produced by the involved parties. Two different levels in a collaboration model can be identified. On a higher level of abstraction there are clearly predefined and well-understood interactions. On the other hand, some information flows cannot be specified beforehand and need to be determined at the time of execution. Approaches for high-level collaboration include rules based on activity models. Activity models organise tasks to be performed and identify the information flows between activities. Not only can activity models support the construction of co-ordination rules, but activity models also define information requirements for the development of information models. For this reason, STEP standards include activity models along with information models. One example of an activity model that covers a broad number of process engineering functions is the PIBASE activity model [16].

In the lower level collaboration models, negotiation and conflict resolution approaches can be used. In [17] negotiation is defined as "the process by which a group of agents come to a mutually acceptable agreement on some matter." Agreements are convening on price, the use of resources, services to be provided, *etc*. The negotiation process may involve things such as the relaxation of constraints. Reported negotiation techniques range from game-theoretical approaches, distributed constraint satisfaction problems [18] to ant-colony methods [19] that are based on trails of pheromone.

6.1.3 DISTRIBUTED PROBLEM SOLVING

In today's environment with ever-increasing pressures from the market and tighter safety, environmental and societal constraints, process industries are being required to develop safer, and more versatile plants, that results in high-quality products in shorter time and less cost. Consequently, new engineering environments are then needed that integrate multiple aspects of the product, process and plant life-cycles.

The agent paradigm is well suited for such environments where multi-disciplinary teams and tools need to collaborate, where design constraints need to be propagated fast, where operators are to be relieved of the stress and fatigue, and where dynamic networks of companies with short-term relationships need to collaborate by integrating computational resources and people.

The following sections discuss applications of agents that aim at making all this possible.

6.1.3.1 Application of Agents in Process Monitoring/Fault diagnosis

The safety of a process plant relies heavily on the timely detection and diagnosis of abnormal situations. Despite that most of the time the control system responds to deviation of the process, humans are greatly involved in the diagnosis and decision-making when an abnormal situation occurs. A number of factors influence the performance of plant operators during these critical situations, including fatigue, and human error to name a few.

The agent paradigm fits in well for such situations. As pointed out by Brazier *et al.*, [20] agent properties such as autonomy, pro-activeness, reactiveness, and co-operation can be exploited in developing diagnosis and other kinds of support systems.

The earliest work on the development of agent systems for fault diagnosis was done as part of the ARCHON project [21]. This project produced a general-purpose architecture for multi-agent systems that was applied in the fault diagnosis of an electricity transmission system and of a particle accelerator. The first application integrated existing expert systems for monitoring the network, diagnosing faults and planning and executing repairs. Co-operation enabled the system to generate results faster as opposed to stand-alone applications. The system was also robust because the functionality of one application of domain had the backup of an application of another domain that was intelligently co-ordinated. The second application distributed and co-ordinated a large number of software applications that were connected to the accelerator. Co-operation was implemented to provide a series of interconnected trees of hypothesis of the faults generated from multiple agents each with its own domain of expertise.

In the ARCHON architecture, each agent is composed of two layers: a domain-specific layer and a generic layer. The domain-specific layer was in most cases the wrapping of an existing application. The generic layer included modules for high-level communication, planning and co-ordination, and co-operation modules. Distributed problem solving in this work was possible through the use of a *self-module model* and an *acquaintance model*. The self-model module was a model about the wrapped application that permits the agent to reason about its workload, previously executed tasks and its capabilities. The agent acquaintance model described other agents in terms of their goals, capabilities, current state of workload, etc.

Chemical plants are designed such that they can be operated for long periods of time with little human intervention. Analogously, in space exploration devices highly reliable operations are required over similar time horizons. In contrast to chemical plants, until recently spacecraft had traditionally required hundreds of ground-control operation team members who corrected malfunctions and operated the space devices remotely. One departure from that situation is NASA's remote agent, developed as part of the New Millennium Program. The program is oriented to allow a handful of human team members to command the spacecraft while reducing the cost of space-exploration missions and simultaneously increasing their number [22, 23].

Remote agent is made up of three independent modules for planning, executing operations, diagnosing faults and recovering from abnormal situations. The architecture integrates real-time monitoring and control with constraint-based

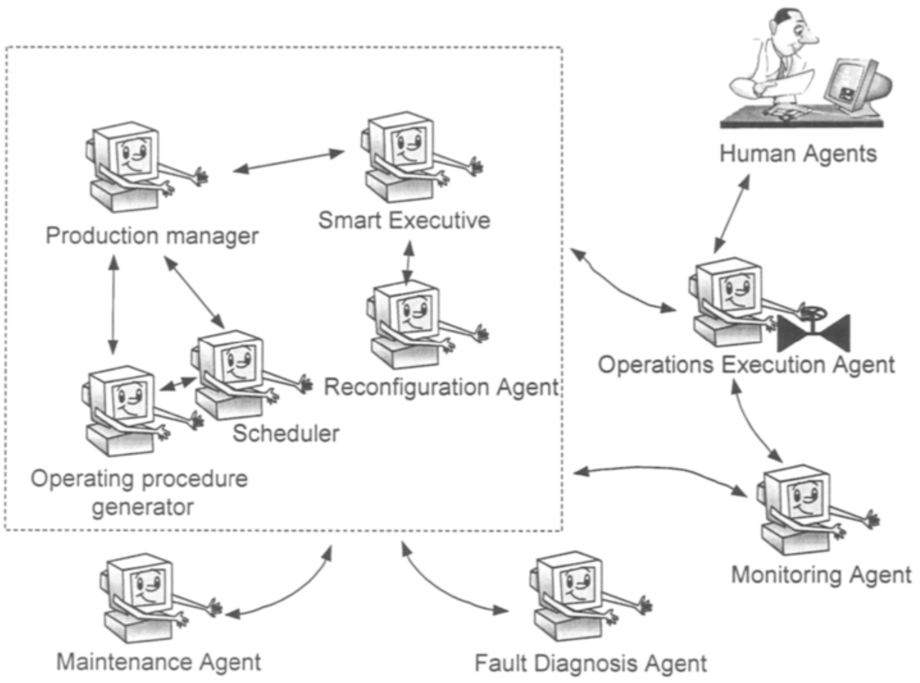


Figure 6. An agent architecture for plant operations based on NASA's remote-agent

planning and scheduling, model-based diagnosis, and reconfiguration. These are essentially, the same kind of operation support function useful in chemical plants.

Remote Agent's fault protection module, Livingston, combines the use of a declarative model of the spacecraft with deduction and search capabilities. It predicts process values using the spacecraft model and the mission information (the set points). When the current equipment configuration (the set of valves and other devices that are operated currently) fails to satisfy a goal of the mission, the agent finds an optimum cost-effective procedure to reconfigure the spacecraft and satisfy the goal. Reconfiguration is similar to changeover operations in chemical processes. An agent architecture for plant operation based on remote agent is shown in Figure6.

Another area of application of multi-agent systems is found in the integration of fault diagnosis and other operations support software. For example [24] reports the use of an agent system for the integration of fault-diagnosis and maintenance activities. These ideas can be extended further to support integration with plant design databases, corrosion simulators, and control systems.

6.1.3.2 Distributed design environments

During the design of a product, a process or a plant, a number of alternatives are proposed, evaluated and screened. Unfortunately, engineers in the design of the process and product frequently face a number of uncertain factors that affect the evaluation

[25]. In today's environment where information is becoming available faster, one can predict that engineers will collaborate with an infrastructure that allows for timely information about life-cycle considerations (such as safety, operability, and site location) to be incorporated in each stage of the design. Then methodologies and infrastructures are needed for managing design decisions as information becomes available. When this is possible, new products, processes, and plants would then be designed in a cost-effective way through collaboration over an information network such as Internet. The following example illustrates the use of such infrastructure. Let us assume that a water treatment facility is being considered for design as part of the design a plant that contractor *A* is managing. As the project evolves, *A* announces that several plant owners of the contiguous facilities have agreed to subcontract the design of a common water treatment process. Consequently, the design assignment considered by *A* becomes invalid for the group of agents (software and people) working for *A*, albeit it remains valid for the agents of the subcontractor that communicate with *A*'s agents via the Internet.

Several multi-agent systems have been proposed to support design in the mechanical, electrical and aviation domains. Some of them will be briefly discussed below.

ProcessLink is an ongoing project that aims at developing an infrastructure and methodology to facilitate the co-ordination of agents (people and software) in a distributed heterogeneous environment [26]. Engineering environments based on ProcessLink allow engineers make design decisions while a group of agents determine the validity of constraints and constraint violations. Agents communicate by exchanging KQML messages using a predefined ontology based on the design model *Redux* [27]. Combining the functions of facilitator and ANS agents, an *agent manager*, manages the inventory of available agents and their capabilities. A *project manager* agent, which is a combination of a software agent and a human agent, sets overall plans and schedules to be passed to the rest of the agents. To generate the schedule the project manager requests the agent manager to provide information about their availability and capabilities. Agents from particular engineering domains co-ordinate their decisions through a *constraint manager* and a *co-ordination agent*. The constraint manager uses constraint propagation techniques in association with the dependency-directed backtracking techniques[2] [28] from the co-ordination agent to test the consistency of a decision. Dependency-directed backtracking techniques also allow the co-ordination agent to evaluate the effects of a change in constraints of the project.

In the area of plant design, Struthers [29] proposes a generic design of autonomous agents for the modelling of design processes. Such modelling is used for organising and managing a pressure relief and blowdown study. Agents exchange messages using a predefined protocol for encoding the agent's intentions during the design process in which functionally similar agents negotiate toward the best design alternative.

Procedural control design is a major issue in process plants where operations are to be synthesised to achieve a fast, safe, and cost-effective transition from one process state to

[22] As a point of interest, Bañares-Alcántara *et al.* [28] describes the application of dependency-directed backtracking for management of change for conceptual chemical process design.

another. Agents that mimic interactive, autonomous plant objects can be used to collaboratively address this kind of design problem. In [30] a simulation-based planning method is proposed in which operations are determined by a number of agents organised hierarchically based on the concept of controlled group unit (CGU) developed by Naka [31]. Operations are organised in three hierarchical levels: plant, CGU, and equipment levels. At the plant level, operations are carried out to control the interactions with contiguous plants or battery limits. Plant agents determine overall goals that are communicated to agents in the lower hierarchies (Figure7). The plant is divided into smaller units (CGUs) that are operated by agents who act co-operatively and report to the plant agent. A CGU can be identified from the flowsheet topology as an assembly of pieces of equipment surrounded by remotely actionable-equipment (RAE), *i.e.*, equipment that can be controlled from the control room such as control valves. Each CGU agent is responsible for controlling a local inventory of material by means of sending commands to lower-level agents that operate its input and output valves as well as internal equipment items. Negotiation is used to determine the actions of contiguous CGUs. Rules for co-ordination are used that deal with decisions taken locally by CGUs about operating their valves with those taken globally by the plant.

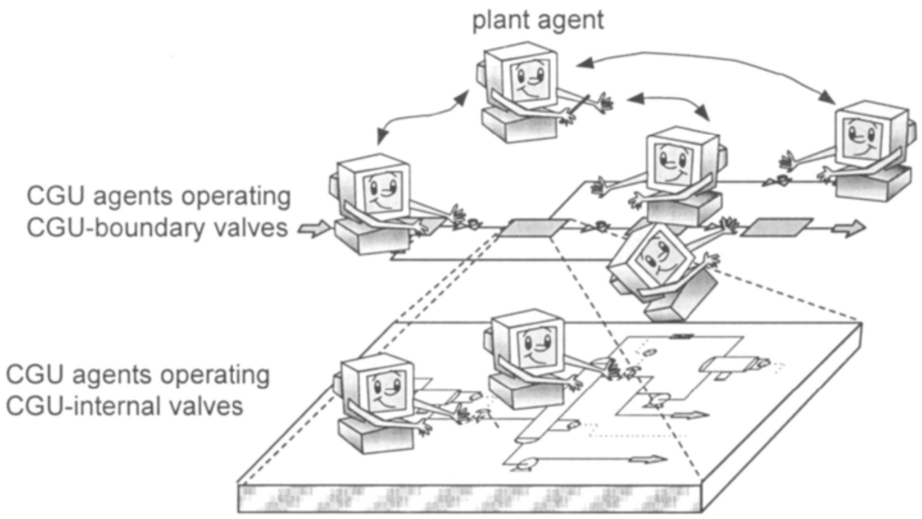


Figure 7. Plant and CGU agents

6.1.3.3 Process workflow supporting agents

Geopolitical, economic and technological forces are changing organisations and consequently affecting the way engineers work. The activities in an engineering work process may be distributed across different team members with different expertise located in different countries, in which co-operation and co-ordination among those distributed activities are frequently required. To effectively deal with the complexity of such work processes, it is desirable to employ workflow support software tools. In general, those tools (often referred to as Workflow Management Systems) help organisations specify, execute, monitor, and co-ordinate the flow of work items within a work group or organisation [32]. In this section, we will take a look at how workflow management systems work and how the agent-based paradigm can be and has been used in realising workflow systems. Afterwards, we will look at the state of the art of the work process support for process engineering including the perspective of applying the agent-based paradigm.

Generally, it is not the goal of workflow management systems to completely automate an entire work process. In most workflow systems, tasks are usually regarded as being performed externally. The workflow management system only concerns itself with management issues such as checking the availability of input and output information and resource allocation [33]. Therefore, instead of automating work processes, workflow support systems are usually used to help the users in planning their actions. However, partial automation can be achieved if an algorithm can describe some of the activities that constitute the workflow.

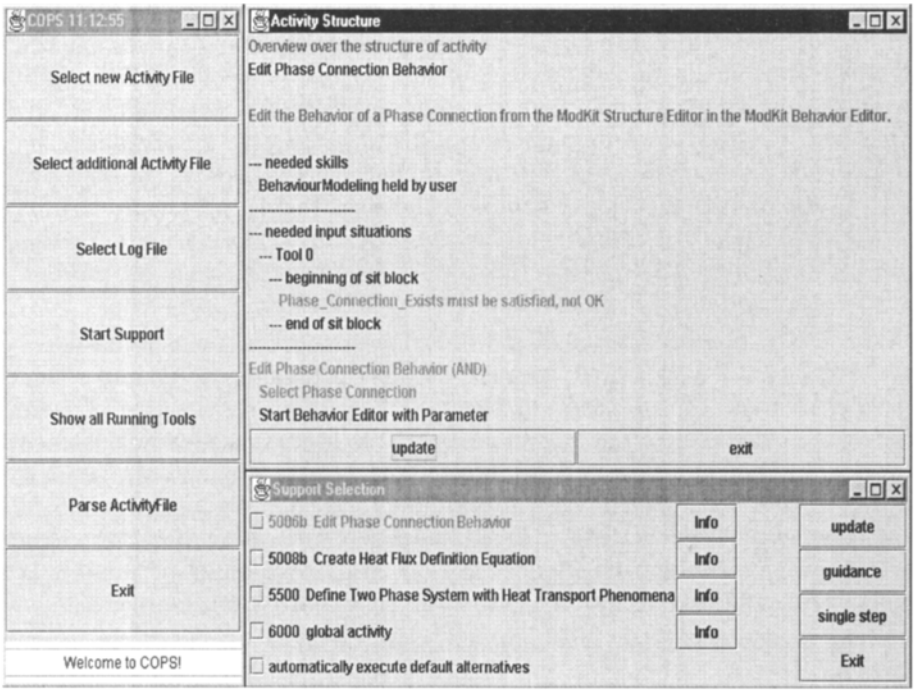
In principle, the support of activities in a work process relies on appropriate process information, which has to be available to the workflow system. For example, guiding a user through a predefined workflow requires knowledge about the work process and the activities of which it is made up, information about possible orderings of the sub-activities, evaluation procedures to find out whether an activity is executable in the given situation, and methods to change the data on which the activity operates. The latter is only relevant if automation of the activity or a whole workflow is the objective of the support. Communication platforms such as the BSCW [34], which can also be regarded as workflow support systems because they facilitate information exchange within teams independently of any prior detailed knowledge of the work process, because they just store and make available information in the form of data files.

Among the alternative approaches to realising workflow-supporting tools, multi-agent systems have been found well suited because of their autonomous, adaptive and collaborative characteristics. In this context, agents act collaboratively with the users by monitoring events and determining ways to support the users in a proactive fashion [35]. There are a number of different possible ways in which an agent can support workflow processes. For example, an agent can guide the user through predefined workflows, suggest executable activities, facilitate the collaboration between team members, or retrieve information on request. By offering these kinds of support, a process support agent can improve both the quality and the speed of work processes.

The field in which workflow support has gained the most attention is that of business processes such as banking, healthcare, and office automation. Here we often find planned, structured and shared workflows, which are good targets for workflow management systems [36]. Nevertheless, some systems have also been developed to support engineering fields. As for agent-based systems, an example for business processes management is the Agent Based Process Management System (APMS) architecture [37] from the ADEPT project [38]. In [39], agents are used in collaborative product development where they delegate work packages and activities and guide relevant data and control flows. [40] reports a software engineering environment which incorporates agent-based workflow management to actively support collaborative work in a software project.

Activities in process engineering have their particular complexities. For example, the activities involved in plant design and retrofitting can be organised in two kinds of work processes that are always combined. On a higher level of abstraction there are clearly predefined and well-understood work processes that always follow the same schematic path of execution (for example see [41]). However, as engineering activities are decomposed further, workflows may exhibit highly creative and impromptu activities that cannot be planned and represented in advance. When trying to describe parts of these work processes, like the initialisation problem for model simulation, on a detailed level, one often has to deal with creative aspects of the engineer's problem solving activities.

Until now, the work that has been done to assist in handling the complex work processes in process engineering is still quite limited. Some commercial modelling and simulation tools for chemical processes, like Aspen Plus, provide the user with automatically generated menus that adapt to the needs of the user depending on the



Overview over the structure of activity
Edit Phase Connection Behavior

Edit the Behavior of a Phase Connection from the ModKit Structure Editor in the ModKit Behavior Editor.

--- needed skills
BehaviourModeling held by user

--- needed input situations
--- Tool 0
--- beginning of sit block
Phase_Connection_Exists must be satisfied, not OK
--- end of sit block

Edit Phase Connection Behavior (AND)
Select Phase Connection
Start Behavior Editor with Parameter

☐ 5006b Edit Phase Connection Behavior
☐ 5008b Create Heat Flux Definition Equation
☐ 5500 Define Two Phase System with Heat Transport Phenomena
☐ 6000 global activity
☐ automatically execute default alternatives

Figure 8. Screendump of COPS in a snapshot when it is supporting process modelling activities

tool state and suggestions of next work items, providing a kind of simple workflow support for individual users. However, they do not provide explicit information on the underlying workflow model, and do not allow it to be changed. Besides, several process design support systems such as n-dim [42] and KBDS [43] have been developed in which the aspect of workflow management is addressed to a certain extent. However, none of these systems has explicitly incorporated a separate workflow management tool equivalent to those that have been applied in other domains but suitable to chemical process design. As an ongoing effort in process design support, an ambitious long-term research project, IMPROVE [44,45], aims at the design and implementation of a process-centred environment for chemical process design. In this environment, work process support is addressed on several levels, from co-ordination of engineering activities by the project management to the support of personal work processes. Incremental replanning of work processes, feedback situations with task re-execution and the integration of existing software tools into the work process are taken into account.

Ongoing development of an information model in the IMPROVE project represents process design processes, actors and the relations between a design process itself and the design artefacts that are obtained during the execution of the workflow process. Using this model, a context-oriented process support system (COPS) has been prototyped [46]. As an independent tool working in a process-centred software environment, COPS guides actors to select and execute a set of activities and performs documentation. It also overcomes the deficiencies existing in those commercial modelling and simulation tools mentioned above.

Figure8 shows a working screen of COPS's supporting process modelling activities. Based on these results, applying a multi-agent paradigm to develop a comprehensive workflow support system for chemical process design is currently being evaluated [47]. In particular, a software system architecture has been proposed which consists of a workflow server, a number of workflow agents and a number of personal agenda agents, as shown in Figure9. The workflow server is responsible to create workflow agents and to allocate designers of specific roles. A workflow agent initiates a design activity, determines the role of designers for this activity, then supports the execution of the activity and documents the activity. Finally, by co-operating with the workflow server and workflow agents, a personal agenda agent works for a specific designer, helping arrange the designer's agenda in the context of the entire design team or project.

In this paradigm, agents can potentially be employed not only in fulfilling the typical workflow management functionality, i.e. managing design activities, but also in other aspects closely related to the definition and execution of activities, such as assisting the designer by showing relevant design artefact information according to the definition and status of an activity.

In summary, we have seen how complex process engineering activities can benefit from workflow supporting agents. Although workflow supporting tools (agent-based tools in particular) have not been widely applied in process engineering, the experience from other domains as well as the current emerging trend in these evolving technologies of agent-based workflow support are showing an encouraging prospective of enabling efficient co-ordination and management of process engineering activities.

6.1.3.4 Distributed simulation

Multi-agent based computation infrastructure for simulation are already finding increasing scope of application in varied domains like social, economic, biological or ecological systems [48]. The autonomous nature of individual agents and their capability of intelligent interactions make them able to represent system modules, which act autonomously in the real world. Thus, simulation of discrete event systems involving naturally intelligent entities witnessed the initial applications of agent-based

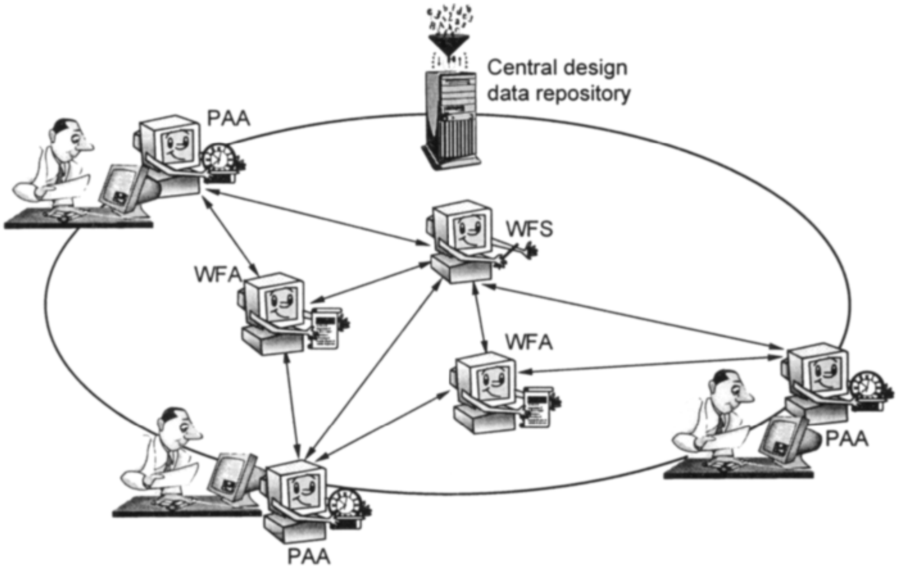


Figure 9. Agent-based workflow management system for process design. WFS -Workflow Server WFA - Workflow Agent PAA - Personal Agenda Agent.

computation. In the case of simulation of continuous processes, though a limited number of research and implementations are reported, the possible application scopes of agent-based computations are yet to be fully explored. One of the efforts in agent-based simulation of continuous phenomena in distributed parameter composite physical systems can be found in [49, 50]. Agent applications specifically oriented towards distributed simulation of chemical process plants are yet to be widely available. In this section we will discuss some of the envisaged benefits of agent based distributed simulation of chemical process plants.

As the demand of powerful simulation approaches to tackle more complex problems increases, a single desktop computation capability is no longer a practical option for problems such as real plant simulation. To circumvent the resulting problem of large initial investment on computation, one research direction is to use distributed computation environments where a cluster of multiple low cost computers are to share the computational burden. Present availability of agent based software communication protocols and the addition of knowledge-based problem solving capabilities potentially offers fast, flexible yet cost effective simulation environment possibilities.

Chemical process plants typically represent a hybrid system, which manifest both continuous dynamics as well as discrete event phenomena. While functions such as planning and executing operations can be assigned to a number of agents, the decomposition of the continuous process dynamics need careful analysis of the system. Some preliminary research in this direction that is targeted to process plant simulation is reported in [51].

The major anticipated roles of agents in an agent based process simulation environment are likely to include: behaviour model decomposition, processing resource allocation, local solver execution, thermodynamic property computation, and distributed computation co-ordination. One or multiple agents may be necessary for each of these activities that implement the desired functions of a distributed simulation environment. During a simulation session each computation module of the behaviour model may be assigned to dedicated agents. Each simulation session could be co-ordinated by one computation co-ordination agent. A typical distributed multi-agent based process simulation environment is shown in Figure10. A scenario of agent creation and inter-agent communication may proceed in the following way.

On initiation of a new simulation session by any external agent (human or software), the behaviour model management agent (referred below as model agent) is instantiated and associated with the desired behaviour model database, if no such instance is already running. The model agent analyses the behaviour model and demarcates multiple computation modules. A computation co-ordination agent is subsequently initiated by the model agent with the information of the number of computation modules. In the case of the existence of a model agent instance, the new simulation request is registered to the existing agent and a new co-ordination agent is instantiated. The co-ordinator agent creates instances of the required number of solver agents and assigns appropriate processor to the solver agents so that the estimated discrepancy in time per computation cycle among the processes is minimum. To support the computation, the solver agents need access to the thermodynamic property computation utility, which may be made available as individual utilities or as a multi-thread server with agent interface. The solver agents are to collect their respective computation model from the model agent and with the help of the co-ordinator agent initialise the inter-

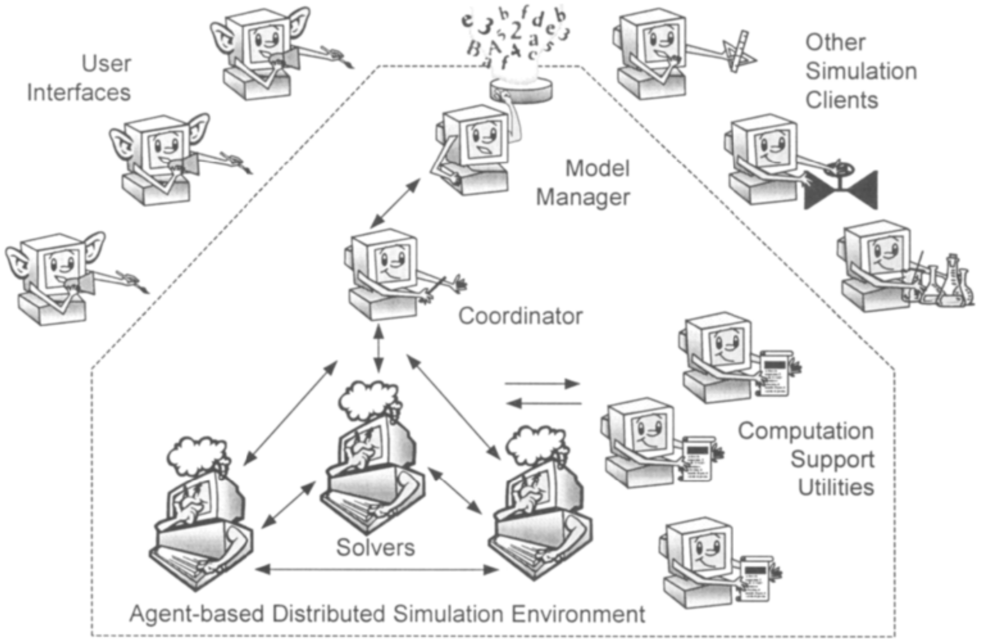

User Interfaces
Other Simulation Clients
Model Manager
Coordinator
Computation Support Utilities
Solvers
Agent-based Distributed Simulation Environment

Figure10. Schematic view of a typical multi-agent based distributed simulation environment and client utilities

module communication ports for information exchange about computation time. During execution on completion of each global time step the solver agents are to exchange relevant variable values with the appropriate solver agents as well as with the co-ordinator agent. On confirmation of convergence, the co-ordinator agent for the next time step would issue the computation command. In the case of perceived mismatches in the values of the variables, the co-ordinator agent requests a re-computation of the last time step with new global time step.

As shown schematically in Figure10, an integrated agent oriented process design environment is likely to employ many other agents or groups of agents to play specialised roles. Some of these activities are product design, process design, plant design, behaviour model editing, operation procedure synthesis, etc. Agents responsible for these activities are likely to interact with one or more of the above described simulation support agents. The interaction could be for executing a new simulation, analysis and interpretation of a process behaviour or to register changes to the ongoing simulation sessions. As the need for concurrent engineering environments are being appreciated, the requirements for intelligent inter-utility interactions are becoming essential. These requirements also gave rise to efforts for developing proper protocols of information exchange across specialised knowledge domains. Several initiatives, as discussed earlier in this chapter (Section 6.1.2), for defining meta-data format standards and domain specific ontologies are making it possible to have meaningful inter-agent communications across the context boundary of individual agents.

The major benefits, which are envisaged to be available in agent based distributed simulation environments, can be summarised as follows:

- Increased speed of computation through concurrent utilisation of distributed processing power
- Scalability of computational complexity as well as computation environment features by gradual incorporation of more processing power or utilities
- Convenient integration of third party utilities and customised selection of appropriate tools for any particular requirement
- Convenience of maintenance and management of software components with possibility of modular upgrade of the simulation environment
- Flexible multi-user support for collaborative evaluation, decision making or training in a distributed environment
- Possibility of intelligent interfaces that adapt to human users as well as to other software components

Possibility of fast and flexible connectivity of software components and users over Internet offers promising directions of distributed computation. Agent based world-wide integration and utilisation of idle processing power, simulation supporting utilities as well as simulation results would aid distributed monitoring, analysis and decision making regarding simulated as well as real online processes. This will also facilitate different design and development activities to be performed concurrently without the constraint of physical location of the involved personnel in a cost effective and flexible way. Some implementation efforts in this direction are already being reported [52].

High-level modelling languages allow users to describe process simulation problems at various levels of abstraction composed of several modelling components. Putting this together, it is now possible to develop a system in which a user would only specify a process engineering problem using the high-level modelling language, and communicate the problem to a component catalogue which would in turn communicate with process modelling agents and let them build a solution collectively. Such a multi-agent framework for process modelling is what can be called CO-GENTS in the CAPE-OPEN project. The CO-GENTS framework is currently being investigated by a group of universities and companies in order to demonstrate the feasibility of the approach.

6.1.3.5 Supply-Chain Management

Supply chains are distributed, integrated processes where the participating parties work together to acquire raw materials, convert them into other products and deliver the finished products to retailers and customers. Two basic processes dominate the dynamics of the supply chain, namely production planning-inventory control, and logistics-distribution. Along the supply chain, parties experience a forward flow of materials, a backward flow of capital, and a bi-directional flow of information. The degree of integration between the participating components depends on the ease with which these flows occur along the chain. It is worth mentioning here that the term *supply chain* is used as a convenient simplification; as each link on the chain actually branches in its own and different suppliers and retailers, a more realistic term would be *supply network*. In the same fashion, some authors consider the term *demand chain* to be more accurate, since the chain begins with a consumer, who creates demand for products, and ends with the raw materials suppliers [53].

From this definition of supply chain we can consider the finished product to be anything from the manufactured items that go to the final customers, the process plant units, control systems, computer hardware, or whatever that has a market niche. From here on, however, we focus our discussion on two of these products, namely the process plant and its manufactured products. These two are the axles that spawn the need for data modelling.

On one side, the complexity of supply chains whose final products are process plants and its relation with the plant life cycle has yielded plenty of studies. Yang and McGreavy [54] identify the requirements for the process data to be used over the plant life cycle under the EPISTLE (European Process Industries STEP[3] Technical Liaison Executive) framework. The need to share data, codes and concepts is stressed. Functional characteristics of a computer integrated CE environment useful along the whole plant life cycle is discussed in [55], stressing the chemical industries' special needs. A project management system that is able to handle different plant life cycle activity models within a concurrent engineering framework is described in [56]. However, the chemical process industries are far from the integration that other productive activities such as aeronautics, automotive and electronics have achieved in aspects like e-commerce and data modelling.

[3] The STEP standard discussed in Section 6.1.2

On the other hand, the supply chains of chemical products have some distinctive features. Firstly, in the market of chemicals, the best customers are usually other chemical companies. Secondly, the final customer is usually farther in the chain and therefore it is harder to *pull* on an extreme to favour integration. In addition, the final product and by-products co-exist in the market. Furthermore, compared to other industrial sectors, the life cycles of plants, products and processes are longer and more complex. Finally, the process is subject to tighter safety and environmental constraints than in other industries [57]. The product life cycle is therefore very different to other manufactured products. Figure11 shows the relationship between the plant life-cycle and the plant[4] and product supply chains[5].

Compared to the many efforts on methods based on control theory and optimisation (ranging from operations research models of logistics/production and statistical analysis to business games), the work on agent-based systems is still very limited, and little has been published about specific applications in the process engineering domain. Agent-based systems have a number of distinctive features that make them attractive for supporting the supply chain. For example, the decentralised interoperability of agents

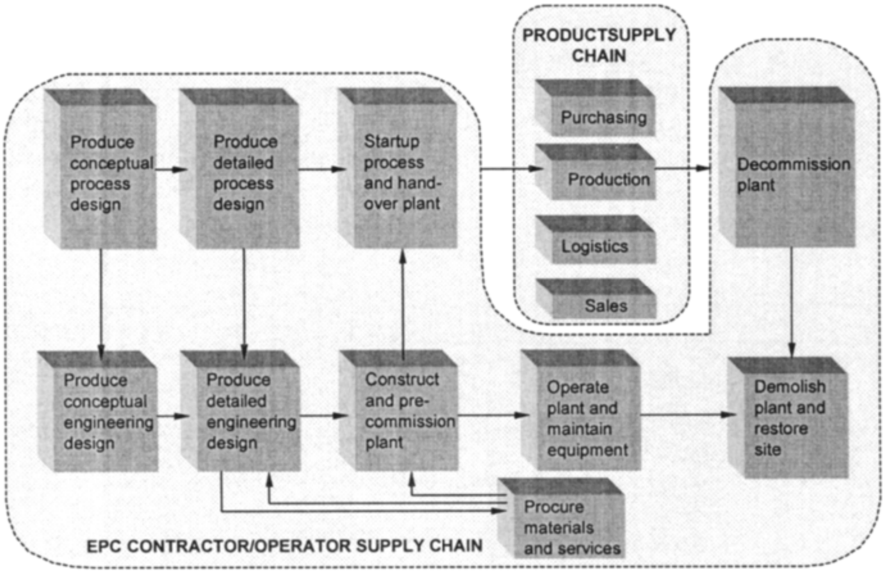

PRODUCTSUPPLY CHAIN
Purchasing
Production
Logistics
Sales
Produce conceptual process design
Produce detailed process design
Startup process and hand-over plant
Decommission plant
Produce conceptual engineering design
Produce detailed engineering design
Construct and pre-commission plant
Operate plant and maintain equipment
Demolish plant and restore site
Procure materials and services
EPC CONTRACTOR/OPERATOR SUPPLY CHAIN

Figure 11. Relationship between the plant's life cycle and the plant and product supply chains

makes them well suited for designing distributed supply chain systems. In addition, the modular, collaborative and autonomous characteristics of agents allow them to mimic the supply chain structure, *i.e.*, systems for individual chain components can be developed and maintained independently.

[4] The plant supply chain involves the flow of information, material and energy between the engineering, procurement and construction service providers (EPC contractor), plant owners, equipment suppliers, control vendors, and process licensors.

[5] Extended supply chains may include recycling facilities, consumers, collectors of recycling materials, etc.

Ongoing work at the University of Leeds aims at developing a multi-agent support system that is intended to address the above challenges. The system is a multi-agent based environment that models the geographically distributed retailers, logistics, warehouses, plants and raw material suppliers as an open and re-configurable network of co-operative agents, each performing one or more supply chain functions. The environment operates at two levels of a decision making process. At the top level, an agent based co-operation mechanism allows compromise decisions to be made through negotiation. When there are conflicts, the next level (the dynamic simulation of chain behaviour) allows optimal compromises through the simulation of the effect of the decisions.

The model is coherent inasmuch as it abets component integration at all the different levels of implementation [58, 59], as shown in Figure12 . Agents are programmed with the Java language so they can run on any platform. The agent instances are shown in Figure *1*13.The use of Java allows the system to be easily adapted to run on the Internet. KQML is used as common language for inter-agent communication. The STEP approach (section 6.1.2) is used to give format to the contents of the messages exchanged between the agents.

The implementation focused on a case study where a multi-purpose batch plant that produces wood paint and coatings schedules its production in order to fulfil warehouses and customers demand. The models deal with inventory levels, optimisation, production scheduling, demand forecasting, transportation logistics and information exchange in order to study the effect of the chain dynamics over the plant's variables, wearing away and idle times during campaigns along a given horizon.

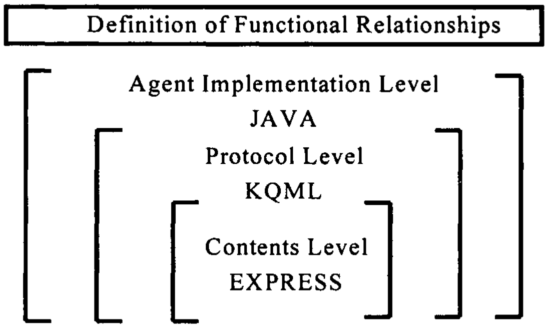

Definition of Functional Relationships
Agent Implementation Level
JAVA
Protocol Level
KQML
Contents Level
EXPRESS

Figure 12. Levels of implementation.

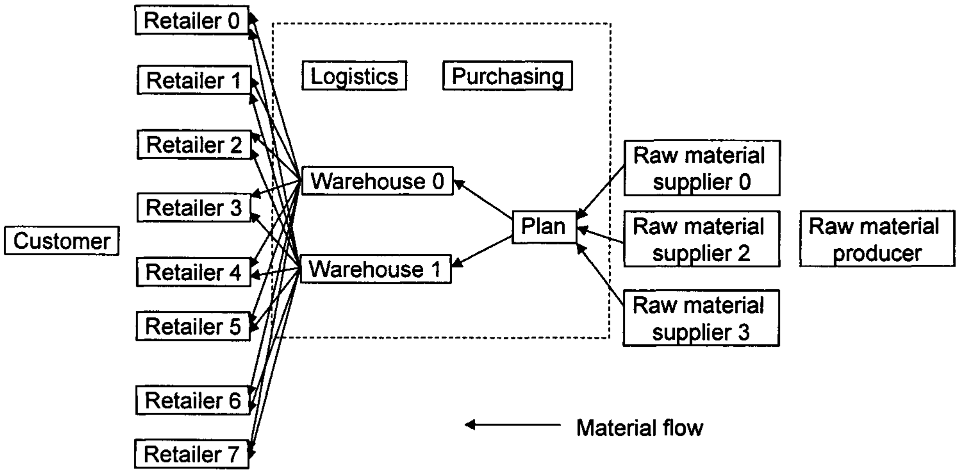


Figure 13. Agent instances in supply chain model.

Figure14 shows some interesting results of a step change in demand that can give insight into the use regime of batch plants regime and their long term scheduling. The plant produces three finished products and has ten processing units.

Multi-agent systems will be deployed over the Internet to act simultaneously as real-time knowledge management tools, corporate decision support tools, simulation test-beds for what-if scenarios and as means for increasing the supply chains integration. This paradigm is envisioned as fundamentally changing the way that partners interact in the supply chain.

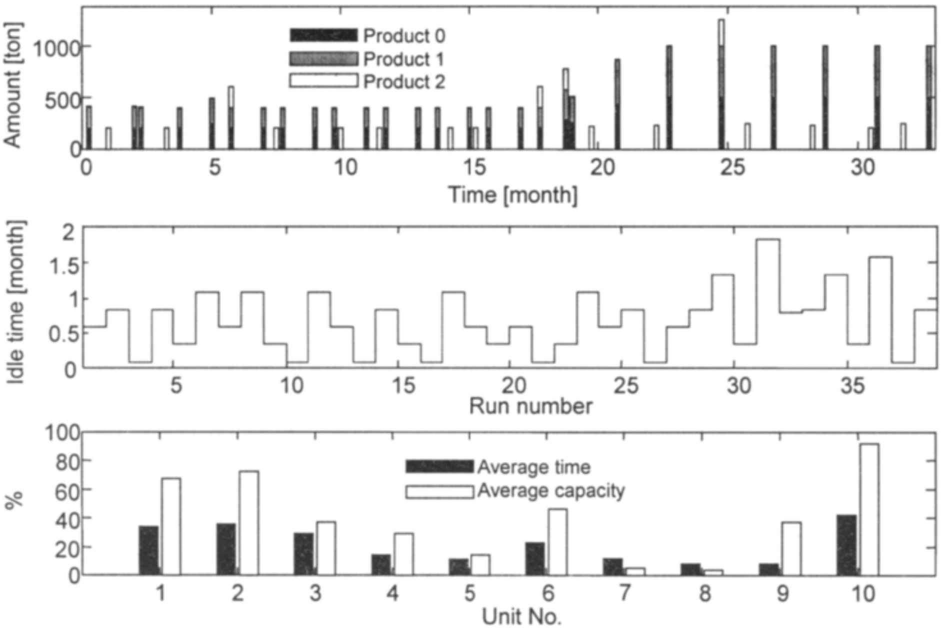


Figure 14. Effect of the chain dynamics over the amount produced, plant's idle time and used equipment capacity.

6.1.4 CONCLUSIONS

The previous sections have shown representative examples of multi-agent systems for application in CAPE. Looking again at the evolution of software artefacts, as presented in the introduction and in chapter 4.1 of this book, we can now predict that agent technology will allow CAPE tools to reach the third stage of evolution, that is, the one of dynamic adaptive components (see Figure 15). In order to reach this stage, process modelling agents will need to be enhanced with dynamic capabilities, such as automatic reconfiguration, learning from examples, from past applications, or from interactions with users. An auto-adaptive numerical solver, for example, would not only offer its services to other modelling agents, but would also be able to adjust convergence criteria based on the solutions of previous problems, or on memories of interactions with the user, remembering preferences and thus offering better default values for its parameters each time it is being invoked.

Since this chapter is about future trends, let's ask: what next? Well, look at the trends in information technologies, such as described in the European Commission's Information Society Technology FET (Future and Emerging Technologies) actions [60], or in the US Information Technologies for the Twenty-First Century (IT2) programme.

You will *see* that computers, ambient intelligence, pervasive computing, million-neurone networks, and computational grids disappear. In other words, clues for the next generation of software objects and agents that was implicitly predicted by Genrich Altshuller in TRIZ with his "patterns of evolution". Time will tell if these futuristic considerations make sense, but remember that CAPE and software were only invented less than fifty years ago, therefore there will obviously be much more to come.

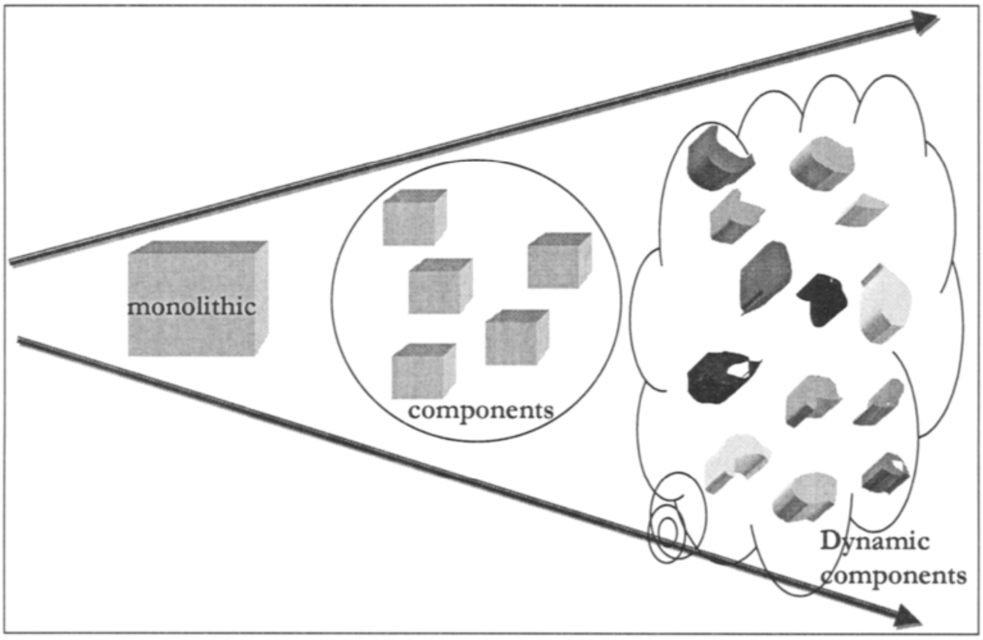


Figure 15. The third stage of evolution: dynamic components

Another crucial aspect is whether there can exist any business model capable of coping with new technologies such as the ones presented in this section and in other parts of the book. This will be the subject of chapter 6.5.

6.1.5 REFERENCES

1 A. Drogul, "Applications des SMA réactifs: simulation et aide à la conception," Tunisian Schoool on AI, May (2001) [On-line] Available: http://www-poleia.lip6.fr/~drogul/
2 MIT Encyclopedia of Cognitive Sciences [On-line] Available: http://cognet.mit.edu/MITECS
3 OMG, "CORBA 2.4 specification," [On-line] Available: http://www.omg.org/ (2000)

4 D. L. Martin, A. J. Cheyer, and D. B. Moran, "The open agent architecture: A framework for building distributed software components," *Applied Artificial Intelligence,* Vol. 13, No. 1, pp. 91-128 (1999)

5 Orbix Programmer's Guide (1997)

6 M. R. Genesereth and R. E. Fikes, "Knowledge Interchange Format Version 3.0 Reference Manual," Technical Report Logic-92-1. Stanford University Logic Group, [Online] Available: http://logic.stanford.edu/papers/kif.ps (1992)

7 W3C, "Extensible Markup Language (XML) 1.0 (Second Edition)," [On-line] Available: http://www.w3.org/XML/ (2000)

8 J. Fowler, "STEP for data management exchange and sharing," Technology Appraisals, ISBN 1-871802-36-9 (1995)

9 NIST, "The STEP Project," Available on the World Wide Web from: http://www.nist.gov/sc4/www/stepdocs.htm (1999).

10 T. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, Vol. 5, No. 2 (1993)

11 R. Batres and Y. Naka "Process Plant Ontologies based on a Multi-Dimensional Framework," In Foundations of Computer Aided Process Design, AICHE Symposium Series, No. 323, ISBN No: 0-8169-0826-5 (2000)

12 E. Mena, V. Kashyap, A. Illarramendi, A. Sheth, "Estimating Information Loss for Multi-ontology Based Query Processing," *Second International and Interdisciplinary Workshop, Intelligent Information Integration*, Aug. 24 –25, Brighton Centre, Brighton, UK (1998)

13 J. Hendler, D. L. McGuinness, "The DARPA Agent Markup Language," *IEEE Intelligent Systems*, Vol. 15, No. 6, pp. 67-73 (2000)

14 T. Finin, J. Weber, G. Wiederhold, M. Genesereth, D. McKay, R. Fritzson, S. Shapiro, R. Pelavin, and J. McGuire, "Specification of the KQML Agent-Communication Language" (1994) [On-line] Available: http:// logic.stanford.edu/ sharing/papers/kqml.ps

15 J. Dale and E. Mamdani, "Open Standards for Interoperating Agent-Based Systems," *Software Focus*, To be appear, (2001) [On-line] Available: http://www.fipa.org/docs/input/f-in-00023/f-in-00023.doc

16 Process Industry Executive for achieving Business Advantage using Standards for data Exchange, "PIEBASE Activity Model," [On-line] Available: http://cic.nist.gov/piebase/ppam20.pdf

17 N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra and M. Wooldridge "Automated negotiation: prospects, methods and challenges," *Int. J. of Group Decision and Negotiation*, Vol. 10, No. 2, pp. 199-215 (2001)

18 S. E. Conry, K. Kuwabara, V. R. Lesser, and R. A. Meyer, "Multistage negotiation for distributed constraint satisfaction *IEEE Transactions on Systems, Man, and Cybernetics,* 6, pp. 1462-1477, Nov./Dec. (1991)

19 H.V.D. Parunak and S. Brueckner, "Ant-Like Missionaries and Cannibals: Synthetic Pheromones for Distributed Motion Control," Fourth International Conference on Autonomous Agents, Barcelona, Spain. June (2000)

20 F.M.T. Brazier, C.M. Jonker, J. Treur, and N.J.E Wijngaards, "On the Use of Shared Task Models in Knowledge Acquisition, Strategic User Interaction and Clarification Agents," *International Journal of Human-Computer Studies,* Vol. 52, No. 1, pp. 77-110 (2000)

21 N. R. Jennings, E. H. Mamdani, I. Laresgoiti, J. Perez, and J. Corera, "Using ARCHON to develop real-world DAI applications," *IEEE Expert* **11**, No. 6, pp. 64-70 (1996)

22 S. Hedberg, "Executive Insight: AI Coming of Age: NASA Uses AI for Autonomous Space Exploration," IEEE Expert 12, No. 3, pp. 13-15 (1997)

23 N. Muscettola, P. Nayak, B. Pell, and B. Williams, "Remote Agent: To Boldly Go Where No AI System Has Gone Before," *Artificial Intelligence* 103, No. 1-2, pp. 5-48, (1998)

24 S. Nagano, T. Kawamura, T. Hasegawa, A. Ohsuga, "A Plant Diagnosis and Maintenance System Coordinated by Bee-gent," Proceedings of the 60th Annual Convention on Information Processing Technology, Tokyo, Japan (In Japanese) (2000)

25 P. M. Herder, M. P. C. Weijnen, "A concurrent engineering approach to chemical process design," *Int. J. Production Economics,* Vol. 64, pp. 311-318 (2000)

26 C. Petrie, "ProcessLink Coordination of Distributed Engineering," Available, [WWW] http://www-cdr.stanford.edu/ProcessLink/papers/osaka/J-PLink.ps (1997)

27 C. Petrie, "The Redux' Server," *Proc. International Conference on Intelligent and Cooperative Information Systems (ICICIS),* Rotterdam, May (1993)

28 R. Bañares-Alcántara, J. M.P. King and G. H. Ballinger, "Égide: A Design Support System for Conceptual Chemical Process Design," in AI System Support for Conceptual Design, Springer-Verlag, Edited by John E.E. Sharpe (1995)

29 A. Struthers, "Use of Intelligent Software Agents in an Industrial Pressure Relief and Blowdown Design Tool," Comp. Chem. Eng. Vol. 21, Suppl., pp. S83-88 (1997)

30 R. Batres, S. P. Asprey, T. Fuchino and Y. Naka, "A KQML Multi-Agent Environment for Concurrent Process Engineering," Computers & Chemical Engineering, Vol. 23, Supplement, pp. S653-656 (1999)

31 Y. Naka, "Operational Design,"Process System Engineering Laboratory, Internal Report (in Japanese), Tokyo Institute of Technology (1989)

32 C.A. Ellis, J. Wainer, "Goal-based models of collaboration," Collaborative Computing, Chapman& Hall, March, 1-1:61-86 (1994)

33 W. M. P. van der Aalst, K. M. van Hee and G. J. Houben, "Modeling Workflow Management Systems with high-level Petri Nets," Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related Formalisms, pp. 31-50, (1994)

34 W. Appelt, "WWW Based Collaboration with the BSCW System," in Proceedings of SOFSEM'99, Springer Lecture Notes in Computer Science 1725, pp.66-78 (1999)

35 K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng, "Distributed Intelligent Agents," IEEE Expert, Vol. 11, No. 6, 1996

36 W. Schulze, "Workflow-Management für CORBA-basierte Anwendungen. Springer," Berlin (2000)

37 P. D. O'Brien and W. E. Wiegand, "Agent Based Process Management: Applying Intelligent Agents to Workflow," The Knowledge Engineering Review, Vol. 13, No. 2 (1998)

38 N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, M. E. Wiegand, C. Voudouris, J. L. Alty, T. Miah, and E. H. Mamdani, "ADEPT: Managing Business Processes using Intelligent Agent," Proceedings of BCS Expert Systems Conference (ISIP Track), (1996)

39 G. Q. Huang and K. L. Mak, "Agent-based workflow management in collaborative product development on the Internet," Computer-Aided Design, 32:133–144 (2000)

40 J. W. Chang and C. T. Scott, "Agent-based workflow: TRP support environment (TSE)," Computer Networks and ISDN Systems, 28: 1501-1511 (1996)

41 J. M. Douglas, "Conceptual Design of Chemical Processes," McGraw-Hill, New York (1988)

42 A. W. Westerberg, E. Subrahmanian, Y. Reich, S. Konda and the n-dim group, "Designing the Process Design Process," Presented on PSE'97 (1997)

43 R. Bañares-Alcántara and H. M. S. Lababidi, "Design Support Systems for Process Engineering--II. KBDS: An Experimental Prototype," Computers chem. Engng., Vol. 19, No. 3, pp. 279-301, (1995)

44 M. Jarke, T. List, and K. Weidenhaupt, "A Process-Integrated Conceptual Design Environment for Chemical Engineering," 18th International Conference on Conceptual Modeling, Paris, France, (1999)

45 M. Nagl, B. Westfechtel (eds.), "Integration von Entwicklungssystemen in Ingenieuranwendungen," Springer, Berlin (1999)

46 M. Eggersmann, C. Krobb, and W. Marquardt, "A Modeling Language for Design Processes in Chemical Engineering," Proceedings of the 19th International Conference on Conceptual Modeling, Lecture Notes in Computer Science 1920, Springer, pp. 369-382, (2000)

47 A. D. Yang, "Workflow Management in Process Design With Software Agents," Internal report, LPT, RWTH Aachen, Jan. 2000

48 N. R., Jennings, K. Sycara and M. Wooldridge, "A Roadmap of Agent Research and Development," *International Journal of Autonomous Agents and Mutli-Agent Systems*, Vol. 1, No. 1, pp.7-38, (1998)

49 T. Drashansky, E. N. Houstis, N. Ramakrishnan and J. R. Rice, "Networked Agents for Scientific Computing,"*Communications of ACM,* Vol. 42, No. 3, pp.48-54 (1999)

50 L. Bölöni, D. C. Marinescu, J. R. Rice, P. Tsompanopoulou and E. A. Vavalis, "Agent based Networks for Scientific Simulation and Modeling," *Concurrency: Practice and Experience*, Vol. 12, No. 9, pp. 845-861 (2000)

51 R., Chatterjee, "Towards An Integrated Simulation Based Process Engineering Environment," *Proceedings of JSPS International Workshop on Safety-Assured Operation and Concurrent Engineering,* December 3-5, Yokohama, Japan (2000)

52 Y. Zeng, S. M. Jang and C. C. Weng, "Consider an Internet-Based Process Simulation System," Chemical Engineering Progress, p.53-60, July 2000

53 L.F. Dugal, M.S. Healy, S. Tarkenton, "Supply chain management: a challenge to change," Coopers & Librand, L.L.P., Boston, MA (1994)

54 X. Yang, C. McGreavy, "Requirements for sharing process data in the life cycle of process plants,"Comp. Chem. Eng., 20, pp. S363-S368 (1996)

55 C. McGreavy, X.Z. Wang, M. L. Lu, Y. Naka, "A concurrent engineering environment for chemical manufacture," Concurrent Engineering: Research and Applications, 3, pp. 281-292, (1995)

56 J.W. Hawtin, P.W.H. Chung, "Concurrent engineering system for supporting STEP based activity model," Comp. Chem. Eng., 22, pp. S781-S784 (1998)

57 C. E. Bodington and D.E. Shobrys, "Optimise the supply chain," In Advanced process control and information systems for the process industries, Kane, L.A. (eds.), Gulf Publishing Co., Houston, pp. 236-240 (1999)

58 R. Garcia-Flores, X. Z. Wang, and G. E. Goltz, "Agent-based information flow for process industries' supply chain modelling," *Comp. Chem. Eng.*, 24, pp. 1135-1141 (2000)

59 R. Garcia-Flores, X. Z. Wang, "Multi-agent based chemical supply chain management and simulation," to appear (2001)

60 European Commission's Information Society Technology FET (Future and Emerging Technologies) actions. [On-line] Available: www.cordis.lu/IST/FET

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 6.2: Tools Integration for Computer Aided Process Engineering Applications

E. S. Fraga, R. Gani, J. W. Ponton & R. Andrews

6.2.1 INTRODUCTION

Computer tools for process engineering cover a wide range of applications. These range from the mundane, such as office productivity tools used to prepare reports, through to highly specific tools for process engineering. As an illustration, Fig. 1 shows the tasks involved in reactor modelling and simulation. Even a small part of the whole design of a process is itself made up of a large number of sub-tasks with complex inter-relationships. For design as a whole, the tasks range from model building, the generation of process flowsheets, heat exchanger network synthesis and rigorous simulation of process flowsheets for detailed analysis. Furthermore, identifying the best process design requires considering a variety of issues. These include economics, operability, safety and maintenance, along with the impact of a design on the environment, including issues related to society. The number of distinct applications used is therefore large and the process of design requires managing the flow of information from one application to another.

There are two approaches for handling this diversity of tasks and requirements. The first is to use a specific tool for each task while the second is to use an application, which attempts to cover the full range of requirements. The former has the advantage of allowing the use of the *best of breed* tool for each specific aspect. However, integrating the different tools used may be an obstacle, which is difficult to overcome. The data generated by one tool may be required as input in another tool. For example, the state of a process flowsheet may be required to define the heat exchanger network synthesis problem to be solved using the pinch method or the results of a detailed simulation of a reactor may be required by the engineers responsible for the design of the separation section.

In many cases, this transfer of data from one tool to another is carried out by hand. Although effective and reasonable in some cases, in general this limits the efficiency and accuracy with which the necessary calculations may be executed. The effect on efficiency is simply due to the time required to translate, by hand, the output of one program into the form required by the second program. More importantly, however, the manual operation is likely to suffer from transcription errors, especially as the amount of data transferred increases. Automated interaction between the diverse tools used in process engineering is therefore necessary for the effective use of all the computer aided tools available to the engineer.

A fully integrated approach eliminates the problem of data interchange. There is only one tool and no manual intervention for data transfer is necessary. Another advantage of this approach is that the engineer need only learn one interface, minimizing the human errors inherent in using complex computer tools. However, defining and implementing a computer based framework, which can be extended for use throughout the design process is also challenging.

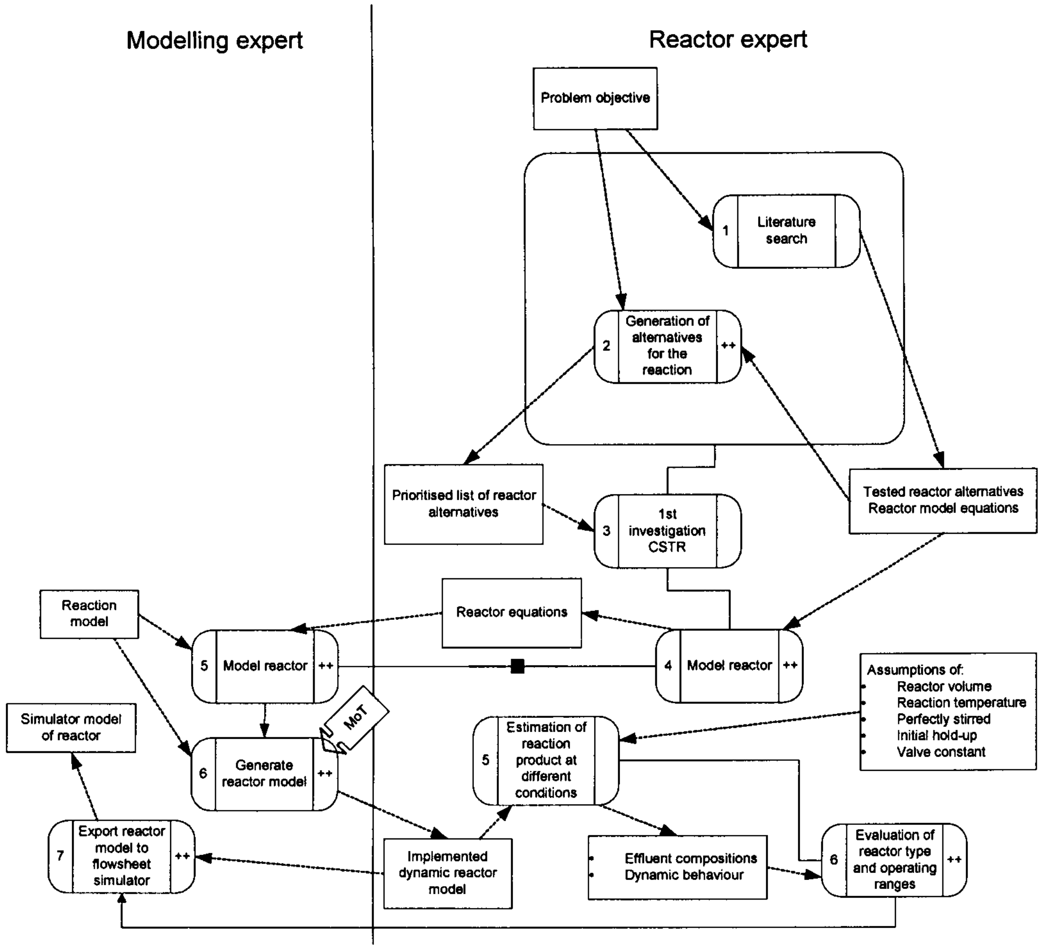

Modelling expert
Reactor expert
Problem objective
1 Literature search
2 Generation of alternatives for the reaction ++
Prioritised list of reactor alternatives
3 1st investigation CSTR
Tested reactor alternatives Reactor model equations
Reaction model
Reactor equations
5 Model reactor ++
4 Model reactor ++
Simulator model of reactor
MoT
6 Generate reactor model ++
5 Estimation of reaction product at different conditions
Assumptions of: Reactor volume, Reaction temperature, Perfectly stirred, Initial hold-up, Valve constant
7 Export reactor model to flowsheet simulator ++
Implemented dynamic reactor model
Effluent compositions, Dynamic behaviour
6 Evaluation of reactor type and operating ranges ++

Figure 1. Data flow, work flow and needed tools in reactor modelling & simulation

Although the two approaches described above are orthogonal, they share the underlying problem of data representation. From the user's point of view, it is likely that both approaches will appear the same (assuming that the user interfaces for the tools in an integrated environment share a common look and feel). Furthermore, in practice, it is likely that a combination of both approaches will be required. Fully integrated tools will be unlikely to cover the whole spectrum of tasks required in design and so even these tools will need to integrate with other tools. In any case, application integration is not simply a matter of transferring data from one tool to another, although this step is necessary. Issues of data interpretation, data management, concurrency in the design process, ease of use, and the generation of audit trails are equally important.

The overall aim of tool integration is to aid the engineer in the process of design, helping meet the increasing demands on process simulation, analysis, optimisation and control. By providing an easy to use environment for design, the engineer is able to consider more issues simultaneously, leading to a better final design. More informed decisions can be made early in the design process and there is therefore a reduction in the iterative procedure that often appears in design where later analysis leads one to throw away the current design and start over. An integrated environment which provides a common interface also has the advantage of minimizing human errors, errors often induced by having to deal with radically different user interfaces presented by separate tools. An integrated environment reduces problem set-up time and, therefore, increases the productivity of the user. Finally, all users of the same environment share the same common data, providing consistency.

This chapter describes the approaches to application integration in process engineering. Three existing environments for process engineering are first described in detail to give an overview of the issues involved. These issues are further discussed in light of current research in both areas. The chapter ends with a discussion of new technologies that may lead to better, fully integrated, easier to use and more comprehensive tools for process engineering.

6.2.2 FRAMEWORKS FOR TOOL INTEGRATION

In this section, we describe three environments or frameworks for application integration in process engineering. The *épée* system, a *proof-of-concept* prototype system, was developed a few years ago with the aim of understanding the issues in application integration. A subsequent project, *Process Web Objects,* is a Web based system developed to demonstrate the potential of the Web as a framework for application integration and has been based on the experience gained in the épée project. An example of a tightly integrated system for application integration, ICAS, is then presented.

At the core of the first two systems is a framework for data interchange. Automating the interchange of data from one tool to another is not only a matter of translating from one format to another. It is a non-trivial exercise for a variety of reasons, summarized here:

- Each tool will typically use a different format for data representation. The data representations will often be based on a binary representation, specific to the type of platform and software used. Even with office productivity tools, this is an issue. Often, enterprises have to standardize on a single word processing tool to ensure that problems with document formats do not arise. For process engineering, this problem is compounded by the large number of diverse tools required and because these tools come from a large number of vendors. A translation from one format to another is therefore likely necessary to integrate any two tools. If there are n tools, there are potentially n^2 translation procedures required.
- Even if the formats were to be the same, the information embodied in the data set may require interpretation. Each application has its own set of assumptions and these assumptions may be in conflict between the tools to be integrated. This is particularly true when comparing tools used early in the design process to those used much later. For early design, quantities may be considered to be approximate or may not even be required. Once a flowsheet design has been chosen, however, many of the quantities will be assumed to be fixed (and accurate).
- Process engineering is a team based activity and data sharing amongst the members of the team, and the applications each member uses, is required for an efficient execution of the design process. Issues of concurrency arise. For instance, suppose one engineer is responsible for the design of a reaction section of a process and another is responsible for the separation section. Design decisions made by one engineer will affect the design generated by the other engineer. The effect of decisions made by each engineer will need to be recorded if not automatically taken into account.
- Process engineering is just one part of the activities in an enterprise. Issues that affect the enterprise will affect the needs for tool integration. For instance, the regulatory environment will typically lead to the need for clear and precise audit trails. Tool integration, which happens automatically, must therefore generate the data necessary to produce a suitable audit trail. This is particularly crucial with respect to the decomposition of design described in the previous point.

The third system has no requirements of data exchange because the system is tightly coupled. All the tools within this system share a common data format. Nevertheless, the issues presented above are still relevant and should be considered in the development of any integration environment.

6.2.2.1 The épée system

Tool integration requires not only the ability to transfer data from one application to another but also interpretation of the data, management of the applications used, and support for multiple users accessing the same data. Two of the authors were involved, at the University of Edinburgh, in the development of a prototype system for application integration, which attempted to address all of these issues. This prototype system is épée, the *Ecosse Process Engineering Environment* (Ballinger *et al.*, 1994; Costello *et al.*, 1996). It was developed to support process engineering in general, although with emphasis on early, conceptual design. Although the épée project was terminated a few years ago, a description of the main features of the system is useful for understanding the issues in tool integration and how they may be tackled.

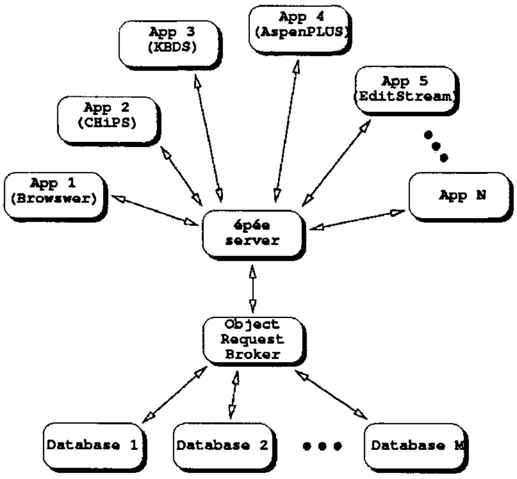


Figure 2. Architecture of the épée system.

The épée system is based on a modular architecture (see Fig. 2). At the centre, is a server that responds to requests from applications. These requests include accessing data objects previously created, storing new data objects, or requesting the application of methods on given data objects. Applications are invoked by the server in response to method application requests by either the user or by another application. The server is network-aware and applications may be started on any system in the network, as required (for instance, some software may be licensed only on specific systems). The system is controlled through a set of databases, which describe the applications that are available, and what methods they provide.

Data representation

The core of the épée system is an extensible object oriented data representation. The aim is to provide an application neutral data representation. Applications can be written to use this data representation directly, through the application programming interface (API) provided by épée. Existing applications can be integrated by implementing a wrapper application, which translates between the neutral representation and the representation used by the application. To ensure maximum portability, the data are encoded in ASCII.

Targeting conceptual design means that it is difficult, if not impossible, to pre-define the full set of data types required or even fully define any single type. For instance, the definition of a process stream cannot be fixed. In the earliest stages, a stream may simply consist of a list of components with little or no information about the actual amount of each component in the stream. In fact, the list of components may be only partially defined. In later stages, however, process streams will often include multiple phases, each phase defined by a fully specified list of components and their flows, a temperature, a pressure, and so on. One aim of the épée system was to enable a wide range of applications to work with the same basic object definitions regardless of the level of detail required.

An épée object consists of a set of *slots*. Each slot consists of a name and its value. The definition of each slot specifies what type of data values it should hold. Certain collections of slots are named and defined as *templates*, mostly for convenience. For instance, Fig. 3 and 4 show the slots that make up the épée stream and mixture templates.

In the definition of the mixture template, we see that the Temperature and Pressure slots represent real values with given ranges and units. The Component_Fractions slot is defined to contain a list of Component_Fraction objects. EGO and EDO are the épée Generic Object and the épée Design Object respectively. The templates show that both IS_A and HAS_A types of relationships between objects exist.

The type of object is defined implicitly by the set of *slots* it contains. That is, an object is not one type or another but is simply a collection of slots. When a method is invoked, the system matches the target object with the requirements of the method. These requirements are specified as a set of slots. If the object has the slots specified, it will be deemed suitable as the target for the method. In practice, the set of slots required for a method will be specified through the specification of a template, essentially a class definition for a type of object. Nevertheless, the épée objects are not explicitly typed and it is only at method invocation that the suitability of an object for the method is checked. This makes the system completely extensible. The implicit dynamic typing of objects, only when methods are applied, imposes no restrictions on what types of objects can exist or what applications can be supported. Furthermore, it provides the capability of using objects that are partially instantiated within tools during the early design stages. These same objects can later be used in more detailed analysis once they are fully defined.

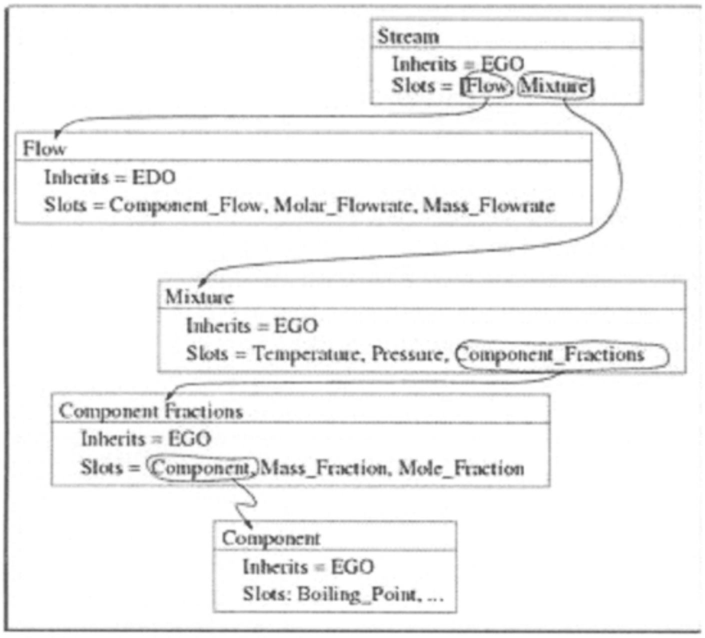

Stream
Inherits = EGO
Slots = [Flow, Mixture]
Flow
Inherits = EDO
Slots = Component_Flow, Molar_Flowrate, Mass_Flowrate
Mixture
Inherits = EGO
Slots = Temperature, Pressure, Component_Fractions
Component Fractions
Inherits = EGO
Slots = Component, Mass_Fraction, Mole_Fraction
Component
Inherits = EGO
Slots: Boiling_Point, ...

Figure 3. Definition of stream class.

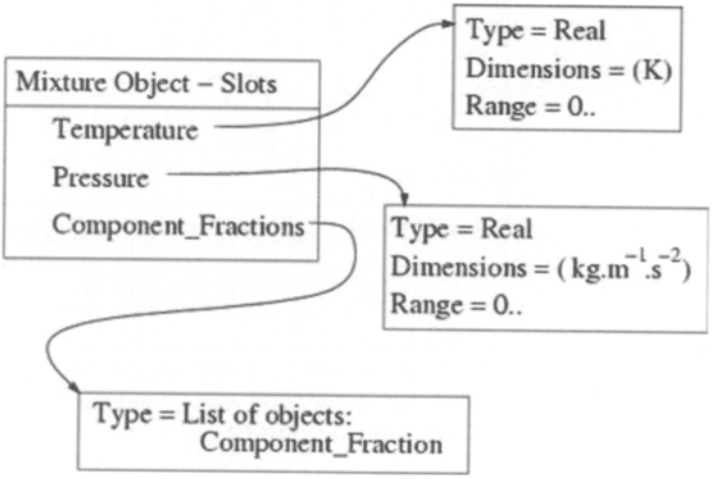


Figure 4. Definition of mixture class.

The definitions of templates, slots, methods and applications are all stored in a set of databases. Methods are defined with a specification of the minimum amount of information required to work with an object. The method definition also includes a list of applications (actual programs), which provide the particular method. Users may specify a preference when there is a choice of application that provides a given method. Applications are spawned as required, using a distributed computing application layer, which can make use of networked computers when appropriate. This can be useful, for instance, when a particular piece of software can only be run on a specific computer, a computer different from the one the engineer is using interactively.

Fig. 5 shows a snapshot of the épée console application displaying currently active applications (known as *Providers*) and the methods they provide. Each application may provide more than one method and a given method can be provided by more than one application. The CHiPS application (Fraga & McKinnon, 1994), for instance, provides a set of different synthesis methods, including heat integrated process synthesis, the use of rigorous models, and a simple short-cut model based procedure.

```
~}--- ps
id     | Provider@Machine     | Info On         | Methods & Control Interests
=======================================================================================
262164 | esh@(?)              | Busy with -->   | Being An Epee Shell
---------------------------------------------------------------------------------------
262157 | Select@tain          | Ready to use    | Select.Slot
       |                      |                 |
---------------------------------------------------------------------------------------
262154 | CHiPS@tain           | Providing -|    | Design.Sequence.Separation.HeatInt
       |                      |            |    | Design.Sequence.Separation.Rigorou
       |                      |            ->   | Design.Sequence.Separation.Simple
       |                      |                 | Design.Sequence.Separation
       |                      |                 | Synthesis.Process.Separation
       |                      |                 |
---------------------------------------------------------------------------------------
262151 | Browser@(?)          | Providing -|    | Get.Stream
       |                      |            |    | Get.Process
       |                      |            |    | Get.Mixture
       |                      |            |    | Get.Generic_Object
       |                      |            |    | Get.Component
       |                      |            |    | View.Contents
       |                      |            |    | View.History
       |                      |            |    | Browse.Structure
       |                      |            ->   | Browse.Objects
       |                      |                 | Browse.History
       |                      |                 |
---------------------------------------------------------------------------------------
262149 | Db.Srver@tain        | Ready to use    | Database.List_Contents.GDB
       |                      |                 | Database.Publish.GDB
       |                      |                 | Database.List_Contents
       |                      |                 | Database.Publish
       |                      |                 | Database.Retrieve
       |                      |                 |
---------------------------------------------------------------------------------------
~}--- █
```

Figure 5. épée console output.

Third party tool integration

The main problem with a system like épée is that applications have to be written to use the system specifically. In practice, however, until a standard for tool integration is adopted, this is unlikely to be the case. The situation is very much a *chicken and egg* scenario in which any application integration environment will not be a standard until the majority of applications are based on it and the applications will not make use of the environment until it becomes a standard.

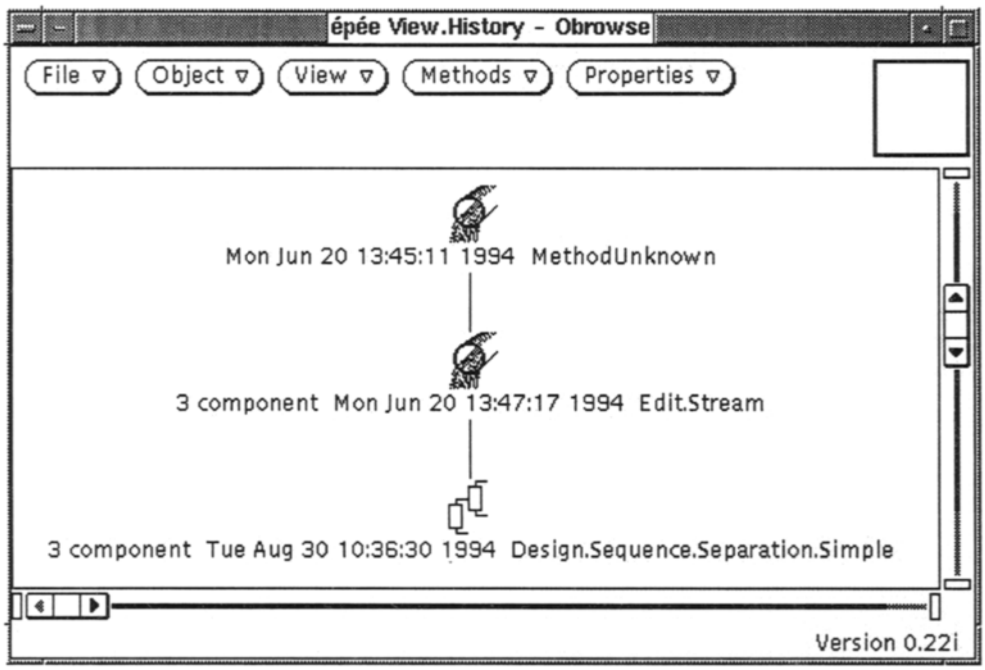
Until such a standard is adopted, the only alternative is to write *wrapper* environments for third party tools. The wrapper translates between the base data encoding provided by the integration environment and the native data format expected by the tool. Although somewhat onerous if a large number of applications are required, the number of translations is equal to the number of applications and not the n^2 required if each application were to be integrated with each other.

The use of a wrapper for integrating third party, legacy applications was demonstrated by Ballinger, Fraga & Bañares Alcántara (1995). This demonstration integrated the basic simulation procedure in Aspen PLUS into the épée system. A flowsheet was generated using CHiPS (Fraga & McKinnon, 1994) and the flowsheet generated was validated using Aspen PLUS. The output of the simulation was compared to the output of the synthesis procedure to determine the level of accuracy of the synthesis procedure. The controlling software was the KBDS system, a design support system described briefly in the next section.

Supporting the design process

Design is a team-based activity. Data must be shared amongst the members of the team. Furthermore, the actual process of design may take months or years and the deliverable is not just a process design but the decisions and steps that led to the actual design. Bañares Alcántara and co-workers have proposed a design support system, known as KBDS, for supporting this process (Bañares-Alcántara, 1991, 1995; Bañares-Alcántara & King, 1997; Bañares-Alcántara & Lababidi, 1995; King & Bañares-Alcántara, 1996). The épée system implements a small subset of the underlying concepts in KBDS to provide support for design and, moreover, provides some of the features necessary for application integration within such a support environment.

What differentiates épée from other proposals for data representation is the incorporation of the concept of *history* in to the object representation. All objects have a record of how and when they were created. This is illustrated briefly in Fig. 6. This figure shows the (very brief) history of a process design. The design is the bottom object, shown graphically by an icon together with some information about the object, including the name, the date on which it was created, and the method used to create it. In this case, the method is one provided by the CHiPS synthesis package (Fraga & McKinnon, 1994). The figure shows that this process design was the result of applying the method to the stream object shown immediately above the design object. This object was created over two months previously. The stream was created by applying the Edit.Stream method (a graphical interface for defining stream objects), starting from the stream object at the top of the figure, which was created directly by the system. As well as the information presented in this figure, each object has a record of which user or users were responsible for these actions.



```
Mon Jun 20 13:45:11 1994  MethodUnknown

3 component  Mon Jun 20 13:47:17 1994  Edit.Stream

3 component  Tue Aug 30 10:36:30 1994  Design.Sequence.Separation.Simple
```

Figure 6. Illustration of history of process design.

Method invocation seldom has any side effects. In the case of the synthesis procedure in CHiPS, a set of other objects is created. The history browser enables us to see these objects, if so required, as shown in Fig. 7.

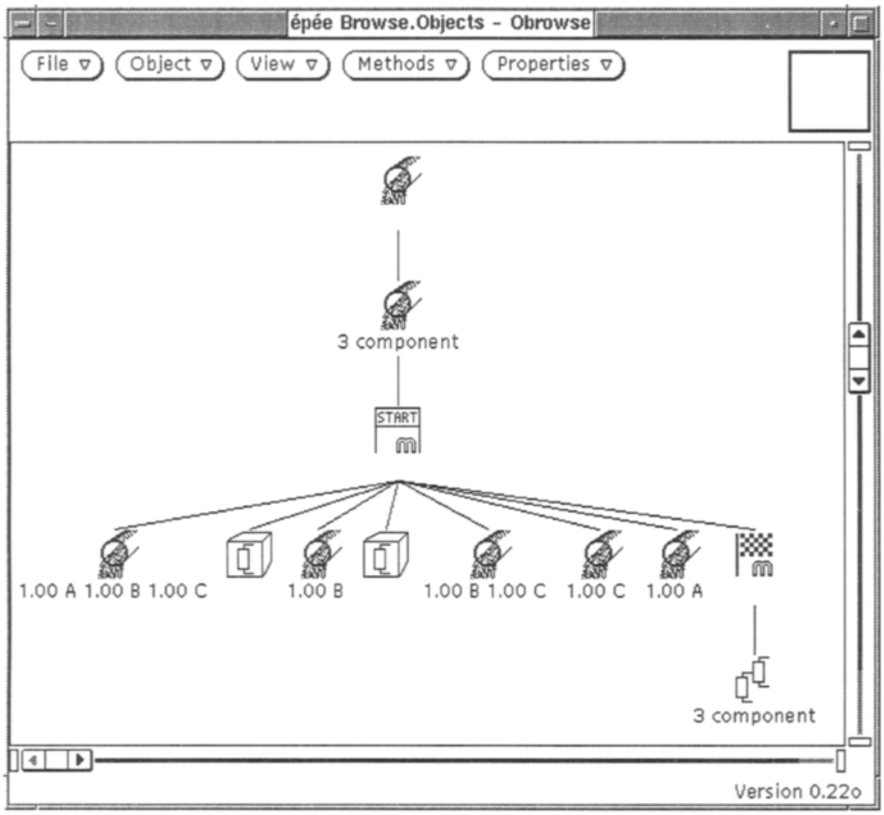


Figure 7. More detailed view of history of process design.

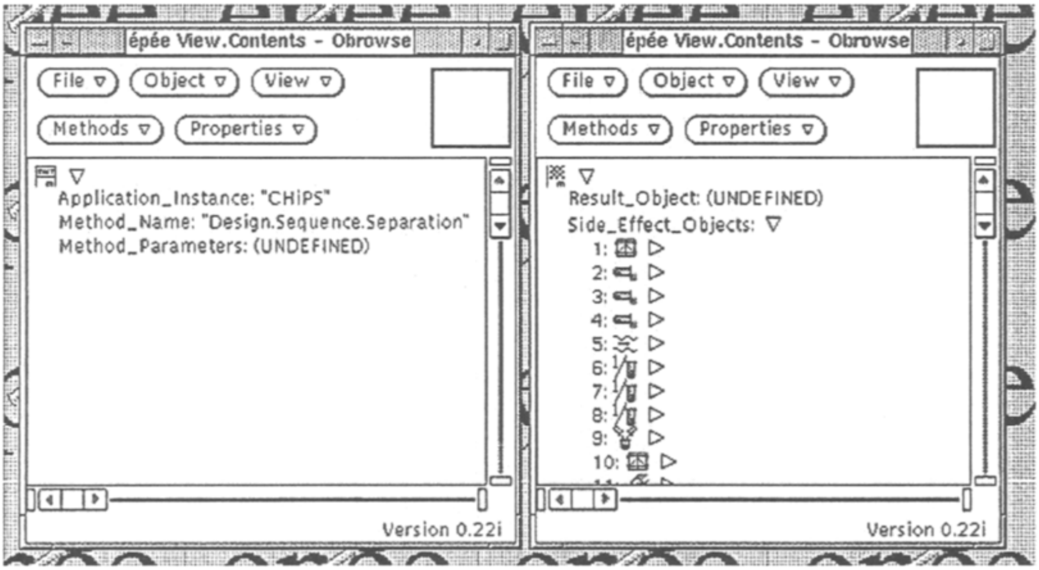
The more detailed view shows that the process design (bottom right of Fig. 7) is the result of applying a method to a stream object (2*nd* object from the top), as before. However, we now see a variety of other objects associated with the method invocation. The method invocation itself is represented by two special objects, a method start object, indicated by the start banner icon, and a method end object, indicated by the chequered flag icon. Linked into these two objects are the side-effect objects, a variety of stream and process block (processing units) objects. The view is history based so these side-effect objects appear as descendents of the method start object. However, they are also referenced from the method end object, as can be seen in Fig. 8. Shown in the figure are the method start and method end objects, in a content view mode. The method start has information about the actual application used. The method end has references to all the side-effect objects described above.

Summary of the épée system

The épée system was developed to demonstrate the potential of an integrated computer based environment to support process engineering. Two main criteria motivated the development of the system:

1. The need for an extensible architecture and an application neutral data representation to cater for the wide range of applications used in process engineering.
2. The support for a design and decision support system which is required for providing the environment necessary for team-based design and the extra functionality demanded by the enterprise.

The system that was developed achieved these two goals. However, it was only intended as a proof of concept and the system was not suitable for day to day use. In particular, épée did not scale well for large design tasks, mostly due to the database technology used (simple relational databases). Furthermore, there was a lack of support for the process of modelling; the data models used in épée were developed through meetings and with little rigour. More detailed analysis of the modelling requirements of the design process would have been required.



Application_Instance: "CHIPS"
Method_Name: "Design.Sequence.Separation"
Method_Parameters: (UNDEFINED)

Result_Object: (UNDEFINED)
Side_Effect_Objects:
1:
2:
3:
4:
5:
6:
7:
8:
9:
10:

Figure 8. Display of method start and end objects for synthesis example.

6.2.2.2 Process Web Objects: A Web based environment

As will be noted later, Lu *et al.* (1996) have incorporated the Web into an environment for tool integration. Andrews & Ponton (1999) have taken this a step further by making the Web, and the browsers used to interface with the Web, be the integration environment. When work on the épée system was started, the Web was in its infancy. Towards the end of épée's development, it was noticeable that many of the features, which had had to be custom designed for the épée server, were features of standard Web servers. Developers of design support and information management systems noted that many of their requirements could be met even by early hypertext systems. In fact, the initial development of KBDS investigated such a system, although its limitations soon became apparent and the vehicle was switched to a Lisp graphic environment. However, the power, and particularly the universal acceptance of the underlying standards that comprise the Web (http, HTML, CGI), suggested that the notion of an object based process engineering environment based on Web technology should be re-examined.

The initial objective of the Process Web Objects project was to see how much of the épée functionality could be reproduced using standard Web servers and creating objects as HTML documents. Subsequently, Web Objects have developed a wider and more flexible context, integrating synthesis and flowsheeting tools, spreadsheets, and have acquired some distinctive features of their own. A fuller description of Process Web Objects is given in Andrews & Ponton (1998) and an online demonstration is also available at www.chemeng.ed.ac.uk/people/ross/public/web_objects/introduction.html. The rest of this section gives an overview of the features and their implementation, particularly with reference to the épée system described above.

Object Hierarchy

This is broadly that of épée, viz components, mixtures, streams, processes etc. It is less systematic than that of épée, having developed in a rather *ad hoc* manner, and containing objects that were felt to be useful, but which do not really fit into a tidy hierarchy. New types of object have been defined to handle higher levels of process representation and abstraction for use in process synthesis. These include a *Task* object which specifies a design or synthesis task to be carried out, a *Domain* object which contains all the Process objects which can achieve a particular Task, and *Class* objects, several of which can contain groups of Process objects within a Domain have particular common features (Williams, 2000).

Object Structure

Objects are HTML documents. They contain slots which are either atomic, i.e. containing simple data, in the form of a text string, or are themselves objects. Atomic slots may either be fixed or changeable. Changeable slots are typically implemented as HTML text form elements. The user may decide changeable slots in one generation of an object shall be fixed in its descendants, but fixed objects cannot be unfixed. Slots, which are objects, are represented by hyperlinks. Thus the whole of the Web is accessible as part of a Web Object. Furthermore, anything that may be pointed at by a link can be part of an object. The link may be to text, graphics, audio, arbitrary files, or any entity which may be created by running a program in the cgi area of a server. This clearly provides great flexibility and richness.

Standard objects come with pre-specified slots. However, a method is available to create a new object with additional slots from a standard one.

Methods

As in épée, methods applied to an object generally create a new object. A key feature of épée was object persistence, i.e. once created, an object could never be destroyed. This highlights one limitation of web technology in this role; for security reasons programs running on the Web cannot access the client's workspace, and are limited even in what they can do to the desktop. Thus Web Objects are ephemeral and, in practice, new objects can only be created on a Web server. We have minimised this problem by using the ideas of a working session and of subsequent publication, as described below.

Web methods are in practice of three types, dependent on how they are implemented. The distinction is largely one of convenience and feasibility rather than of anything fundamental. The methods can also be categorized according to where they execute:

Server Methods and Method Application: Most methods are applied by programs running on the server to which the object is submitted as a form using CGI protocols. The list of methods applicable to an object is specified by a form select list and chosen by the user (Fig. 9). An object's methods are thus effectively defined within the object itself. Indeed they might almost be thought of as specialised slots, which cause an action to be performed when they are accessed. Some methods are common to all objects, while others are specific. A user can also create objects that inherit all the methods of a standard object but add a new method as a URL on a server. The user must provide code at that location to implement the method.

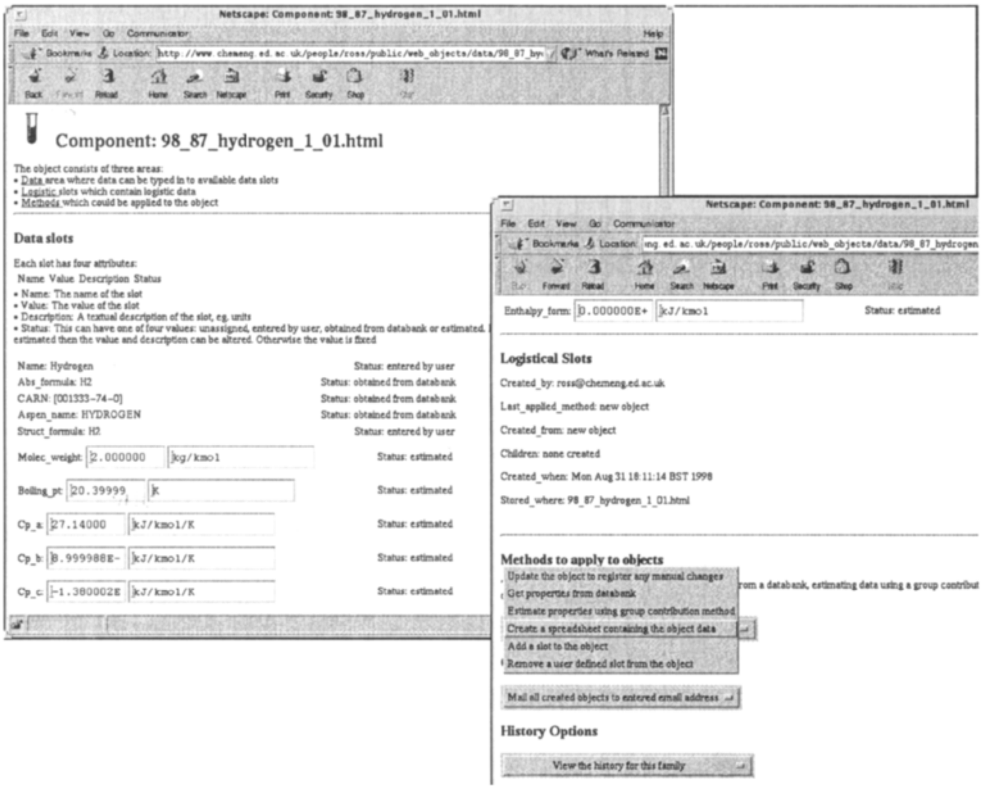
Most methods result in the creation of a new object. This is stored on the server and usually also sent as an HTML page to the user's browser. `Child' links to it are added to the object from which it was created and the new object is linked to the appropriate places in the History object.

Server methods may make use of any software accessible to the server. For instance, two methods available in the prototype are `Create an Aspen Model' which creates an Aspen input file object from a Process Object and `Get data from Aspen Model' which fills slots in a Process Object from an Aspen output file.

These can easily be combined into a single `Balance Process with Aspen' method with the AspenPlus system run by the server program.

Client Methods: Most Web Object methods run on a server but they may also, with some restrictions, be implemented on the client by Java Applets or some even in JavaScript. In principle, most server methods could be implemented in Java to run on the client. Although the Java security model allows a Java applet to write only to the server from which it was loaded, most of the existing server methods could be implemented this way. In a system with a large number of users, this could offer a significant reduction in server load.

JavaScript is much more restricted in that it cannot write to anywhere except a browser on the user's desktop, and is limited even there. In practice, JavaScript only provides `viewing' methods and can be used to change what the user sees, e.g. to change the units or form in which information from slots is displayed. However JavaScript can initiate action on a server or apply any public method of an applet in the object document, making it a convenient means of implementing user interfaces. An example of this, the `molecular typewriter', shown elsewhere in this book (Chapter 6.3 - Cameron and Ponton), is used as an interface for creating Component objects.


Netscape: Component: 98_87_hydrogen_1_01.html
File Edit View Go Communicator Help
Bookmarks Location: http://www.chemeng.ed.ac.uk/people/ross/public/web_objects/data/98_87_hyc
What's Related
Back Forward Reload Home Search Netscape Print Security Shop Stop

Component: 98_87_hydrogen_1_01.html

The object consists of three areas:
- Data area where data can be typed in to available data slots
- Logistic slots which contain logistic data
- Methods which could be applied to the object

Data slots

Each slot has four attributes:
Name Value Description Status
- Name: The name of the slot
- Value: The value of the slot
- Description: A textual description of the slot, eg, units
- Status: This can have one of four values: unassigned, entered by user, obtained from databank or estimated. estimated then the value and description can be altered. Otherwise the value is fixed

Name: Hydrogen		Status: entered by user
Abs_formula: H2		Status: obtained from databank
CARN: [001333-74-0]		Status: obtained from databank
Aspen_name: HYDROGEN		Status: obtained from databank
Struct_formula: H2		Status: entered by user
Molec_weight: 2.000000	kg/kmol	Status: estimated
Boiling_pt: 20.39999	K	Status: estimated
Cp_a: 27.14000	kJ/kmol/K	Status: estimated
Cp_b: 8.999988E-	kJ/kmol/K	Status: estimated
Cp_c: -1.380002E	kJ/kmol/K	Status: estimated
Enthalpy_form: 0.000000E+	kJ/kmol	Status: estimated

Logistical Slots

Created_by: ross@chemeng.ed.ac.uk

Last_applied_method: new object

Created_from: new object

Children: none created

Created_when: Mon Aug 31 18:11:14 BST 1998

Stored_where: 98_87_hydrogen_1_01.html

Methods to apply to objects

rom a databank, estimating data using a group contribut

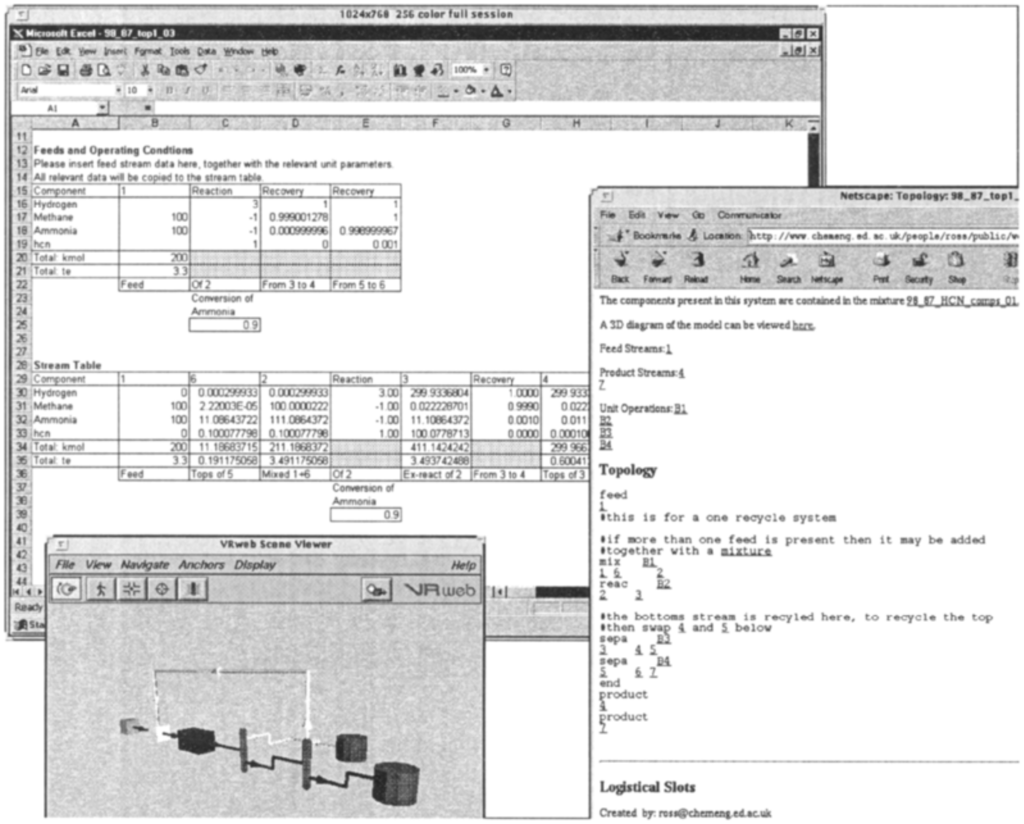
Update the object to register any manual changes
Get properties from databank
Estimate properties using group contribution method
Create a spreadsheet containing the object data
Add a slot to the object
Remove a user defined slot from the object
Mail all created objects to entered email address

History Options

Figure 9. Method selection in Web Process Objects interface.

Extensibility

As already noted, it is possible to create new versions of standard objects with additional slots and methods. Since methods can involve any program, which the server can run, complete flexibility and extensibility has been achieved. Since an object slot is a hyperlink, and hyperlinks can point to any type of file that is given a URL anywhere on the Web, Web Objects can effectively contain any kind of information. Furthermore, browsers may be set to use `plugins' to open particular types of file with a client application. For example, another method for a Process Object is `Create Spreadsheet model'. This creates a spreadsheet format file with appropriate extension, containing a linearised mass balance model of the process, either with user specified process parameters or those derived from a solved Aspen Process Object. Clicking on the hyperlink in the generated Spreadsheet Model object will, once browser options have been set, automatically bring this up in a spreadsheet on the client (Fig. 10).


1024x768 256 color full session
Microsoft Excel - 98_87_top1_03
File Edit View Insert Format Tools Data Window Help
VRweb Scene Viewer
File View Navigate Anchors Display Help
VRweb
Netscape: Topology: 98_87_top1_
File Edit View Go Communicator
Bookmarks Location
Back Forward Reload Home Search Netscape Print Security Shop

Feeds and Operating Condtions
Please insert feed stream data here, together with the relevant unit parameters.
All relevant data will be copied to the stream table.

Component	1	Reaction	Recovery	Recovery
Hydrogen		3	1	1
Methane	100	-1	0.999001278	1
Ammonia	100	-1	0.000999996	0.998999967
hcn		1	0	0.001
Total: kmol	200			
Total: te	3.3			
	Feed	Of 2	From 3 to 4	From 5 to 6

Conversion of Ammonia 0.9

Stream Table

Component	1	6	2	Reaction	3	Recovery	4
Hydrogen	0	0.000299933	0.000299933	3.00	299.9336804	1.0000	299.933
Methane	100	2.22003E-05	100.0000222	-1.00	0.022228701	0.9990	0.022
Ammonia	100	11.08643722	111.0864372	-1.00	11.10864372	0.0010	0.011
hcn	0	0.100077798	0.100077798	1.00	100.0778713	0.0000	0.00010
Total: kmol	200	11.18683715	211.1868372		411.1424242		299.966
Total: te	3.3	0.191175058	3.491175058		3.493742488		0.60041
	Feed	Tops of 5	Mixed 1+6	Of 2	Ex-react of 2	From 3 to 4	Tops of 3

Conversion of Ammonia 0.9

http://www.chemeng.ed.ac.uk/people/ross/public/w

The components present in this system are contained in the mixture 98_87_HCN_comps_01.

A 3D diagram of the model can be viewed here.

Feed Streams: 1

Product Streams: 4
7

Unit Operations: B1
B2
B3
B4

Topology

```
feed
1
#this is for a one recycle system

#if more than one feed is present then it may be added
#together with a mixture
mix   B1
1 6    2
reac    B2
2    3

#the bottoms stream is recyled here, to recycle the top
#then swap 4 and 5 below
sepa    B3
3    4 5
sepa    B4
5    6 7
end
product
4
product
7
```

Logistical Slots

Created by: ross@chemeng.ed.ac.uk

Figure 10. Integration of flowsheet viewer and spreadsheet calculator in Web Process Objects.

The History Object, Work Sessions and Publication

When a new Web Objects `Work Session' is initiated, the user is allocated space on the server for the objects that will be created. The first object created is an empty History object. The history object will contain a tree diagram showing all objects created, with hyperlinks to those objects. Clicking on an object brings it up in a browser window. Any method applied to it that creates a new object will enter a new branch into the History object. Additionally, objects have logistical slots, which point to parent and child slots. The server workspace is preserved so long as it is deemed to be required by the user (interpretation of this depends on the type of user). It is normally private to one user and password protected, although it can in principle be shared. The general method `Mail all created objects...' compresses all objects in the workspace into a zip file and emails them to their registered owner. The owner can then unzip the directory and install it in his own web space. By telling others the URL of the History object, the user effectively `publishes' the entire workspace in the sense used in n-dim

(Westerberg *et al.*, 1997). Anyone with access to the URL can bring up an object created by the user and, if the new user has access to a Web Objects server, can apply a method to and start a new family of descendants which belong to the new user rather than to the owner of the original object.

6.2.2.3 The ICAS system

The two systems described above are frameworks for integrating diverse applications, taking care of the data management issues presented early in this chapter. These approaches assume a loose interaction between the different tools. To be effective, a loose integration approach requires that all the tools present a common look and feel to the user and that all the tools are able to interpret any data exchanged correctly. The latter is particularly difficult to guarantee in such a loosely coupled environment. In this section, we present an environment, which is fully integrated, using a common data representation and a common user interface.

The *Integrated Computer Aided System*, ICAS, developed at CAPEC, Department of Chemical Engineering of the Technical University of Denmark (Gani, Hytoft, Jaksland & Jensen, 1997) is an example of a fully integrated computer environment for process engineering (http://www.capec.kt.dtu.dk). ICAS combines computational tools for modelling, simulation (including property prediction), synthesis/design, control and analysis in a single integrated system. These computational tools are present as *toolboxes*. During the solution of a problem, the user moves from one toolbox to another to solve problems requiring more than one tool. For example, in process synthesis, one option is to define the feed stream, then analyse the mixture (analysis and utility toolbox), generate a flowsheet (synthesis toolbox), optimise the flowsheet (design toolbox), and finally verify the design (analysis toolbox). From any toolbox it is possible to invoke the simulation engine to perform steady state and dynamic simulations for batch and continuous process operations. From the synthesis toolbox, it is possible to invoke the solvent design tool (from the design toolbox) if a solvent is required for a specific separation task. There is also a utility toolbox, which determines properties, phase diagrams, *etc.*, which can be used by the other toolboxes or by the user to analyse the behaviour of the specified system.

The following tools are currently available in ICAS:

- Property prediction: Tools include CAPEC database, pure component property prediction (ProPred), mixture property prediction (TML), model parameter estimation (TML) and thermodynamic model selection (TMS).
- Modelling: Computer aided model generation (ModDev), model analysis, translation and solution (MoT) plus addition of user-defined models (MoT).
- Process synthesis and design: Tools include computer aided flowsheet generation (CAPSS), design of reactive/azeotropic distillation (PDS),

configuration of distillation columns (PDS), solvent search/design (ProCamd).
- Reactions: Tools include reaction database, kinetic model parameter estimation, reaction path synthesis based on data (RAS).
- Simulation: The ICAS-SIM simulation engine allows steady state and dynamic simulation of continuous as well as batch operations (BRIC).
- ReacPar: Reaction kinetics modelling, including parameter estimation.
- Simul: Simulator engine - includes an integrated steady state and dynamic simulation features and DynSim (a dynamic simulation system).
- Control: Design and analysis of control systems, including a MPC toolbox for model predictive control.
- BRIC: Batch operation records and simulation.

Fig. 11 gives an overview of what is available in ICAS, how the various tools are connected and the functions of each tool. The figure shows that ICAS consists of four major parts. That is, problem definition, simulation engine, tools for extending the system through the definition of, for instance, new compounds, models and reactions, and specific process engineering tools which may be used to solve sub-problems. In problem definition, the user defines the problem to be solved by selecting from the available options. The job of the simulation engine is to obtain a numerical solution to all the model equations representing the specified system/process. Knowledge based systems are available to select phenomena (property) models, solvents, separation techniques for a specified separation task and numerical methods (simulation strategy).

ICAS is available to CAPEC member companies and is being used as a teaching aid in courses on separation processes, chemical process design, advanced process simulation/optimisation and in process & tools integration at graduate and post-graduate levels. In industrial use, the tools of ICAS help to reduce the time of project work within a company. As a teaching aid, it provides students with tools that help to generate the data/information that is not part of the course but is needed to solve the course problems. A typical example is distillation design - it is necessary to compute phase diagrams, to find solvents (in case of solvent based distillation), to configure the distillation column and finally to verify the design by simulation. Having all the tools present in an integrated manner helps users not only to concentrate on the problems related to the specific application but also to become aware of what information is needed and how this information can be obtained. The additional knowledge helps the user to consider more alternatives and to obtain better solutions, which otherwise would not be possible to obtain.

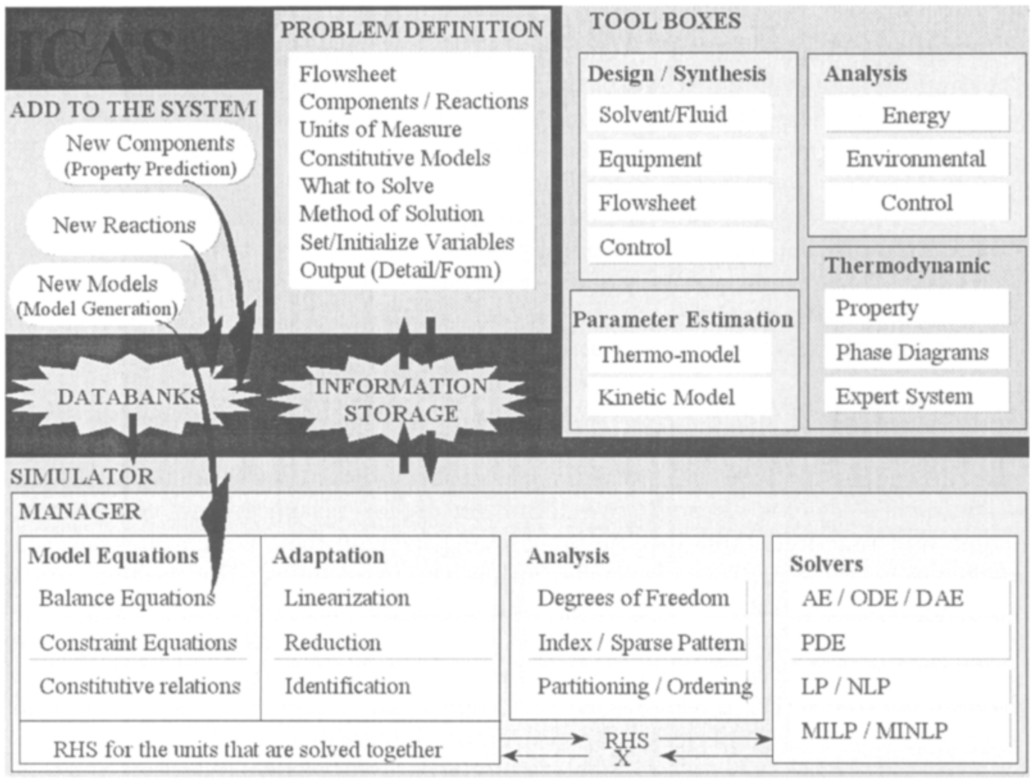


Figure 11. Overview of the ICAS system.

6.2.3 CURRENT PROJECTS IN TOOLS INTEGRATION

The systems presented above demonstrate the capabilities of integrated systems and how some of the issues in application integration can be tackled. These systems are a small sample of the work currently in progress in different research groups (and software vendors). An overview of some of the current projects in application integration is presented in this section, grouped according to the particular issue they address.

6.2.3.1 Data models for integration

The importance of the underlying data representation to process integration is emphasized by the work of Bayer, Shneider & Marquardt (2000). As part of a larger programme of research aiming to develop an integrated environment to support process engineering, they have developed a conceptual data model, which is designed for the integration of existing data models.

McGreavy *et al.* (1996) use STEP models, represented using the EXPRESS language, for computer aided concurrent engineering. They have developed a multi-dimensional object oriented information model (MDOOM), based on STEP. They decompose the information required for the whole design process into process, facilities and equipment. The distinction between these is that processes generate products using the facilities, which are a logical abstraction of the process plant, presenting a functional view of the equipment that makes up the plant. STEP is intended for more detailed representation than is typically required for conceptual design and, as a result, the system is most appropriate for use late in the design process.

Maguire *et al.* (1998) propose the use of *intelligent agents* to represent data as well as to provide the mechanism for actual integration. They define intelligent agents as objects with extensions to provide the following properties:

Flexibility:	The ability to learn from the environment.
Planning:	The ability to synthesize a sequence of actions in order to achieve a specific goal.
Prediction:	The ability to predict what the actions of other agents will be.
Selectivity:	The ability to choose the appropriate data from a larger amount of information.
Robustness:	The ability to cope with failure.

Their view is that agents can be used for data representation, for managing the integration of applications, and for manipulation of the models. The use of agents is appealing but their potential has yet to be demonstrated. Maguire *et al.* limit themselves to examples which are within the remit of any object oriented system and so the potential of agents is not fully demonstrated.

Agents can ease the management and representation of data by combining the two aspects. What is not entirely clear at this stage is how the combination affects the extensibility of a system built up using agents. There is also the issue of audit trails and, if the agents are truly autonomous, consistency within a team based project and guarantee of the correctness of the results generated could be difficult.

6.2.3.2 Data sharing

Although tool integration is an important aspect of computer aided process engineering, the problem of data sharing is more general. Data sharing can include tool integration but also includes sharing data amongst the members of a team and passing information from one team to the next in the design process. The issues that apply to tool integration apply even more so in data sharing.

Some fundamental questions need be addressed, including the following:

- When are data objects available for use by other members of a team and other teams?
- What happens when somebody changes an element of data?
- What is the context of a data item and how does this context change as the design progresses?

These issues have been considered in detail by Westerberg *et al.* (1997). They have developed a system, *n-dim*, for describing the modelling process. Of particular relevance is the concept of *publishing* whereby data are made available to other members of the team and other teams. Once published, data cannot be changed, ensuring global consistency and providing support for audit trails.

Related issues arise from considering the role of design within the enterprise. Struthers & Curwen (1998) have developed an agent-based system for tool integration, through the use of *agents*, known as VIBE. The aim of this environment is to support enterprise or business processes as well as the process of design.

6.2.3.3 Application interfaces

The CAPE-OPEN and Global CAPE-OPEN projects, described in detail elsewhere in this book, deal with the definition of open interfaces for process engineering applications. Although they do not deal directly with the integration of these applications, by defining open standards for access to the different components that make up the applications, integration becomes feasible through the development of the underlying data representation and manipulation layers.

McGreavy *et al.* (1996) use message passing between agents as the integration mechanism. In their work, agents are self-contained, concurrently executing software processes. An agent encapsulates state information and can communicate with other agents via message passing. A knowledge based rule system is used to co-ordinate the application of tools. The decomposition of applications used within design by agents is hierarchical. Each agent may have sub-agents that are agents that are restricted to communicate only with other sub-agents contained within the same agent. Finally, the co-ordination of agents within the framework is controlled by the use of a Petri Net model.

6.2.3.4 User interaction

An interesting aspect of the work of McGreavy *et al.* (1996) is that the user is also considered to be an agent, treated as any other application within the framework. The user interacts with other agents through the graphical interface. This interface allows a user to browse the list of agents available. McGreavy *et al.* further argue that it is necessary to consider models for the management of the design teams and the role of the project manager as well as modelling the design task itself. This fits naturally with their view of users as agents.

Lu *et al.* (1996) present a web based manager for process engineering tool integration. Based on the MDOOM models (McGreavy *et al.*, 1996), Lu et al. introduce the concept of *hyper-objects* to represent the inter-linked nature of objects in an object-oriented database. This is similar in concept to the hypertext links that underpin the Web. The links between objects bind these objects to the common tools used to create the objects, although this binding is dynamic. The activities of the engineers change as the design progresses and, hence, the tools used also change. A *hypermanager* is introduced to manage and represent these dynamic object groupings.

The hypermanager incorporates a user interface for viewing and managing objects. The tool is written in tk/tcl and provides a graphical representation of the process design. A Web browser interface is also available. The interface allows the user to inspect individual components of the design and provides access to tools such as editors, simulators, *etc.* The main feature of this system is the decomposition of the data into the process, facilities and equipment groupings described above and the ability to switch between the different groupings easily.

6.2.3.5 Design and decision support

As described above, application integration is not only about sharing data between applications; it is also about managing the design process. This management must support the requirements of the enterprise. One aspect is to provide a design support system, an example of which is the KBDS system described earlier (Bañares-Alcántara & co-workers). The aim of a design support system is to cater for the requirements of the engineer when tackling a large and complex design problem, to support team based activities, and to meet the demands imposed on the enterprise by society. The core of any such system is the need to manage the complexity of design as a process. KBDS provides support for recording the process of design, including a full history of decisions taken (including information on the reasons for decisions and the arguments made both for and against any positions taken in the decision process), and for managing the complex interactions between multiple designers working on a given design problem.

Han *et al.* (2000) have also developed a framework for concurrent process engineering. Their framework is based on agents and the development of a knowledge based interchange format (KIF) together with a knowledge query and manipulation language (KQML). Furthermore, they have integrated a decision management system. The system is written using the Java Agent Template package, JATLite, available from Stanford University [java.stanford.edu]. The framework presented is similar, in concept, to KBDS. It has been demonstrated on a plant layout design problem. The user and three agents are integrated with the user interfacing through a CAD package. The underlying system ensures that any changes made by the designer violate no constraints by communicating with the appropriate agents when constraints are violated.

6.2.4 FUTURE TRENDS

The development of a framework for tool integration in computer aided process engineering or a fully integrated process engineering environment requires the following elements:

1. A data representation suitable for process engineering applications throughout the life cycle of a design, including commissioning, operation, and decommissioning.
2. A management layer, which starts the appropriate applications and manages the communication of data between applications, possibly including the translation between native and integration data representations.
3. The specification of an application programmer's interface, which allows new applications to be written to fit into the integration environment.

The systems described above address one or more of these issues. The épée system, for instance, to some degree addresses all of them. However, there have been and will be developments in the area of computing which will make each of these elements easier to implement. These are described in this section.

6.2.4.1 The Web

Arguably, the most important development in computer use over the past decade has been the introduction of the Web. The web provides an architecture neutral protocol and user interface specification. Through the Web, any computer (or, more generally, any form of connected device including process equipment, sensors, phones, etc.) can access and manipulate data on any other system with a Web server and with suitable access rights. The http protocol is simple and suitably efficient for the requirements of application integration. The browsers used to access the Web provide an interface the users are comfortable with. Accessing applications through such an interface can ease the difficulties associated with using a large variety of tools. The Process Web Objects project, described above, is an example of the use of Web technologies for application integration.

Of particular note, however, are the more recent introductions of two languages for use with the Web, Java and XML. Java is meant to be a portable high-level language, similar to C++ but somewhat easier to program. Originally intended for embedded systems, the language was targeted at Web applications and, in recent times, has found a particular niche as a Web server language. Being a full programming language, it provides the features necessary for the development of a complete integration environment.

The portable nature of Java programs makes them ideal for use as agents, as described above. For this reason, there are a variety of packages for agent development in Java:

1. The Java Agent Template, from Stanford University [java.stanford.edu], provides the basic functionality for writing agents. Communication between agents can be in KQML format, as used by Han *et al.* (2000), as well as arbitrary data.
2. JATLite [java.stanford.edu], Java Agent Template Lite, is another package for agent development, also from Stanford University. This is the package used by Han *et al.* (2000).
3. JAFMAS [www.ececs.uc.edu/~abaker/JAFMAS] is a framework for agents in early development. It also supports KQML.
4. The Aglets Workbench [www.trl.ibm.co.jp/aglets] provides support for migrating agents, agents that can move from one computer to another. This can prove to be useful not only with mobile computing but also for applications with varying computational demands.

XML, the extensible markup language [w3c.org], is a language for describing data. Through the definition of a document type definition (DTD), XML can be used to represent any type of data. It is ideal for the data representation needs of process engineering applications. Tools are now available for developing applications, which process and generate XML documents. In particular, there is a natural synergy between XML and Java. For example, JSX [www.csse.monash.edu.au/~bren/JSX] provides a set of classes for automatically generating XML code for any Java object and for creating an instance of a Java class based on XML input. This is similar in concept to the Java serialization procedures except that the data is represented in XML making it suitable for integration with non-Java applications and for dissemination through Web browsers. Although still in rapid development, the current version of JSX is usable and useful.

6.2.4.2 The Grid

A recent development in networked computing is the Grid. This is a view of networked computers as a dynamic resource, which can be used by applications when required. A good overview of the Grid, and its potential, can be found on the

Web site for the Globus project [www.globus.org]. Globus is an open source project, which aims to develop the underlying framework to support Grid applications.

One of the aims of the Grid is to deliver guaranteed performance over the Internet, allowing a software developer to treat remote computational resources as if they were local. For process engineering applications, the Grid can be useful in a variety of ways. For instance, it can be used to provide access to applications, which must reside on specific systems on the network (for example, due to licensing restrictions). It can also be useful for connecting data monitoring stations to simulation and control packages.

6.2.4.3 Data management infrastructure

Management and manipulation of both data and applications requires a repository of information and protocols for transferring data between applications and computers. The repository of information is needed by the integration environment and typically underpins the whole system. The framework developed by Han *et al.* (2000), for instance, uses an SQL database to store design information generated by the CAD package. However, that same system stores the knowledge base within a Lisp program. The experience gained from the épée system demonstrates that the underlying database technology is crucial to the scalability of the system. Object oriented database management systems (OODBMS) have been in existence for a relatively long time now and are stable and easily available. Through the development of the ODBC (Object Database Connectivity) and JDBC (Java Database Connectivity) standards, accessing these databases programmatically is straightforward.

Previously, issues about the transport protocols, such as Microsoft's OLE & COM protocols versus the CORBA standard, were important. Recently, the actual transport protocols have become less important, replaced instead by discussions about the middle layer, which is responsible for the actual development of interacting applications, often known as *components*. Two initiatives are currently being advertised, the *Brazil* [http://www.sun.com/research/brazil/] framework from Sun Microsystems and the *.net* [http://msdn.microsoft.com/net/] framework from Microsoft. It is still too early to say if either of these will prove to be useful for process engineering application integration. What is interesting, however, is that both of these initiatives use the Internet as the underlying transport medium. They both aim to make it easier to develop applications from small components and to provide user interaction via distributed computation.

The combination of Java, XML, object oriented database systems, and general Internet integration initiatives provides the necessary underpinnings for the development of an integration environment. Technology itself is no longer an issue.

6.2.5 CONCLUSIONS

Tool integration for computer aided process engineering is not simply a matter of data translation. To be effective, tool integration requires the combination of design support environments (Bañares-Alcántara & King, 1997; Han et al., 2000; Lu *et al.*, 1996; McGreavy *et al.*, 1996; Struthers & Curwen, 1998; Westerberg *et al.*, 1997) with methodologies for manipulating and interpreting process engineering data throughout the design process (Bayer *et al.*, 2000; Braunschweig *et al.*, 2000; Maguire *et al.*, 1998). The aim of tool integration is to provide the engineer a seamless environment through which the large variety of tools required for process engineering can be access. Such an environment, whether the result of a loosely coupled tools through a system like épée or as a tightly integrated system such as ICAS, is essential for tackling the increasingly complex problems in process engineering.

New technologies, particularly Web or Internet oriented, will provide the standards required to ensure a stable target for developers of tool integration environments. XML, in particular, has the potential to provide an architecture neutral language for representing process engineering data. The development of formal requirements for these data will be necessary first. The development of easy to use Java based agent packages will also make development of design environments easier and more portable. Finally, standards developed by the process engineering community, such as those expected from the Global CAPE-OPEN project, will be crucial for the long-term development of integrated process engineering tools.

6.2.6 REFERENCES

1.Andrews, R. & Ponton, J. W. (1998), *A Process Engineering Information Management System Using World-Wide Web Technology*, AIChE Symp Ser No 320, **94**, 501-506.

2.Ballinger, G. H., Bañares Alcántara, R., Costello, D. J., Fraga, E. S., King, J., Krabbe, J., Laing, D. M., McKinnel, R. C., Ponton, J. W., Skilling, N., & Spenceley, M. W. (1994), *Developing an environment for creative process design*, Chemical Engineering Research and Design, **72**(A3), 316-324.

3.Ballinger, G. H., Fraga, E. S. & Bañares-Alcántara, R. (1995), *Case study in process engineering tool integration*, Computers & Chemical Engineering **19**(Suppl.), S759-S764.

4.Bañares-Alcántara, R. (1995). *Design Support Systems for Process Engineering. I. Requirements and Proposed Solutions for a Design Process Representation*, Computers & Chemical Engineering, **19**(3) 267-277.

5.Bañares-Alcántara, R. (1991). *Representing the Engineering Design Process:*

Two Hypotheses, Computer-Aided Design (CAD) Journal, **23**(9) 595-603.

6.Bañares-Alcántara, R. & King, J. M. P. (1997), *Design Support Systems for Process Engineering. III. Design Rationale as a Requirement for Effective Support*, Computers & Chemical Engineering, **21**(3), 263-276.

7.Bañares-Alcántara, R. & Lababidi, H. M. S. (1995). *Design Support Systems for Process Engineering. II. KBDS: an Experimental Prototype*, Computers & Chemical Engineering, **19**(3) 279-301.

8.Bayer, B., Schneider, R., & Marquardt, W. (2000). *Integration of data models for process design - First steps and experiences*, Computers & Chemical Engineering, **24**(2-7) 599-605.

9.Braunschweig, B. L., Pantelides, C. C., Britt, H. I. & Sama, S. (2000). *Process modeling: The promise of open software architectures*, Chemical Engineering Progress, **96**(9) 65-76.

10.Cameron, I. & Ponton, J. W., Web based systems, <this book – Ch. 6.3>.

11.Costello, D. J., Fraga, E. S., Skilling, N., Ballinger, G. H., Bañares Alcántara, R., Krabbe, J., Laing, D. M., McKinnel, R. C., Ponton, J. W., & Spenceley, M. W. (1996), *épée: A support environment for process engineering software*, Computers & Chemical Engineering **20**(12), 1399-1412.

12.Fraga, E. S. & McKinnon, K. I. M. (1994). *CHiPS: A process synthesis package*, Chemical Engineering Research and Design, **72**(A3) 389-394.

13.Gani, R., Hytoft, G., Jaksland, C. & Jensen, A. K. (1997). *An integrated computes aided system for integrated design of chemical processes*, Computers & Chemical Engineering, **21**(10) 1135-1146.

14.Han, S. Y., Kim, Y. S., Lee, T. Y. & Yoon, T. (2000). *A framework of concurrent process engineering with agent-based collaborative design strategies and its application on plant layout problem*, Computers & Chemical Engineering, **24**(2-7) 1673-1679.

15.King, J. M. P. & Bañares-Alcántara, R. (1996). *Extending the Scope and Use of Design Rationale Records*, Artificial Intelligence in Engineering Design Analysis and Manufacturing (AIEDAM) Journal, **11**(2) 155-167.

16.Lu, M. L., Batres, R., & Naka, Y. (1996). *A hypermanager for a computer integrated concurrent process engineering environment*, Computers & Chemical Engineering, **20**(12) S1155-S1159.

17.Maguire, P. Z., Struthers, A., Scott, D. M. & Paterson, W. R. (1998). *The use of agents both to represent and to implement process engineering models*, Computers & Chemical Engineering, **22**(10) S571-S578.

18.Marquardt, W. (1999). *From process simulation to lifecycle modeling*, Chemie Ingenieur Technik, **71**(10) 1119-1137.

19.McGreavy, C., Wang, X. Z., Lu, M. L., Zhang, M. & Yang, S. H. (1996). *Objects, agents and work flow modelling for concurrent engineering process design*, Computers & Chemical Engineering, **20**(12) S1167-S1172.

20.Struthers, A., Curwen, R. (1998). *A new approach to integrated process systems engineering - the VIBE Agent environment*, Computers & Chemical Engineering, **22**(10) S745-S748.

21.Westerberg, A. W., Subrahmanian, E., Reich, Y., Konda, S. & the *n-dim* team

(1997). *Designing the process design process.* Computers & Chemical Engineering, **21**(Suppl.) S1-S9.
22.Williams, G. W. (2000), *Multiobjective Process Synthesis and a Post-Synthesis Environment for Process Design*, PhD Thesis, University of Edinburgh.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 6.3: Web-Based Systems

I. T. Cameron & J. W. Ponton

6.4.1 INTRODUCTION

The world wide web (WWW) is a ubiquitous phenomenon, influencing the lives of all of us. We either use it for our email, purchase goods via web suppliers or do our banking on-line. It's almost e-everything!

Here in this chapter we look at the effect the web is having on engineering and CAPE, as well as look at past and current paradigms for web use and do some crystal ball gazing for the future. One thing is certain, opportunities for using the web are moving so quickly that comments on these issues are almost out-of-date before the latest webpage is mounted!

The chapter considers the basic infrastructure changes that have and are now taking place on the web including the future delivery of broadband capabilities which will permit activity not generally possible at present.

As well, we discuss and outline current trends in electronic, business and collaborative activities or E-, B- and C- activities. What is happening in these areas? Where is it going? What are the implications for CAPE? We also look at developments and opportunities in the area of CAPE education via web technologies.

One thing we can predict is that the world wide web will change the way we do business at the educational level and at the business level. If you are not seriously involved or actively planning your web-based business or education systems then opportunities may well pass you by.

6.4.2 THE PHENOMENA OF WWW

1994 doesn't seem that long ago in terms of time. It was the year that the World Wide Web came into existence through the work of Tim Berners-Lee and it has grown exponentially since that time. Not that this was the start of serious information sharing and computer networking. That dates back to the US government ARPAnet initiative in the 1960's with the subsequent development of the TCP/IP network protocol which is dominant today.

Growth of activity on the WWW has been exponential as measured by host numbers which have grown rapidly from several hosts in 1981 to over 93 million in 2000 [16], as seen in Figure 1. Growth has almost doubled each year since the early days of the internet. This growth in computers connected to the internet is also matched by people connected or having regular access to the internet [17]. Figure 2a shows the percent linguistic distribution on-line for 2000 and the projected percentages in 2005. The internet population will grow from 400 million to 1.15 billion over that time with non-english speakers dominating the linguistic population.

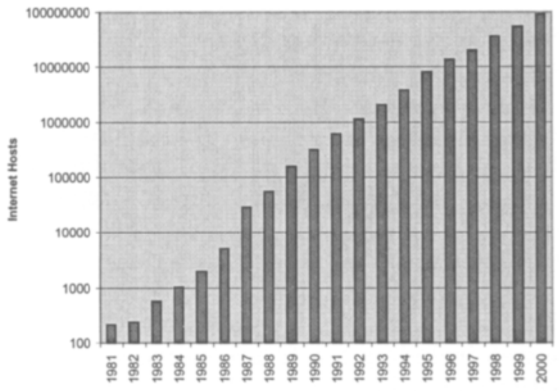
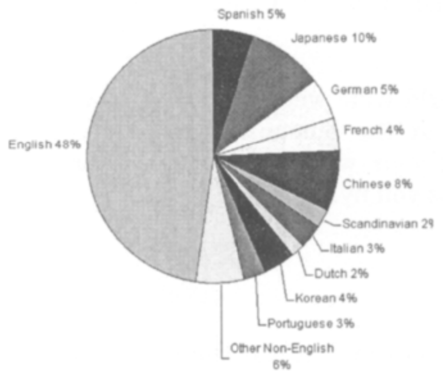


Figure 1. Internet Hosts



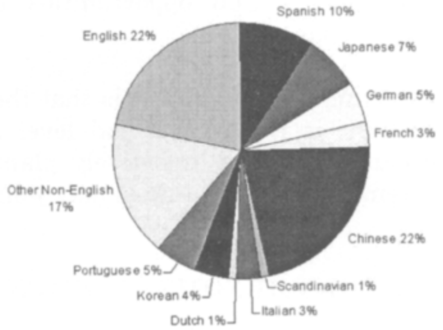


Figure 2a Linguistic Distribution On-Line for 2000 *Figure 2b Projected Percentages in 2005*

This represents a significant change in the internet population as well as a challenge for those providing services to such a wide and diverse group of users.

6.4.3 CURRENT ENABLING TECHNOLOGIES

The effective use of the web and its continuing growth relies heavily on a number of enabling infrastructure technologies and related developments. These are both hardware and software systems. Of clear importance to increased web-based engineering systems is the ability to handle large amounts of digital traffic. This increased bandwidth is being provided by current developments and rollout of digital subscriber line (DSL) and related technologies.

6.4.3.1 DSL Technology

There are several flavours of DSL technology including:

- Asymmetric DSL (ADSL)
- Symmetric DSL (SDSL)
- Very high speed DSL (VDSL)

ADSL technology provides much higher download rates than upload, typically 0.7Mbits/s versus 0.25Mbits/s, whilst SDSL provides equivalent rates in both directions at rates up to 2.3Mbits/s. These broadband technologies are now being deployed in the consumer market. SDSL is clearly aimed at the business market as will VDSL (60Mbits/s) in the future. The advantages of these technologies will be to provide capabilities for more extensive use of content-rich applications over the internet. It will directly impact on the ability to deliver CAPE technologies via the internet. Testing and roll-out of these technologies is now underway in Europe, the USA and the Asia-Pacific region.

The days of a 33.6 or 56kbits/s modem are fast vanishing.

6.4.3.2 XML

The development and use of eXtensible Markup language (XML) is having a significant impact on web-based systems, especially upon electronic document or data interchange (EDI). This coupled to powerful SQL servers such as MySQL or MS SQL provide much of the needed infrastructure for developing powerful web applications.

XML has significant advantages over standard HTML since it provides application control over the content of the document. Users of XML can define the protocols at each end of a transaction with data transfer handled via XML. It thus provides a standard that links web and database systems into a unified framework.

In the CAPE area, XML has been used to set a standard such as *pdXML* [8] for exchange of process engineering data. Similarly, *aecXML* provides a standard to communicate information related to the architectural, engineering and construction industries. The advantage of XML should be an increased willingness for companies and educational organizations to develop improved electronic interoperability. It should help drive down inter-company costs and help open markets.

6.4.3.3 Java Applets

With the increasing power of personal computers and workstations, it has become feasible to run very extensive computations locally even if their implementation is not particularly efficient. Java was designed to be platform independent, and, despite the efforts of some major software companies, it effectively remains so. The potential efficiency disadvantage of it being at least partly interpreted rather than fully compiled is, for most purposes, balanced by the increased speed of low cost computers.

Java provides elegant and relatively easy to use graphics capabilities, as well as being highly interactive. Its main disadvantage is likely to be the need to completely rewrite, and probably redesign, any existing code in Java and as an Applet.

However, much legacy code is probably due for rewriting in the interests of transparency and security, and at least as regards educational software, the size of the task may tend to be overestimated. Java is a better language for development of complex programs than the early Fortran in which most existing simulation programs were written and the rewriting of well-defined numerical procedures in it can be quite rapid.

6.4.3.4 JavaScript

Both Java and JavaScript both run on the user's (client) computer, but while Java is partly compiled, JavaScript is always interpreted and is thus seen by many users as `inefficient'. In fact JavaScript is a powerful and flexible language. Its syntax is similar to Java, but its philosophy is quite different.

JavaScript provides many useful techniques for making the web browser interactive, its most common (mis)use being to pop up subsidiary advertising windows on commercial web sites. However, it is also a useful general purpose language for any web application which is not too computationally intensive. JavaScript can be used to apply any Java public method. This means that the interface to a Java applet may be written in JavaScript; because the latter is fully interpreted, it can be a faster way of developing such interfaces.

6.4.3.5 Server programs

In principle a web browser can run any program that is on a server or is accessible to a server using the WWW Common Gateway Interface (CGI). This provides a means of making existing programs available over the WWW with minimum reprogramming. Older `batch mode' programs are particularly easy to adapt to this method of operation and may require no modification to the existing code at all.

The user is presented with a form in which he supplies the input data to the program. Submitting the form causes a program to be run in the CGI area of the server. This could in principle be the main program, but it will usually be more convenient to run a short program or script, which converts the user's form input into whatever the main program requires, and then causes it to be run. The main program can run either on the server itself or on another machine on a network accessible to the server (but not necessarily to the WWW). Similarly, the script can reformat the main program output into HTML for display on a web page, and send it back to the user. Alternatively, the output can be emailed to the user.

6.4.4 B2 -WHAT?

The use of the WWW is now in its third generation. First was the simple, static provision of information from web sites. Second were the business-to-customer (B2C) systems as exemplified by a*mazon.com* and other on-line customer oriented systems. That generation has matured and has passed. Now is the emerging generation of business-to-business (B2B) and its variants such as business-to-multiple businesses (B2MB) and B2everyB.

Worldwide B2B e-commerce is anticipated to be worth some $US7.3 trillion dollars by 2004 [15]. Most of this activity is driven by manufacturing and commercial industries. Supply chain issues will inevitably drive further expansion of the B2B or B2MB systems. A major example is the newly formed chemicals E-Marketplace ELEMICA sponsored by some of the world's largest chemical manufacturers [19].

CAPE tools are certainly in-line for exploitation within the B2B business model currently evolving. However the evidence shows that few systems are being made available online at this time. In the next section we consider the key areas of E-engineering specifically related to CAPE technologies and highlight what is being provided.

6.4.5 e-ENGINEERING

6.4.5.1 Engineering Portals

A portal is simply a "one-stop shop" website that provides a comprehensive set of services and information to a particular audience. The portal is a gateway into specific content. Yahoo! provides the best known of these web systems, however it is now an important development for process engineering business applications as seen in *myplant.com* which is shown in Figure 3. Other portals such as *processcity.com* (see Figure 4) are beginning to emerge with the purpose of providing news, services, on-line computation and discussion for the process industries. These sites will continue to grow, providing a useful gateway into process specific information.

Figure 3. Portal: myplant.com

Figure 4. Portal: processcity.com

We are now seeing many application service providers (ASPs) emerging in the engineering area. For example *Engineering-e.com* provides a marketplace for a range of engineering services, one of the most important being the *Simulation Center* that provides access to on-line computational fluid dynamics (CFD) and structural finite element applications. Several other players such as *e-CFD* and Moldflow's *plasticzone.com* provide internet CFD facilities based on fixed period, unlimited time rentals or on CPU hour computation times. Others are available. Making these work well needs appropriate software architectures.

6.4.5.2 Clients – Fat and Thin

Most new web-based engineering applications adopt architectures that range from thin clients to fat ones. A client is simply an application located on a local PC that uses a separate server for a variety of functions. Thin clients rely on a server to carry out the bulk of the data processing, whilst fat clients have the local PC doing the majority of the processing. There are advantages to each architecture depending on the application area.

The following sections outline some application areas that use various forms of clients for interactivity.

CFD applications

Most CFD and computational ASPs use thin-client architectures to provide input and display capabilities on the client machine, whilst server side activity concerns the computation of the specific CFD problem. The server side in these circumstances is usually accessing super-computer facilities to perform computations.

Properties prediction

In the CAPE area, a growing number of interactive internet applications are becoming available such as on-line calculations of physical properties, as illustrated by Aspen WebProperties Express under *processcity.com.* In this case a thin client provides data input and results display.

Hazard and risk assessment

In the area of risk assessment for land-use planning, web-based risk management tools are available for local government authorities to make impact assessments of hazardous scenarios via shortcut or detailed model estimates. Decision support facilities are also available. Such a system is seen in Figure 5 [9]. These are purely web-based server side applications with dynamic generation of web pages to the client.

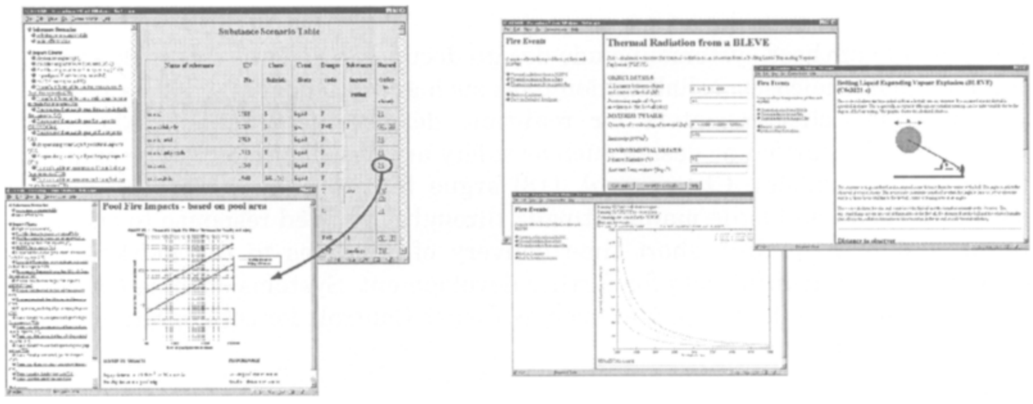

Substance Scenario Table
Pool Fire Impacts - based on pool area
Thermal Radiation from a BLEVE
Fire Events

Figure 5. Web-based risk assessment

Other applications include web-based expert systems for workplace hazards identification and assessment such as the *e-Compliance Assistance Tools* from OSHA [10].

Optimisation

On-line optimisation services are available through the use of a Java thin client that helps prepare problem specifications for submission to the NEOS server. This system provides a wide-ranging suite of optimisation tools for integer, linear and nonlinear programming problems [20]. Its use tends to be dominated by government and academic users, with a smaller number of commercial users.

Most CAPE tool vendors have not moved from monolithic code, distributed via CD or web downloads, towards architectures that provide web-based engineering capabilities. Thin client architectures provide a means to generate these web-based solutions, with little major rewriting of the server side CAPE software. This is a likely scenario given past investment in major coding (legacy code), the cost of re-implementation and the hesitancy of some major companies to fully embrace web-based systems. We are yet to see a wider range of significant web-based tools such as process design, modelling and simulation applications becoming available.

6.4.6 e-Collaboration

One of the rapidly growing areas of web-based applications is in the area of collaborative systems [11,12]. Here formal design methods and tools are used by geographically separated design teams to work on product development. In many cases web browsers act as the interface.

Components of these systems include design documents, containers such as filing cabinets and briefcases, collaboration tools such as whiteboards, address books as well as relay chat repeaters for real-time designer interaction. Other tools relevant to the particular design such as safety analysis methods can be available to all team members. Cheng *et al.* [13] argue that e-collaboration can greatly assist in the area of agile manufacturing through improved response to customer needs and high quality, short time delivery of solutions to the market. The design area is a fruitful field for further development. Systems such as *OneSpace* [14] are being used by companies such as Fisher Controls for collaborative design with significant impacts.

One of the current limitations for effective collaboration is internet bandwidth, especially where 3D designs are being developed. New software developments are addressing the data transfer issue to make real-time collaboration on such projects a reality.

6.4.7 Engineering Education in CAPE

The WWW offers opportunities to make education in a technical subject both more effective and more widely available. As a minimum, WWW technology enables a course lecturer (say) to make all his material immediately available at any time and place to his class, or indeed to anyone whom he wishes to allow access. At the other end of the scale of complexity, students can be given complex 'virtual worlds' with interactive multimedia presentations.

Major companies such as DOW [18] are embracing web technology to deliver on-the-job training to their workforce with significant cost savings and improved learning outcomes.

6.4.7.1 Overview of Possibilities

Simple Things

As soon as we first encountered the WWW and the early Mosaic browser, sometime in the early 90's, we saw it as a way of making lecture slides, already at that time prepared on a computer, available in full to the class immediately after a lecture. This remains a simple and sensible use of web technology. Interestingly, however, this model has now come full circle; our tendency now is to prepare web material in the first instance, in some cases presenting the lecture material from a web browser through a computer and data projector. Hard copy material is now usually the last thing to be produced.

Access to data

All of the general facilities of the WWW are of course available for teaching. Amongst the most useful are the range of data sources and search facilities. Process engineers regularly require information about chemicals, their physical and chemical properties, hazards, *etc.* Much of this information is now available online

For educational purposes one of the most useful `chemical' sites is NIST [2], the US National Institute of Standards and Technology. This provides a database of physical and chemical properties such boiling points, enthalpies of formation, etc. which, given the availability of a Web connected computer, can be accessed by students faster than finding and looking up practically any reference text.

6.4.7.2 On line Computation and Simulation

The WWW can provide access to computational tools by a variety of methods. These are described briefly in the following sections along with examples of their use.

Server programs

Server side programs using existing large-scale CAPE tools can provide the basis for effective web-based applications. This approach has been used successfully to run the AspenPlus simulation engine remotely for students. No changes could, or needed to be made to the code, which for licensing reasons only ran on a computer which was not itself a web server, but which could be accessed by the server over the local network.

Java Applets

Use of Java applets can be extremely effective in delivering web-based solutions. For example, a dynamic distillation simulation that we use, took the equivalent of less than a month of full-time work by a masters level project student. Part of the interface is seen in Figure 6

More time is usually required to write the interactive `front end' of applet based simulations, but these would in any case require development. Our experience suggests that using JavaScript (see below) rather than Java may often speed development.

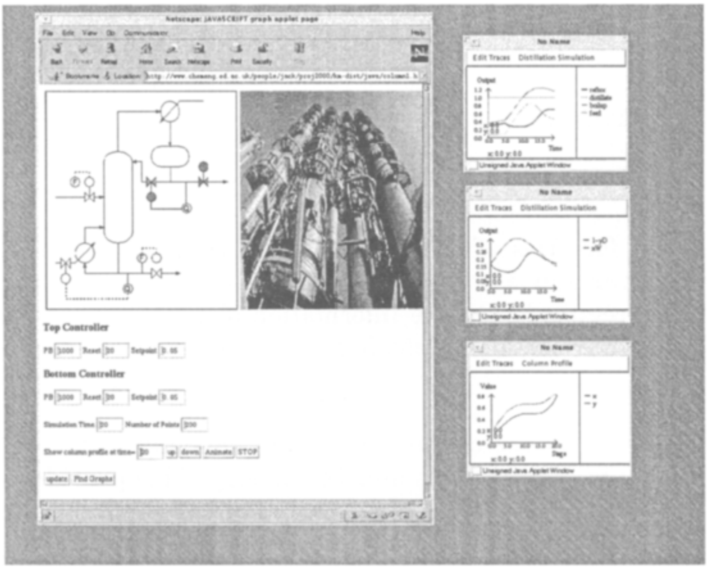


Figure 6. Dynamic Distillation Simulation

JavaScript

The simulations in the Control HyperCourse [3], discussed in section 6.4.8.1 are all implemented in this way. JavaScript provides a good way of implementing fairly short interactive programs. For the Process calculations course described below several interactive applications were implemented including a 'triangular diagram reader', an approximate vapour pressure calculator and a linear equation solver.

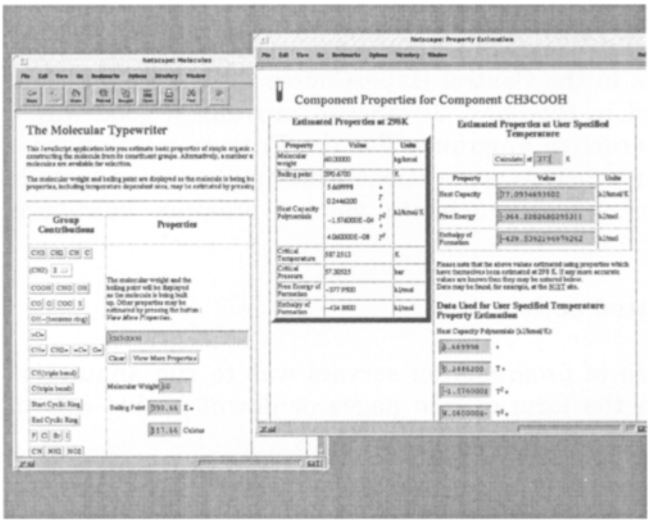
Server generated code

Our original use of programs on servers was to run simulations and generate their results in the form of web pages or downloadable information. Two less obvious applications follow from this.

Active JavaScript pages

Since JavaScript code is just a part of the web page text, it can be generated and sent by a server program, for example to provide results which can be manipulated interactively by the user.

One of the early teaching tools for the Process Calculations course, was a 'molecular typewriter' which implemented a simple group contribution method to estimate boiling points of organic species. This was extended to the estimation of other properties, for which Fortran programs already existed and rewriting was not desirable. The JavaScript code was augmented to send the structure of the molecule to the server, where the larger program was run to estimate, inter alia, standard free energies of formation and coefficients for a heat capacity polynomial. Originally these were simply returned to the user as a formatted web page. It was then realised that the server could easily write a page containing not only the estimated values, but JavaScript code to calculate heat capacities and free energies at a temperature specified interactively by the user. The final form of the expanded Molecular Typewriter is shown in Figure 7.



The Molecular Typewriter

This JavaScript application lets you estimate basic properties of simple organic [illegible] constructing the molecule from its constituent groups. Alternatively, a number [illegible] molecules are available for selection.

The molecular weight and boiling point are displayed as the molecule is being [illegible] properties, including temperature dependent ones, may be estimated by pressing [illegible]

Group Contributions	Properties
CH3 CH2 CH C (CH2) 2 COOH CHO OH CO C COC S OH–(benzene ring) =C= CH= CH2= =C= C= CH(triple bond) C(triple bond) Start Cyclic Ring End Cyclic Ring F Cl Br I CN NH2 NO2	The molecular weight and the boiling point will be displayed as the molecule is being built up. Other properties may be estimated by pressing the button : View More Properties. CH3COOH Clear View More Properties Molecular Weight 60 Boiling Point 390.66 K = 117.66 Celsius

Component Properties for Component CH3COOH

Estimated Properties at 298K

Property	Value	Units
Molecular weight	60.00000	kg/kmol
Boiling point	390.6700	K
Heat Capacity Polynomials	5.669998 + 0.2446200 T + −1.5760000E−04 T^2 + 4.0600000E−08 T^3	kJ/kmol/K
Critical Temperature	587.2513	K
Critical Pressure	57.30525	bar
Free Energy of Formation	−377.9500	kJ/mol
Enthalpy of Formation	−434.8800	kJ/mol

Estimated Properties at User Specified Temperature

Calculate at 373 K

Property	Value	Units
Heat Capacity	77.0934693502	kJ/kmol/K
Free Energy	-364.2282680295311	kJ/mol
Enthalpy of Formation	-429.5392196979262	kJ/mol

Please note that the above values estimated using properties which have themselves been estimated at 298 K. If any more accurate values are known then they may be entered below. Data may be found, for example, at the NIST site.

Data Used for User Specified Temperature Property Estimation

Heat Capacity Polynomials (kJ/kmol/K):

5.669998 +

0.2446200 T +

-1.576000E T^2 +

4.060000E- T^3 +

Figure 7. Molecular Typewriter

Downloadable code

In the HyperCourse [3] over the WWW, students needed to write and run simple dynamic simulations. This could have been implemented by providing a commercial dynamic simulator on a server, but we had access to no particularly suitable system, and also wished to avoid licensing complications. We also wished to provide something that could provide more immediate interactivity at run time than was available by the essentially batch mode of this approach.

The solution was to develop a simple simulation language based loosely on the syntax of gPROMS [1], and write a compiler to produce code for a target language which could be run on the user's own PC. The choice of target caused some problems; initial experiments were carried out with MathWorks popular Matlab system, but the cost of providing this on individual PCs was found to be excessive.

In the end, the solution was to generate spreadsheet code compatible with the major packages. This has proved very successful, as essentially all students already had a spreadsheet on their computers and were familiar with its use. It also meant that they could exploit all the useful features of the spreadsheet, such as graphing and customised report generation, without the need to provide them in the simulation system.

Three tools now work this way, the simple dynamic simulator, a linear equation solver (this being a feature not normally implemented in spreadsheets) and now a moderately sophisticated flowsheeting system. The last of these also takes the converged output of an AspenPlus model effectively turned into a linearised process model, and allows the user to carry out limited sensitivity analysis without having to access the whole AspenPlus package.

6.4.7.3 Multimedia Systems

Chemical engineering does not rank amongst the most graphics-orientated of disciplines. While mechanical engineers create 3 dimensional virtual motor cars, chemical engineers perform most of their work with boringly two dimensional flowsheets. Chemical engineering movies seldom go beyond the bubbling fluidised bed.

Architects and civil engineers provide virtual tours of buildings, but the corresponding facility for chemical engineers, the virtual plantwalk, while of undoubted use at the appropriate stage of plant design or for operator familiarisation, is of limited value in undergraduate education. The possibilities for web based interactive multimedia are potentially so rich that they must be worth exploring. However, chemical engineers have not taken very naturally to the exploitation of their possibilities.

A number of textbooks [4,5] are now issued with CDs featuring both audio-visual clips of plant (Felder and Rousseau, see also Virtual Plant section 6.4.7.6) and voice augmented interactive tutorials (Seider *et al*) both of which could be implemented over the web given the sort of bandwidth which is now becoming available.

Seider *et al.* give an excellent introduction to the mechanics of using flowsheeting packages, which is essentially hypertext plus voice. If used carefully, this approach could significantly enhance the use of hypertext alone, and add very much less bandwidth than video.

6.4.7.4 Virtual Laboratories

We have developed some rather limited applications in the area of `virtual laboratories'. The intention of this is to provide something of the flavour of the department's laboratories for online external users. To do this one can construct simulation models of items of and set up web pages photographs, videos or sound files to augment the simulation.

6.4.7.5 Real Laboratories Online

One chemical engineering department has put actual laboratory experiments online via the WWW [6]. The hardware required to do this is no more than would be required to attach a control computer in the usual way, provided the computer has WWW access. Although individual PCs on the web are usually `clients', making contact with hosts, which are `servers', the network does not make this distinction, and any connected machine can in fact be a server.

The software effort required to implement such a system is substantial. And there are clearly issues of safety and security in giving the world access to one's laboratory hardware! Although this is a very impressive example of what can be done with the web, our own instinct is to endeavour to make simulations more realistic, and more immediate, for example by adding sound and video clips.

6.4.7.6 Virtual Plant

The obvious next step from a virtual laboratory is a complete virtual plant. Early attempts have been made to reproduce a process operator training rig at a local chemical company which used to be the subject of an annual visit by students. This was a very simple 'process', which pumped water around, heated and cooled it and provided the company's apprentice operators with experience of basic instruments and plant operations. However, it was closed down as an economy measure.

A basic simulation of this was augmented with flowsheet, P and I diagrams, 3D layout in VRML, and a series of pictures. It was never used because the simulation of such a simple process was felt to be too uninteresting. Much of the point of simulation is to do what cannot easily be done in reality, and so a 'virtual plant' should really have complex chemistry and interesting separations. Furthermore the audio-visual effects now available are superior to those implemented at the time.

The virtual plant idea will be revisited making the process a more interesting if not a completely real one. We have recently developed a simplified but realistic dynamic simulation of both binary and multicomponent distillation that runs surprisingly fast as a Java applet on the current PC hardware. This will form part of the virtual plant that will also incorporate audio and video features. The nonideal models are sufficiently rigorous to illustrate for example distillation boundary crossings in a 3-component mixture, illustrating a column, which starts up containing feed from one side of the distillation boundary, and ends up with a product on the other side.

6.4.8 CASE STUDIES IN WEB-BASED EDUCATION

The following sections briefly describe experience with courses from United Kingdom second year undergraduate level to Masters level. Only one of these is intended as a fully web based course. In most the web material is an adjunct to a greater or lesser amount of conventional teaching.

6.4.8.1 The Control HyperCourse and Distance Learning MSc

The Control HyperCourse [3] was developed originally as supplementary material to a chemical engineering undergraduate course where teaching of process control was spread over all of years 2 through to 4 and 5. This was the first major attempt at creating web based teaching material, and our approach developed considerably as we put the course together.

The material intended for undergraduates in year 2 was the first module to be developed. This consisted largely of simply turning lecturers' notes into HTML. However, we conceived the idea of a Virtual Control Laboratory, which included a simulation of a `hardware' laboratory experiment. The purpose of this module was to introduce basic qualitative ideas about control and instrumentation. The second module, intended for year 3 undergraduates, is in two parts. One part deals with essentially qualitative issues in placement of control loops, such as the existence or not of a cause-and-effect between a proposed measurement and an adjustment. Originally a formal degrees of freedom analysis was presented; more recently we have realised that this can be replaced by the observation that the maximum number of possible control loops is equal to the maximum possible number of control valves which in turn equals the number of lines or streams on the flowsheet!

The second part of this module is about controller tuning and simple input-output models. This uses a number of tuning exercises in the Virtual laboratory. The lecture material accompanying this course is also prepared in HTML, which the lecturer displays by running a browser in a laptop, connected to a data projector. This includes a `whiteboard' that allows the user to drag and drop controllers into different configurations on a flowsheet (Figure 8).

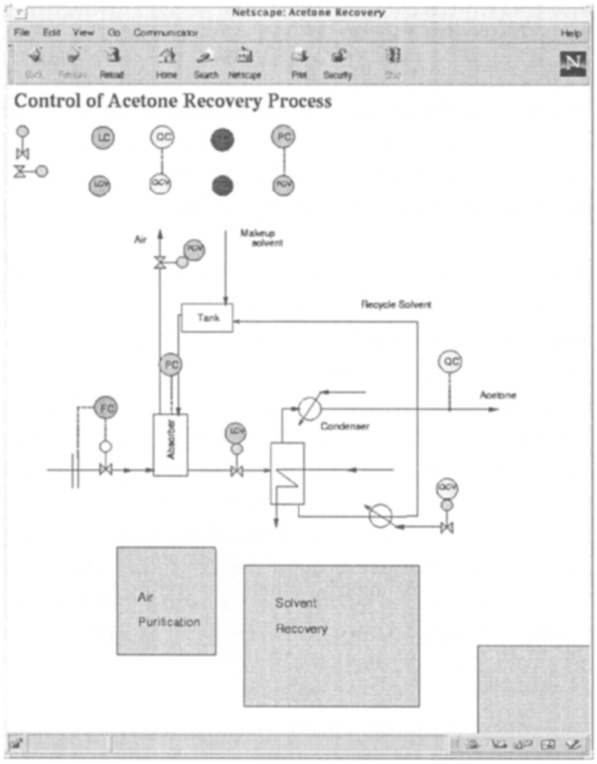


Figure 8. Control Loop Drag and Drop

Students taking this course are given a copy of the lecturer's material and the relevant section of the online course, including the Virtual Lab applets, as a self-extracting zip file so that they can use it on their own computer without a web connection. The increasing number of students who have their own computers has very positively received this.

The material for year 4 students is about the placement of control loops on complete processes, using a hierarchical approach [8]. This is illustrated by a series of examples, two of which are worked through in the web pages, and one of which has interactive question-and-answer material (written in JavaScript) which provides critical comment on the student's responses.

The final module that is still being extended contains optional material taken by year 4, 5 or graduate programme students. This includes material on frequency response methods illustrated by simulations and Virtual Lab exercises. We have found this to be a more effective way of introducing the concept than the traditional analytical approach. There is also material on multiloop controllers with corresponding simulations. A final exercise involves tuning the controllers on a simulated binary distillation column, which provides an interesting contrast, and supplement to earlier tuning exercises using very simple or idealised model processes.

Essentially the same material is used in a distance learning MSc module aimed at graduates in industry. When taken in this mode year 2 material is designated as revision, and all other material as new. Students work in their own time through a series of exercises, submitting their solutions by email, in some cases the submission being automatically generated by the Virtual lab server. Since many of the students are working at home through dial up lines and slow modems we have redesigned the material to minimize the need for a server, and now provide the students with a self contained version of the course on disc or CD.

Three cohorts of graduate students have now taken this module and although we have modified some parts of it in the light of experience it has in general been very well received.

6.4.8.2 Process Calculations

By contrast to the control course, which was a planned major exercise, covering a complete topic, a range of small tools and aids has gradually evolved to support one of our year 2 courses called Process Calculations. This is essentially a material balances course, but seeks to draw together other topics already taken by the students, in particular phase equilibrium separation processes.

The course is a conventional lecture course, and so all the material developed was intended to supplement rather than replace the usual lectures and problem classes. The exception to this is the online AspenPlus material. We now no longer give lectures to students on the use of flowsheeting packages. The same material can be used by faculty members who have never learned to use such systems but wish to keep up with their design project students.

As well as having all lecture notes and slides, problem sheets and, eventually, solutions, available on line, the course home page provides:

- Links to a range of data sources such as NIST.
- Self-instruction material for AspenPlus. (After seeing the more sophisticated material of Seider *et al* we would consider adding spoken commentary to this.)
- Links to property estimation methods and thermodynamic calculations (see Figure 8 above).
- Simple calculators for vapour pressure, humidity etc.
- General-purpose calculators for solving linear and nonlinear equations.

Students are encouraged to use these to supplement and check conventional data sources and hand calculations. These have been added to over the three years that this course has been taught. Students have taken well to using the web both for calculations and as a data source. They quite regularly come up with information sources of which the course tutors were unaware.

Developing simple 'value added' material for use in supplementing a conventional lecture course is relatively painless compared with the effort required to produce a complete course or module.

6.4.8.3 The Design Project

Our fourth year process design project is a group activity carried out by about ten students over the equivalent of a semester. Group working, self-organisation and the production of interim reports and working documents is seen as an important part of this project. The WWW provides a powerful vehicle for collaborative working by groups and is already being used in industry to manage design teams, sometimes working round the clock worldwide.

Over the last two years we have encouraged our students to use the web to help manage their own projects. It was intended that the initiative for developing this should come from the students, and all that was done was to set up a skeleton web page with the design brief and a list of the group and their email addresses, but with links to subpages for:

- Collections of chemical properties, hazard data, *etc.*
- Minutes of weekly project meetings.
- Interim and final reports.

Both the enthusiasm and aptitude of students for developing data pages and putting their ongoing and finished work on line has been variable. However, this mode of working has undoubtedly appealed to all of them and electronic communication and data presentation has become the norm for most design teams. Feedback suggests that this has helped the organisation of the project both for students and staff. For example part time industrial design tutors working with the groups can view current progress from outside the university. They can also be contacted by email with student queries at times other than those designated for their visits to the department, which has proved a mixed blessing from their standpoint!

6.4.9 FUTURE DIRECTIONS

The World Wide Web has become extremely pervasive throughout the academic world and is rapidly achieving a similar penetration in industry and commerce.

It has increasingly become a first, if not the first, data source consulted by students. Its utility as such is limited by the unselectivity of current search engines, and we would anticipate the development of both specialised and tailorable search facilities. In principle, any program on a web-connected computer can access the web and download information from it. One of the future developments we expect is the use of more selective and automatic web searches. For example, if the required properties of a substance are not available to a program, it can connect to the web, search for the information, find and verify it, and then continue the calculation using it.

The WWW will become the primary delivery vehicle for software. Already the downloading of software and updates from the web is probably the most common route for obtaining software which does not come installed on a PC. However, the more sophisticated mechanisms described above for running programs over the web or `just in time' loading of the specific applets or scripts required for an application will become more prevalent. The motivation for this is likely to be commercial; this is an easy way for vendors to charge customers on a `per use' basis.

The further development of engineering portals will see a greater range of interactive CAPE tools being made available to the process engineering community. Both steady state and dynamic simulation tools together with modelling and control design systems will begin to emerge as web-based systems. Bandwidth increases will enhance both academic and commercial organizations to work interactively and collaboratively with one another other. The problems supporting large monolithic simulation codes on multiple internal company machines or servers will help move the community in the direction of web-based delivery of solutions.

In education the advantages of this approach will be flexibility in providing exactly what students require, and no more. It will encourage students to use computer methods for short calculations when the required software is compact and loads quickly from a local server, rather than having to load a complete large flowsheeting package, most of which is not required. It will also be possible to monitor student use and progress as an aid to targeting support and providing feedback.

Web-based systems will provide incentive for further development of extended site learning experiences for students who spend a substantial part of their education working for a company. The ability to access and learn online will enhance the overall learning experience.

Increasing bandwidth will encourage the use of more graphic material and audiovisual presentations. The most effective use of this still has to be determined. Some uses of bandwidth, such as a video of a lecturer talking in front of a blackboard, are less useful than others.

6.4.10 REFERENCES

1. Internet Software Consortium, Internet Domain Survey, http://www.isc.org/
2. Global Reach, Languages on the I-net, http://www.glreach.com/
3. pdXML, http://www.pdxml.com, 2001
4. Foreshew, J., E-commerce takes digital silk road, The Australian IT, The Australian, NewsCorp, Dec 5, 2000.
5. ELEMICA, Online chemicals E-marketplace, Press release, http://www.basf.com/
6. Cameron, I.T., An interactive web-based decision support system for hazardous industry land-use planning, *Computers and Chemical Engineering* 24, 1057-1062, 2000.
7. Occupational Safety and Health Administration, e-Compliance Assistance Tools, http://www.osha-slc.gov/
8. Argonne National Laboratory, NEOS Server for Optimization, http://www-neos.mcs.anl.gov/
9. Brandenburgh, J. et al., A framework for low-overhead web-based collaborative systems, CSCW 98 Seattle, ACM 1-58113-009-0/98/11, 1998
10. Huang, G.Q. and K.L. Mak, Web-based collaborative conceptual design, J. Eng. Design, 10 (2), 183-194, 1999
11. Cheng, K. et al., The internet as a tool with application to agile manufacturing: a web-based engineering approach and its implementation issues, Int. J. Prod. Res., 38(12), 2743-2759, 2000
12. CoCreate, Collaboration software solutions, http://www.cocreate.com
13. Dow Chemical Company, Learn@Dow.online.
14. The National Institute of Standards and Technology, http://webbook.nist.gov/chemistry/
15. The ECOSSE Control HyperCourse http://ecosse.org/control/
16. Barton, P.I., "The Modelling and Simulation of Combined Discrete/Continuous Processes", PhD Thesis ICST, London 1992.
17. Felder, R.M. and Rousseau, R.W. "Elementary Principles of Chemical Processes", John Wiley, 2000.
18. Seider, W.D., Seader, J.D. and Lewin, D.R. "Process Design Principles", John Wiley, 1999.
19. The University of Tennessee at Chattanooga, Control Systems Lab, http://chem.engr.utc.edu/webres/Stations/controlslab.html
20. Ponton, J.W. and Laing, D.M., "A Hierarchical Approach to the Design of Process Control Systems", *ChERD*, 7, 181-188, 1993.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 6.4: Fault Diagnosis Methodologies for Process Operation

D. Leung & J. A. Romagnoli

6.4.1 INTRODUCTION

Today's petrochemical and mineral processes are far more sophisticated than those as of ten or twenty years ago in terms of process complexity. This is the result of a combination of factors. People are more aware of the environmental impact that process industries have on the environment and as a result governments all around the world are imposing increasingly stringent environment restrictions on gas emissions and waste disposal. If a company wants to stay competitive and profitable, its process has to be flexible enough to adapt to the fluctuation in demands of various chemical products of different purities. At the same time, the process has to be stable and controllable with "minimum" operating costs.

From an operational perspective, the increase in process complexity translates into a significant increase in activities involved in process operations and management. Process operational tasks can be classified into three levels according to the frequency of execution. The low-level tasks include data acquisition and regulatory control, which is normally automated by Distributed Control Systems (DCS) or the more advanced Supervisory Control and Data Acquisition systems (SCADAs). The medium-level tasks ensure the integrity of the operation, and they include process monitoring, fault administrations, optimization and other advanced control strategies. A majority of processes still relies on human personnel in performing these tasks. The high-level management tasks involve planning and scheduling. In most circumstances, it is the responsibility of the management team of the organization. Figure 1 illustrates a graphical representation of the above-mentioned task hierarchy. The pyramidal shape symbolizes the frequency of execution.

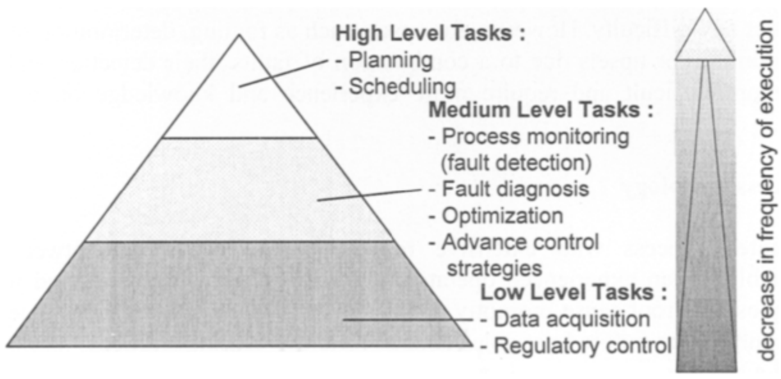


Figure 1: Task hierarchy pyramid

Among all the medium level tasks, fault detection and fault diagnosis can be considered as the two most influential tasks. When there is a process upset, the identified root causes can have adverse effect on the execution of optimization and other advance control strategies. Furthermore, depending on the severity of the identified root causes, they can also post a significant impact on planning and scheduling in management level. If the malfunction of a major process equipment is detected and confirmed, a medium or long term planning and scheduling may be required to compute a new production plan to meet the orders and production targets.

6.4.2 AUTOMATED FAULT DETECTION AND DIAGNOSIS

As chemical and mineral processes become more complex, they also become more vulnerable to process upsets and hardware failure. Fault diagnosis, therefore, becomes more important and difficult to perform. Although enormous effort has been put into designing better equipment and control systems, most of the fault diagnosis still heavily relies on plant operators. They are "expected" to respond accurately in locating the source of abnormalities and make quick and appropriate corrective actions. The difficulty of manual fault detection and diagnosis depends primarily on three factors:

1. Human knowledge and their mental status;
2. The nature of the anomaly; and
3. The topology of the process.

6.4.2.1 Human knowledge and Mental Status

The more knowledgeable and experienced the operator, the more accurate they can diagnose upset correctly within the shortest period of time. This well-accepted statement, however, does not take into account the mental status of the operators. Fatigue, emotional and personal matters can all affect their diagnostic capability dramatically.

6.4.2.2 Nature of anomaly

Single fault from a major piece of equipment, such as pumps, can be detected and diagnosed without any difficulty. However, for upsets such as fouling, deterioration of catalysts, valve failure, or upsets due to a combination of faults, their detection and diagnosis are more difficult and require more experience and knowledge on the operator's part..

6.4.2.3 Process Topology

For a complicated process with extensive recycles, and interactions between monitoring variables, even experienced operators and process engineers will find it difficult to diagnose the root cause promptly. In such a process, a fault can propagate its effect to a number of monitored variables almost instantly. During a process upset, it is rare to find a single alarm from the control systems. In reality, the control system will be flooded with tens of alarm messages. Either one of the two scenarios will happen. In the first scenario, the flooding of alarms will create additional mental stress on the operators. If they are inexperienced, this mental stress will lower their diagnostic capacity, resulting in false diagnosis and unnecessary panic. In the second scenario, the operators may choose to ignore the alarms by repeatedly clicking the "Acknowledge" button on the control system keyboards without paying attention to any of the alarms until all the alarm messages are acknowledged. In doing so, they ignore the process anomaly completely. This behavior often led to unnecessary downtime or even catastrophic accidents.

Automated fault diagnosis systems can help in ensuring that each and every operator will make fast and accurate decisions under all conditions, even the most abnormal ones. In the following sections, we will highlight some of the key fault detection and identification methodologies.

6.4.3 METHODOLOGIES

Industries begin to realize the importance of automated fault detection and diagnosis. Over the past decade, fault diagnosis receives enormous attention, and it is now one of the key research areas in process systems engineering. A series of technologies have been developed, ranging from purely quantitative to purely qualitative. Each of these methodologies has their pros and cons. A methodology that works well in one process may not give the same performance in another process. In selecting the right technology for fault diagnosis, one should take into account the plant topology, the sensitivity of the anomaly, and the complexity of the process equipment.

We classify fault detection and diagnosis techniques into the four following major categories:

1) Residual generations;
2) Statistical analysis; and
3) Knowledge-based (qualitative) approach.
4) Hybrid techniques

6.4.3.1 Residual generation

Analytical residual generation approaches use a series of mathematical equations to model the processes. These process model equations are used to calculate the residuals between sensor outputs and expected values, given a set of process parameters or operating conditions (Chow and Willsky, 1984; Frank 1990). If the discrepancy exceeds a certain threshold value, a fault is detected, and the corresponding process equipment or process section is being diagnosed as the root cause of the anomaly. The concept of fault detection and diagnosis can be represented by Figure 2.

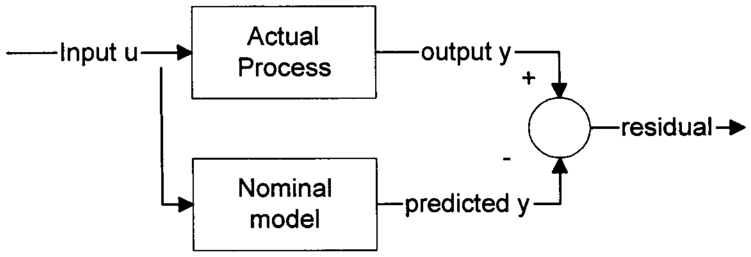


Figure 2: The conceptual representation of residual generation technique

Redundancy estimation can further be classified into two subclasses: residual due to redundancy and residual due to process parameters.

Residual due to redundancy

This approach makes use of the redundancy of the process. The redundancy can be either spatial or temporal. Spatial redundancy is a result of utilizing information from redundant sensors in process streams. This kind of redundancy is usually described by simple algebraic equations. Temporal redundancy exploit on the dynamic relationship between the input and output relationship of the process or the equipment. This type of redundancy is typically being described by differential or difference equations.

In general, the process can be described by the following linear discrete state equations:

$$\begin{aligned} \mathbf{x(k+1) = Ax(k) + Bu(k)} \\ \mathbf{y(k) = Cx(k)} \end{aligned} \qquad (1)$$

where **x** is the state vector (n*1), **u** is the input vector (p*1), **y** is the output vector (q*1), k is the time instant, **A**, **B** and **C** are known matrix describing the nominal process.

In regards to the modeling involved, there is no straight rule. Modeling can be done in many forms, as long as they are adequate to describe the input-output relationship of the process and its dynamics. Mechanistic models develop from first principal mass and energy balances or time-series models, frequency response or transfer function models obtained from historic plant data can all be used.

Residual due to process parameters

Redundancy residual approach uses a fixed nominal process model to calculate the predicted process output y. Residual due to process parameters, which is also known as parameter identification, takes the opposite approach. During a process upset, the fault(s) will change the physical properties of a system. The principle of parameter estimation is to detect and identify the fault by calculating the parameters of the mathematical model and the corresponding physical parameters and compare it against the nominal physical parameters.

The technique can be summarized by the following procedure (Isermann, 1984):

1. During offline calculation, choose a parametric model to represent the dynamics of the process. The parametric model can be a linear model with input/output differential relationship or it can be linear time series model. Let **A** be the model parameters.
2. Determine the relationship between the model parameters **A** and the physical parameters **P**:
$$\mathbf{A} = f(\mathbf{P}) \qquad (2)$$
3. During online execution, the model parameters $\mathbf{A}_{\text{real-time}}$ are being identified from input **u** and output **y** of the real process.
4. The real-time physical parameter, $P_{\text{real-time}}$ is being calculated by
$$\mathbf{P}_{\text{real-time}} = f^{-1}(\mathbf{A}_{\text{real-time}}) \qquad (3)$$
5. The deviation of real-time physical parameter $\mathbf{P}_{\text{real-time}}$ for its nominal value P is calculated.
$$\Delta\mathbf{P} = \mathbf{P}_{\text{real-time}} - \mathbf{P} \qquad (4)$$
6. A fault is detected if **ΔP** is greater than the threshold value, and the root cause of the fault is identified from the pattern of **ΔP**.

Figure 3 illustrates the above procedure.

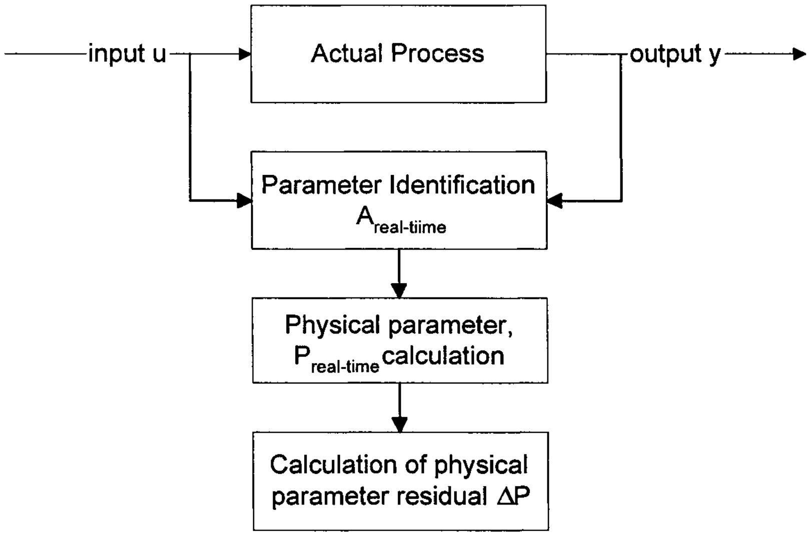

input u
Actual Process
output y
Parameter Identification
A real-tiime
Physical parameter, P real-time calculation
Calculation of physical parameter residual ΔP

Figure 3: Concept of parameter identification

The detection power of residual techniques depends on the accuracy of the mathematical models. This technique is sensitive for detecting upset at early development stage.

6.4.3.2 Statistical Analysis

For large scale industrial processes with numerous sensors and complicated process equipment, it is difficult to develop residual generation fault detection and identification systems, not to mention that it is also extremely computational expensive to execute online.

In statistical analysis, statistical information of the variable of interests is being used for fault detection and identification. This approach is considered to be completely "data-driven". Instead of using pre-determined process models, statistical models are built from normal operating data. Statistical thresholds are defined and they are used to distinguish the random process fluctuation due to noise from significant process deviation due to major disturbances and process upsets. Statistical analysis is commonly known as statistical process control (SPC). If the statistics of the interested variables are within the threshold values, the process is said to be in statistical control. When a process upset occurs, the upset will cause a directional deviation of the real-time statistics from its normal operating value, and eventually, the value will exceed pre-defined statistical thresholds. The process is not in statistical control and a fault is detected.

Statistical techniques can further be categorized into univariate techniques and multivariate techniques.

Univariate statistical process control

Univariate SPC techniques perform statistical tests on one process variable at a time. SPC fault detection is carried out through various statistical control charts. They include Shewhart charts (Shewhart, 1931), exponentially weighted moving average charts, EWMA and cumulative sum charts, CUSUM, (Woodward and Goldsmith, 1964).

Shewhart Charts: Shewhart charts are plots of real-time process variable x. When a number of observations can be recorded simultaneously, as in the case of offline laboratory analysis, Shewhart charts are then plots of mean ($\bar{x}$), range (R) and standard deviation (S) of a data set of n observations. The statistical hypothesis is that the mean and standard deviation should remain the same as the mean μ and standard deviation σ of the normal operating data. Upper control limit (UCL) and lower control limit (LCL) are calculated by specifying the level of significance α. In case of plotting real-time process variable x, assuming x follows a normal distribution, and assuming the UCL and LCL cover 99.7% of the normal operating data, the UCL and LCL are defined as

$$\begin{aligned} \text{UCL} &= \mu + 3\sigma \\ \text{LCL} &= \mu - 3\sigma \end{aligned} \qquad (5)$$

For samples with a number of observations, n, the UCL and LCL for $\bar{x}$ are defined as:

$$\begin{aligned} &\text{UCL on } \bar{x}: \quad \bar{\bar{x}} + A\bar{R} \\ &\text{LCL on } \bar{x}: \quad \bar{\bar{x}} - A\bar{R} \end{aligned} \qquad (6)$$

where $\bar{\bar{x}}$ is the arithmetic mean of $\bar{x}$, and $\bar{R}$ is the arithmetic mean of R. The UCL and LCL for R are defined as:

$$\begin{aligned} &\text{UCL on R:} \quad D_1\bar{R} \\ &\text{LCL on R:} \quad D_2\bar{R} \end{aligned} \qquad (7)$$

Values of A, D_1 and D_2 can be obtained from statistical tables. Note that the values of $\mu \pm 3\sigma$ can be significantly different from $\bar{\bar{x}} \pm A\bar{R}$.

CUSUM Charts: CUSUM chart plots the cumulated statistics on a regular time basis. Common form of cumulated statistics include the monitored variable itself, its deviation from a reference value, its deviation from its expected value, and its successive difference. Let S_n be the cumulative sum at time n, and X is the statistics of interest, CUSUM can be described by the following equation:

$$S_n = X + S_{n-1} \qquad (8)$$

The objective of using CUSUM is to detect changes in monitoring statistics. Therefore, in using CUSUM charts, it is not our concern whether or not the cumulated sum of the statistics falls over a fixed UCL and LCL. The real concern is the slope or the deviation between successive data points. Due to this nature, the definition of control limits of CUSUM is not UCL and LCL. The control limit of CUSUM is expressed as an overlay mask. It determines the maximum statistically allowable deviation of the previous data points. If the previous points fall out of the mask, the process is said to be not in statistical control. It signifies a noticeable change in process dynamics due to major disturbance or fault is detected.

EWMA Chart: Exponential Weighted Moving Average (EWMA) chart is a weighted plot of statistics of process variable, usually the process variable x itself or the sample mean $\bar{x}$, by placing a weight w, $0 \le w \le 1$ on the most recent data point and a forgetting factor 1 – w on the last statistics.
Assuming the information to be plotted is **Z**, EWMA can be represented by the following formula:

$$Z^*_{n+1} = w * Z_{n+1} + (1-w) * Z^*_n \qquad (9)$$

where Z_{n+1} is the raw information at time (n+1), and Z^*_{n+1} is the EWMA information at time (n+1). For a special case where w = 1, EWMA will be the same as Shewhart statistics.

The UCL and LCL of EWMA can be calculated by:

$$\mathrm{UCL} = \mu + 3\delta * \left(\frac{w}{2-w}\right)^{0.5}$$

$$\mathrm{LCL} = \mu - 3\delta * \left(\frac{w}{2-w}\right)^{0.5} \qquad (10)$$

where μ is the mean of **Z** and δ is the standard deviation of **Z**.

Multivariate statistical process control (MSPC)

Univarate SPC only considers one variable at a time. In order to use this technique to monitor a process with many process variables, it is necessary to apply these statistics or control charts on each individual variable. For the testing hypothesis to be valid for these univariate statistics, one has to assume that these variables are not cross-correlated to each other. However, in reality, many of these variables are cross-correlated. A deviation of one variable will also cause deviations in other variables. Such a process is said to be multivariate. This variable-variable interaction invalidates the assumption in univariate SPC. Applying univariate statistics to a multivariate process will only over-define the control limits, and thus leading to a wrong definition of the statistical normal operating region (NOR). This is illustrated in Figure 4.

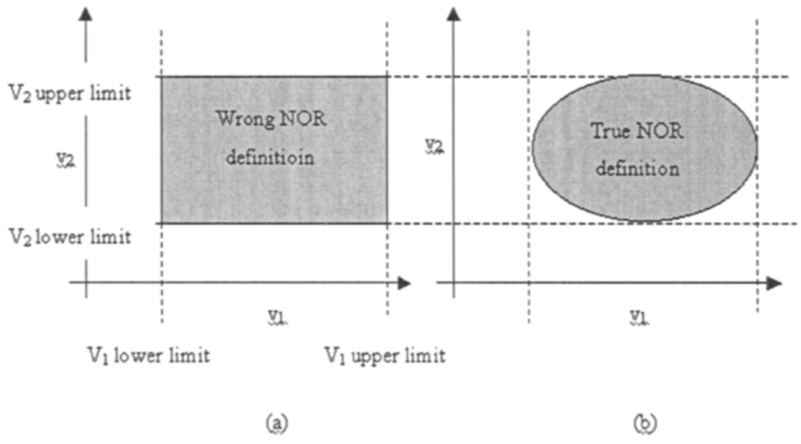


Figure 4: Representation of multivariate NOR a) using univariate approach (improper definition), b) using multivariate approach (true definition)

MSPC can overcome this problem by taking variable correlation into account. Multivariate projection-based approaches such as Principal Component Analysis (PCA) and Partial Least Square (PLS) are the most common and well-studied MSPC techniques (Wold *et al*, 1978; Jackson, 1991). They project all the process and quality variables onto an orthogonal state-space. In doing so, all the correlated variables are transformed into a new set of independent variables. These techniques can significantly reduce the dimensionality of the monitoring problem without a significant reduction in monitoring accuracy, and therefore they are especially suitable for simultaneously monitoring a large number of correlated variables (MacGregor and Kourti, 1995).

Principal components (PC) are based on a special property of matrix algebra. For a set of normal operating condition (NOC) training data $\mathbf{X}$ of n variables over m observations $(n \times m)$ with covariance matrix Σ:

$$\mathbf{P}'\Sigma\mathbf{P} = \lambda \qquad (11)$$

where the columns of $\mathbf{P}$, $p_1, p_2, \ldots, p_i$ are the principal component loading vectors $(m \times 1)$ or the eigenvectors of Σ, λ is a diagonal matrix and its elements $\lambda_1, \lambda_2, \ldots, \lambda_i$ are the latent roots or the eigenvalues of Σ.

The ith transformed variable, called principal component (PC) or score z_i, is a linear combination of $z_i = X * p_i$. Jackson (1991) outlined several criteria for selecting the number of PC, **A**, required to "successfully" describe a process.

The original data **X** can be reconstructed from PC by

$$X = z_1 * p_1^T + z_2 * p_2^T + \ldots + z_A * p_A^T + \mathbf{E} \qquad (12)$$

where **E** is the residual. If **A** is chosen correctly, there should not be any significant process information left in **E**, and the magnitude of **E** should be relatively small.

PCA can only deal with process variable data **X** that are measured on-line at the same frequency. There are also product quality variables **Y** being measured on-line at a lower frequency or off-line in laboratory. Partial Least Square (PLS) uses regression techniques to correlate **X** to **Y**.

Using the previous notation, over time t there are n process variables with m observations. Process variable matrix **X** has a dimension of $(n \times m)$. Let us assume that there are k quality variables with l observations over the same time t. This gives us a $(k \times l)$ quality variable matrix **Y**. In order to perform criss-cross regression on variable matrix X and Y, we have to extract a submatrix of X $(n \times l)$ such that the submatrices of both X and Y have the same number of rows.

Basic PLS algorithms have been well documented in chemometrics literature (Höskuldsson, 1988, 1992). The training of PLS involves sequentially calculation of the loading vectors for matrix X and Y such that they give maximal reduction in the covariance of the remaining X and Y matrices. PLS models can be represented by

$$\mathbf{X} = \sum_{a=1}^{A} \mathbf{t}_a \mathbf{p}_a' + \mathbf{E}$$

$$\mathbf{Y} = \sum \mathbf{u}_a \mathbf{q}_a' + \mathbf{F} \qquad (13)$$

where t_a and u_a are the ath latent variables or scores, and p_a and q_a are the ath x loading and y loading respectively. E and F are the residual matrices.

Similar to univariate SPC, MSPC monitoring is done through various control charts.

Score Plots: PCA has the ability to reduce the dimensionality of the process into just a few PC. In some processes with highly correlated process variables, they may be modeled "sufficiently" with two to three PC's. For a 2 PC's scenario, a 2-D PCA score plot of Z_1 vs. Z_2 is adequate to encapsulate the variability of the process. In the case of a 3 PC's scenario, either one 3-D PCA plot or three 2-D PCA score plots of Z_1 vs. Z_2, Z_2 vs. Z_3 and Z_1 vs. Z_3 are required. Score plots of PLS are similar to those of PCA. Q scores are calculated from P scores and therefore they are not plotted in score plots. Only P scores are plotted in PLS score plots. Unfortunately, it is obvious that score plots are not visually possible for processes with more than three PC.
In score plots, NOR are defined from the training data. By assuming normal distribution, NOR can be defined by an ellipsoid,. In reality, this may not be valid and a skew ellipsoid is more appropriate (MacGregor *et al.*, 1994). In some cases,

probability density function (PDF) can be calculated and applied to define NOR (Chen *et al.*, 1996; Martin and Morris, 1996). Figure 5 shows a comparison between two NOR of a score plot, one defined by ellipsoidal approach and the other defined by kernel PDF approach.

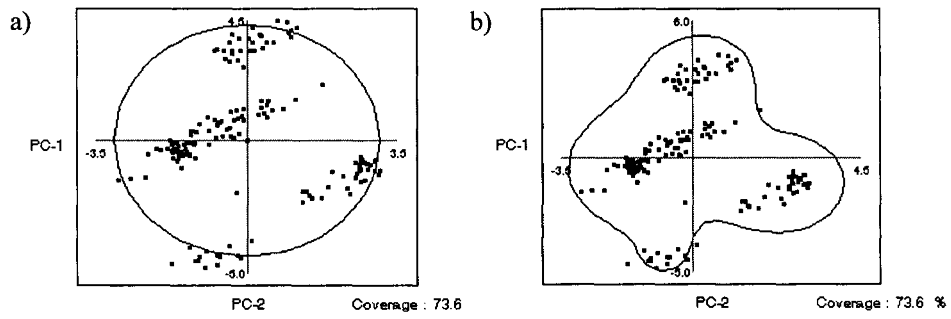


Figure 5: Comparison between a) ellipsoidal and b) Kernel PDF definitions

T^2 Plots: Another useful statistical chart is the T^2 plot, which is based on Hotelling's T^2 statistics (Hotelling, 1947):

$$T_A^2 = \sum_{i=1}^{A} \frac{z_i^2}{\lambda_i} \qquad (14)$$

It can be detected from T^2 whether the variation in **X** is within the space defined by the first A PC. The upper control limit (UCL) of T^2 can be approximated by F-distribution (Jackson 1991).

$$\mathrm{UCL} = \frac{p(A-1)}{(A-p)} F_\alpha(p, A-p) \qquad (15)$$

where p = the number training data points, A = the number of principal components used, and F_α is the F-distribution with p and n-p degree of freedom and significance level of α.

SPE Plots: If a new event occurs and if it is not present in the training data, the statistical projection model will not be adequate to describe the event. Consequently, a Type B outlier occurs. T^2 plot cannot detect the invalidity of the statistical model. A chart of the square predicted error (SPE) of the residuals could successfully detect this abnormal event.

$$\mathrm{SPE}_x = \sum_{i=1}^{A} (X_i - \hat{X}_i)^2 \qquad (16)$$

where $\hat{X}_i$ is the predicted variable set calculated from PCA or PLS model.

SPE follows Chi-squared distribution and its UCL can be calculated using Chi-squared approximated (Jackson, 1991; Nomikos and MacGregor 1995).

Statistical Fault Diagnosis

Various univarate and multivariate SPC charts are used for fault detection only.

Fault Clustering: Statistical fault diagnosis is primarily performed by clustering. Fault clusters are defined from faulty data of know root cause. Assuming that one has fault clusters of all root cause defined. During process upset, the real-time data point will move into one of the fault clusters. By performing test on which cluster the real-time data point is moving towards or position upon, root cause of the upset can be diagnosed. Mathematically, clustering can be performed on problems with any dimension, although it can only be presented graphically only on 2 and 3 dimensional space.

Fault clustering can be applied to original process variables. If the dimensionality of the process is high or the variables are highly correlated, fault clustering technique can be applied to PC scores of PCA or PLS. There are not many statistically clustering approaches available.

The simplest way to define the cluster is the angle of which the cluster locates relative to the axis and the NOR. Let $\mathbf{F_{rt}}$=(x, y) be the coordinate of the real-time faulty score in a PC score plot, and let $\mathbf{F_i}(x_{fi}, y_{fi})$ be the mean coordinate of a faulty data group i, the angle α between the faulty data and the faulty cluster can be determined by:

$$\mathbf{F_{rt}} \bullet \mathbf{F_i} = \left|\mathbf{F_{rt}}\right| \left|\mathbf{F_i}\right| \cos\alpha \qquad (17)$$

The simplicity of this approach makes it very appealing for online implementation. However, it is apparent that this method is insensitive to fault clusters with similar angles. Therefore this method is not recommended for processes with many root causes of similar symptoms.

A more advanced alternative is to define the cluster probabilistically by specifying a significance level. Clusters can be defined by assuming normal distribution or by calculating the PDF of the faulty data set. Assuming the monitored variables are independent, the probability of the fault cluster in n dimension $\mathbf{X_n}(x_1, x_2, \ldots\ldots x_n)$ can be defined by:

$$\mathbf{p(x_n)} = \frac{\mathbf{k}}{\sigma_1\sigma_2 \ldots \delta_n * (2\pi)^{n/2}}\left[\exp - \frac{1}{2}\left\{\frac{\delta_1^{\,2}}{\sigma_1^{\,2}} + \frac{\delta_2^{\,2}}{\sigma_2^{\,2}} + \ldots\ldots \frac{\delta_n^{\,2}}{\sigma_n^{\,2}}\right\}\right] \qquad (18)$$

where $\delta_n = x_n - \mu_n$, the deviation of the data point from its mean μ_n, and σ_n is the standard deviation. However, if the root cause affects two independent variables, it will be likely that the cluster is skewed with an angle. This can happen even if one applies PCA/PLS to correlated data. After the projection, a root cause, which only affects one process variable in X subspace, can affect more than one PC. This can be overcome by first finding the angle between the axis and the principle skewed axis of the fault cluster, and then perform a rotation of axis before the calculation of the cluster.

This rotation of axis can be avoided if PDF is used. One of the well-known estimation approaches is the kernel estimator technique (Chen *et al.*, 1996). The general formulation of kernel estimator is:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \qquad (19)$$

where $\hat{f}$ is the probability density, n is the number of data points, h is the window width of smoothing parameter. Many approaches are available to determine the smoothing parameter. The simplest is Scott Rule, which states that:

$$h = \frac{3.5 \times \delta}{\sqrt[3]{n}} \qquad (20)$$

The final kernel function K must satisfies the following condition:

$$\int_{-\infty}^{\infty} K(x)dx = 1 \qquad (21)$$

The choice of function K is arbitrary. For clustering and NOR definition on a two dimensional score plot using Gaussian distribution as the kernel function, the probability density function (PDF) becomes:

$$\hat{f}(x,y) = \frac{1}{2\pi n h_x h_y} \sum_{m=1}^{M} \sum_{i=1}^{n} \left[\exp - \frac{1}{2} \left\{ \frac{(x_i - x_m)^2}{h_x^2} + \frac{(y_i - y_m)^2}{h_y^2} \right\} \right] \qquad (22)$$

where (x_m, y_m) is the coordinate of the grip around that data clusters, and h_x and h_y are the smoothing parameters of the x- and y-axis. Like the normal distribution approach (equation 18), this approach can be extended to multi-dimensional problem.

Score and SPE Contribution Plot: In MSPC, when Hotelling's T_2 and SPE exceeds its UCL, a breakdown of the contribution from each original process variable x_i can help to diagnose the underlying anomaly. During a faulty condition, observing the deterioration pattern of PC scores and SPE contribution plot can identify the "suspicious" variables, which do not follow the projection model. These variables usually associate with events that cause the anomaly (MacGregor, 1998).

6.4.3.3 Knowledge-based (qualitative) approach

Knowledge-based (KB) approach employs qualitative modeling and reasoning as the fundamentals, although there are also new methodologies that incorporate probability and other mathematical techniques. As the name suggests, knowledge-based approach captures the knowledge of human experts. Knowledge can be classified into deep knowledge, which is knowledge from first principals, and shallow knowledge, which are heuristics that gathered from experience. Compared to residual generation and statistical analysis, KB is more suitable for fault diagnosis than fault detection.
Different KB based fault diagnosis approaches use different knowledge representation method to capture deep and shallow knowledge.

Hierarchical Knowledge Representation

Fault Tree Analysis: One of the earliest qualitative approaches is fault tree (FT) diagnosis (Himmelblau, 1978; Lapp and Power, 1977). FT arranges the knowledge about fault diagnosis in a hierarchical, inverted-tree fashion by using logic operators. The top of the FT is a node representing an upset or abnormal event. As it searches downwards, the detail of the knowledge regarding the nature or the location of the fault increases. This is equivalent to a backward-search through the topological structure of the plant from a coarser to a finer scale. The search will continue until the root cause of the upset event is identified. In between each level of the FT, logic AND and OR gates are used to represent how the lower level nodes can satisfy the high level objectives. Construction of fault tree can be time-consuming. There are automatic fault tree generation procedures available to speed up the knowledge representation process (Lapp and Power, 1977). Figure 6 shows the general structure of a tree analysis. Due to the inverted tree structure, probability can be incorporated into fault tree diagnosis without difficulty.

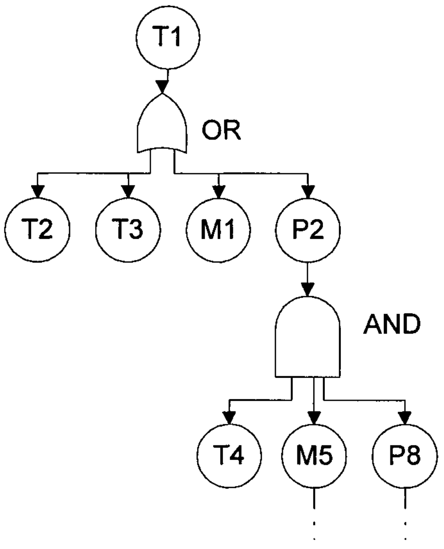


Figure 6: Structure of a typical fault tree

Goal Tree Success Tree: Goal Tree Success Tree (GTST) is another qualitative approach which uses hierarchical knowledge representation (Chen and Modarres, 1992). GTST uses goal-oriented knowledge representation. The tree can be divided into 2 parts, the goal tree (GT) and the success tree (ST). GT describes how the process goal can be achieved by various sub-goals, and ST summaries how the bottom goal can be achieved logically by different equipment, automatic and human actions. The logical structure of ST and FT shares many similarities. Figure 7 illustrates the general structure of a GTST.

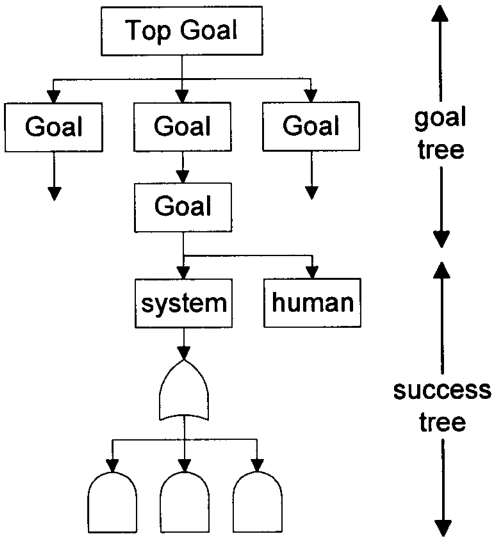


Figure 7: Typical structure of a Goal Tree Success Tree

In GTST methodology probability calculated onlineis used to determine which is the optimum branch that should be used to continue within the search through. The underlying principle is somewhat similar to a "best-first search", in which probability is used as the evaluation function.

Signed Direct Graph (SDG) Based Approach

Signed Directed Graph: Another early qualitative fault diagnosis methodology is Signed Directed Graph (SDG) model, which was proposed by Iri et al (1979). In this model, SDG is used to describe the qualitative cause-effect influences between various process variables and events. These causal relationships can be derived through deep knowledge mathematical models and plant and instrumentation diagrams (P&ID).

SDG can be described by a (**G**, s) pair. The signed directed graph, **G**, is made up of a number of nodes **N** and branches **B**. Nodes **N** correspond to state or process variables or fault origins such as equipment failure. Branches **B** are arcs that join two nodes together. The directions of the branches **B** are described by two incidence functions, i and j. A simple SDG and its representation are illustrated in Figure 8.

N = { n1, n2, n3 }
B = { b1, b2, b3 }

b_i	$i(b_i)$	$t(b_i)$
b1	n1	n2
b2	n2	n3
b3	n1	n3

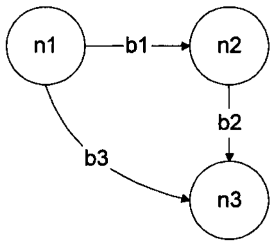


Figure 8: A simple SDG and its representation

Three states, namely high, normal and low, are used to describe the state of each node in SDG. Function s maps the direction of deviations between adjacent nodes to the set {+, -}. For instance,

$$A \xrightarrow{+} B$$

means deviation in A will induce deviation in B in the same direction, whereas

$$A \xrightarrow{-} B$$

implies deviation in A will cause opposite deviation in B.

SDG is an attractive technology because it has the ability to handle cyclic causal relationships and uncertainties, the capability of locating all possible causes and the ease of construction from first principle. Diagnosis using SDG can be done directly by implementing search algorithms to search the SDG, or by transforming SDG information into a rule-based systems (Kramar *et al.*, 1987).

Possible Cause and Effect Graph: Although SDG is a powerful technique, it has its own weaknesses. It restricts the state description statement of each node to one of the three symbols, namely high, normal and low. In SDG, all nodes that appear in the Cause and Effect (CE) graph could be the possible root cause of the abnormality. In many cases, this demerit makes the diagnosis results meaningless and hard to interpret (ambiguous outcomes).

Possible Cause and Effect Graph (PCEG) approach is a modified SDG approach which is proposed to overcome these weaknesses (Wilcox, 1994). In PCEG, the domain is divided into a set of partitions or subjects **S**, which is mutually exclusive. A set of abnormal statements **A** and normal statements **N** is defined. A subset **X** of **A** is defined as the exogenous causes (possible root causes). Each subject **s** will have one normal statement and at least one abnormal statement. Causal relationships **R** relate the cause-and-effect relationships between the abnormal statements of different subjects. Function **f** maps between subject s and its corresponding normal statement **n**. The family of all possible state of subject **s** is s_k+ f(s), with s_k being all the abnormal statements of subject S. Figure 9 illustrates the concept of PCEG model.

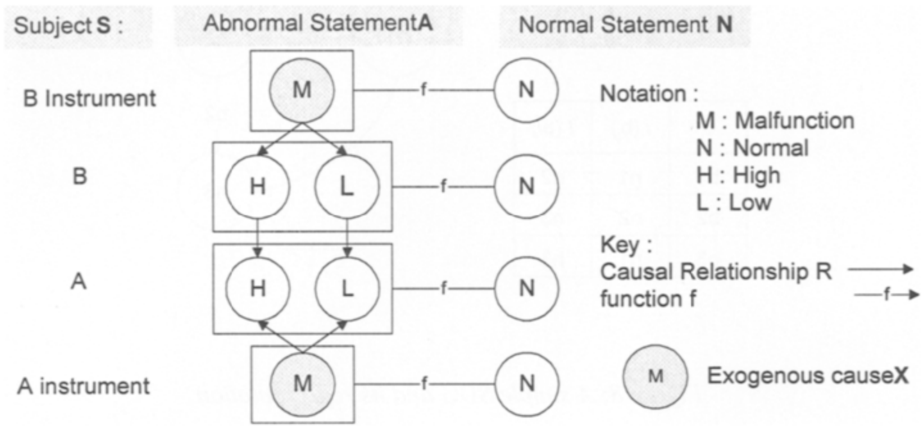


Figure 9: Illustration of PCEG concept

Instrument A, Instrument B, Variable A and Variable B are the subjects of the domain. The normal statement of the domain includes a "normal" node from each of these subjects, and they are represented as circles with letter "N" on the right hand side. The members of the abnormal statement of each subject are enclosed inside the rectangles in the diagram. Each of Instrument A and Instrument B has only one "abnormal" statement. They are the exogenous causes of the domain because they are the source nodes of the CE graph. They are denoted "M" for malfunction. Each of Variable A and Variable B has two "abnormal" statements, namely high and low. They are denoted as "H" and "L" respectively in the diagram. Depending on the application, high-high or low-low or other abnormal conditions can be added to the abnormal statement of the subjects. The arrows represented the causal relationship between two members of abnormal statements of different subjects. For example, Instrument A malfunction "can-cause" high value of Variable A.

In PCEG, more meaningful state description about the nodes can be used, making it user-friendlier both in terms of knowledge representation and state partitioning. Another major improvement is the distinct definition of exogenous (root) causes in the domain.

Probabilistic Possible Cause and Effect Graph: Unlike FT and GTST, the possibility of existence of multiplies connected nodes and cyclic loops within SDG and PCEG make online probability calculation difficult. Leung and Romagnoli (2000) proposed a novel approach to overcome this difficulty.
First, a Bayesian Belief Network, as illustrated in Figure 10, is incorporated into the PCEG methodology as the basis for probability calculation. A set of evidence is used to change the real-time conditional probability between parent and child nodes as follow:

If evidence is true

$$\text{Cond}_{\text{cal}}(V, U) = \text{Cond}_{\text{expert}}(V, U)$$

else

$$\text{Cond}_{\text{cal}}(V, U) = 1 - \text{Cond}_{\text{expert}}(V, U) \qquad (23)$$

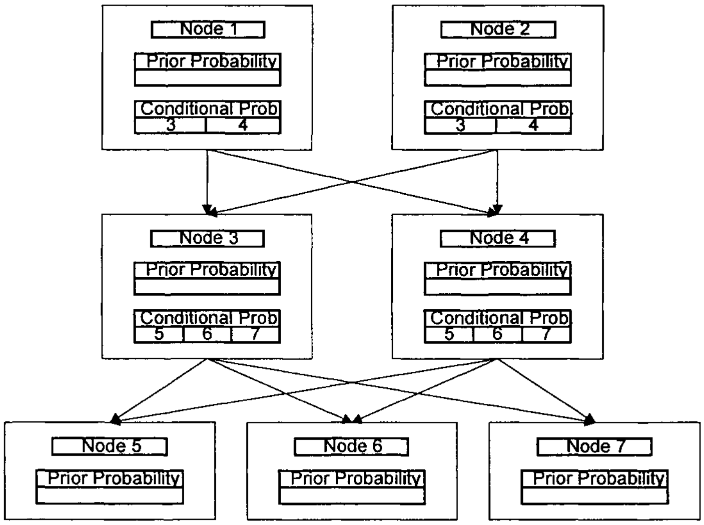


Figure 10: Simplified Bayesian Belief Network

In solving the difficulty of multiply connected nodes, special search mechanism which is mathematically equivalent to conditioning is used. Traditional Bayesian Belief Network does not support any existence of directed cycles (Russell and Norvig, 1995). A rule-based system is proposed to overcome this difficulty. The states of process variables that are not involved in the cyclic loops can sometimes provide vital information of the possible process upset scenarios. During manual diagnosis, human experts apply the same mental model to achieve accurate results. In this expert system, we apply the same conceptual model. A special set of "expert" rules is modeled to make on-line causality adjustment (arc deletion). These rules break the causality loops on-line, and turn the final PCEG into an acyclic network for fault diagnosis.

Time Delay Management: Alarms can be classified into root alarms and effect alarms, as illustrated in Figure 11. All fault diagnosis methodologies can respond to root and effect alarms. However not many methodologies can handle time-delayed alarms, especially when uncertainty of their occurrence has to be taken into account.

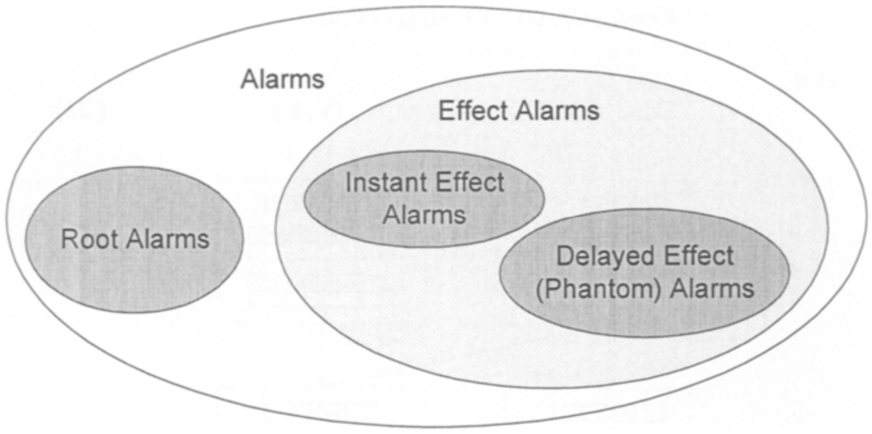


Figure 11: Alarm Classification

The uniqueness of the probabilistic PCEG approach also enables the system to handle delayed alarms, which is also known as phantom alarms effectively. As anomaly escalates, abnormal level upstream process variables can propagate to cause abnormal level of downstream process variables. For small processes, effect alarms are "virtually" instantaneous. For large complex process with time delays, effect alarms are "time-delayed". Abnormal level on one variable may take minutes or even hours to propagate to other downstream variables. Even if the upstream variables have been restored to normal operating range, time-delayed propagation will continue and cause "phantom alarms" downstream. Phantom alarms are "effect" alarms that only appear some time after the appearance and reparation of the "root" alarms due to time delay. Phantom alarms are considered as abnormal statement about the process. Using the same notation as above:

$$A=\{a_1,a_2,......,A_{S_n,S_m}\} \qquad (24)$$

A is a set of abnormal statements about the process at any time t. A_{S_n,S_m} is the complete list of the time delay abnormal statements, a_{S_n,S_m}, which is defined as the time delay abnormal statement that causes abnormal statement of subject S_m as a result of a historical abnormal statement of subject S_n.

$$A_{S_n,S_m} = \{a_{S_{n1},S_{M2}}, a_{S_{n2},S_{m2}}, ...\} \qquad (25)$$

The list is stochastic, i.e. it varies with time and process conditions. Phantom alarms can be the potential root causes of observed anomaly, and therefore they are also modeled as a subset of exogenous causes.

$$A_{S_n,S_m} \subseteq X \qquad (26)$$

There is a subset of the binary relation R, called the delay relation R_D. It defines the "delay can cause" relationship between the abnormal statement set **A**.

$$R_D \subseteq R \qquad (27)$$

Time delay management adopts the AI approach of dynamic belief network (DBN) management. A list of anticipating time delay exogenous causes for the future is computed every time fault diagnosis is run, and this list is updated into the network by appropriate node creation and deletion. The system prepares the belief network structurally for any future diagnosis.

Let n_a be all the active nodes at time t_1, and let n_i be all the inactive nodes. If a delay relationship R_D exists between n_a and n_i (n_a delay can cause n_i), then it can be concluded that there is a possibility of phantom alarm of n_i due to n_a.

$$\begin{gathered} n_a \subseteq N_{active},\ n_i \subseteq N_{inactive} \\ \forall n_a, n_i \quad R_D(n_a, n_i) \Rightarrow \text{phantom\_alarm}(n_i) \end{gathered} \qquad (28)$$

When the expert system identifies possible phantom alarms, it will dynamically append the possible time-delay nodes onto the original PCEG belief network, with each of them properly time-stamped (Primary Update).

When abnormality is detected in future time t_2 ($t_2 > t_1$), the expert system will execute a "secondary update" mechanism with the following rules:

1. Phantom alarm cause deletion when the variable causing the delay effect, n_a, is still in alarm mode;
2. Conditional probability revision when the variables which cause the phantom alarms are "normal"; and
3. Cause deletion when the predicted "phantom" alarms do not appear or their anticipated effects have been expired.

This "update" sequence prepares the phantom alarm section of the belief network with the latest real-time information regarding the process. After the completion of belief network updates, the probabilistic PCEG fault diagnosis algorithm will be executed. The overall interface for phantom alarm management is shown in Figure 12.

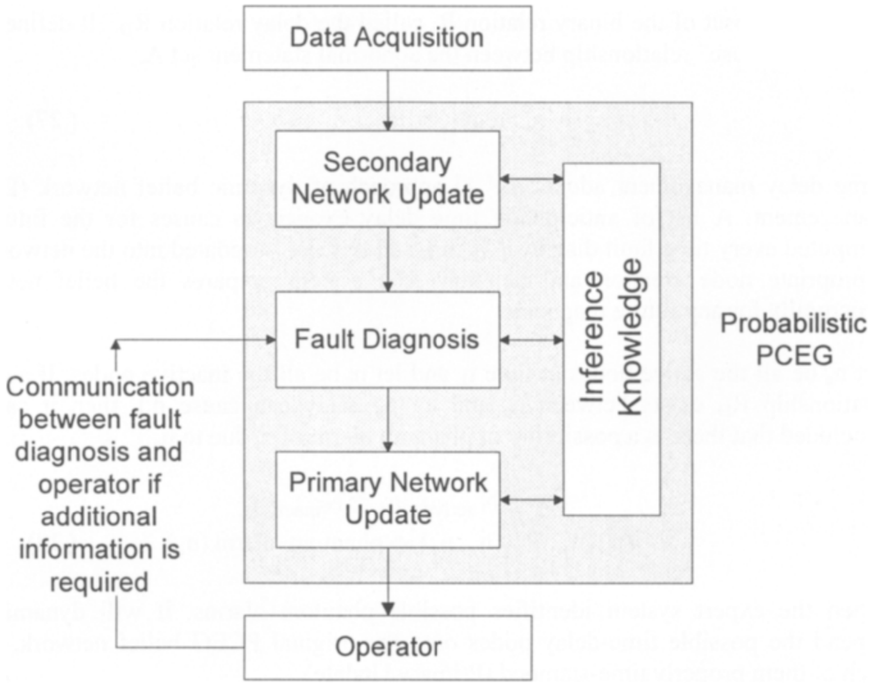


Figure 12: Overall inferences for dynamic time delay management

6.4.3.4 Hybrid Techniques

In the above sections, we have described a number of basic fault detection and diagnosis technologies. Researchers in fault detection and identification focus on fault identification algorithms and they usually overlook the importance of fault detection. In many cases such as all kb techniques, only simple univariate limit-checking technique is used as the means of fault detection. Using primitive univariate fault detection technique sufficiently reduces the sensitivity and accuracy of the fault diagnosis mechanisms.

To effectively coordinate between these two tasks, one should concentrate on planning a framework that allows flow and utilization of information from MSPC monitoring to fault administration. Research work in interfacing MSPC with fault diagnosis has taken off only about two years ago, and some initial successes have been documented in literature. Norvilas *et al.* (1998) interfaced MSPC KB fault diagnosis for monitoring-fault diagnosis purpose. In their work, they used MSPC to perform monitoring. If the MSPC statistics exceeds the predefined limits, the numeric MPSC information will be compressed into symbolic "high/low/normal" and the variable nodes in the fault diagnosis knowledge base will be activated/deactivated accordingly. Vedam and Venkatasubramanian (1998) used individual SPE to link the MSPC information to a SDG-based fault diagnosis module. In their work, they assume that fault will cause displacement in SPE plane of the PCA projection and the variables, which have abnormally high individual SPE, are identified as "alarm variables" in SDG fault diagnosis. Leung and Romagnoli (2000) use a KB to mimic how a human engineer would interpret various MSPC statistics contribution charts (MacGregor et al., 1994). In addition to this symbolic interpretation of various contribution plots and control charts, they also used real-time MSPC statistics as inputs into their probabilistic PCEG KB approach (Leung and Romagnoli, 2000) in order to increase confidence and accuracy of the probability calculation.

6.4.4 CONCLUSION

An overview on fault diagnosis methodologies for process operation has been given with a view to development of computer aided tools. First, different features of automated fault diagnosis has been defined and explained, followed by explanation of different types and classes of methodologies. We hope that the contents of this chapter will help the reader to design the corresponding software architecture for fault detection and diagnosis and to develop new state-of-the-art software.

6.4.5 REFERENCES

Chen, J., A. Bandoni, J. Romagnoli (1996). Robust PCA and normal region in multivariable statistical process monitoring, *AIChe Journal*, **42**, 12, 3563-3566

Chen L W, M Modarres (1992), Hierarchical Decision Process For Fault Administration, *Comput.chem.Engng*, **16**, 425-448

Chow, E.Y., A.S. Willsky (1984). Analytical redundancy and the design of robust Failure detection systems, *IEEE Trans. Automat.* Contr., AC-29, 7, 603-614

Frank, P.M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - a survey and some new results. *Automatica*, **26**, 3, 459-474

Himmelblau, D.M. (1978). *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*, Elsevier.

Höskuldsson, A. (1988), PLS Regression Method, *J. Chemometrics*, **2**, 211-228

Höskuldsson, A. (1992), Quadratic PLS Regression, *J. Chemometrics*, **6**, 307-334

Hotelling, H. (1947), Multivariate quality control, illustrated by the air testing of sample bombsights, *Techniques of Statistical Analysis*, 113-184. McGraw-Hill, New York.

Iri, M., K. Aoki, E. O'Shima, H. Matsuyama (1979). An algorithm for diagnosis of system failures in the chemical process. *Comput.chem.Engng.*, 3, 489-493

Jackson, J.E. (1991). *A User's Guide to Principal Components*, John Wiley and Sons, New York.

Kramer, M. A., B. L. Palowitch (1987). A rule-based approach to fault diagnosis using the signed digraph, *AICHE J.*, **33**, No.7, 1067-1078

Kresta, J.V., J.F. MacGregor, T.E. Marlin (1991). Multivariate Statistical Monitoring of Process Operating Performance. *Can. J. Chem. Eng.*, **69**, 35-47.

Lapp, S.A., G.A. Power (1977). Computer-aided synthesis of fault trees. *IEEE Trans. Reliability*, R-37, 2-13

Leung, D., J. Romagnoli. (2000) Dynamic Probabilistic Model-based Expert System for Fault Diagnosis, *Comput.chem.Engng.*, **24**, 2473-2492

Leung, D., J. Romagnoli (2000), A framework for full integration of multivariate statistical information into knowledge-based fault diagnosis, ADCHEM2000, Pisa, Italy

MacGregor, J.F., T. Kourti (1995). Statistical Process Control of Multivariate Processes. *Control. Eng. Practice*, **3**, 3, 403-414.

Norvilas, A., A. Negiz, J. DeCicco, A. Cinar (2000). Intelligent process monitoring by interfacing knowledge-based systems and multivariate statistical monitoring. Journal of Process Control, **10**, 341-350

Rich S H, V. Venkatasubramanian (1987), Model-Based Reasoning in Diagnostic Expert Systems for Chemical Process Plants, *Comput.chem.Engng*, **11**, 111-122

Russell, S., P. Norvig (1995), *Artificial Intelligence A Modern Approach*, Prentice Hall

Shewhart, W.A. (1931) *Economic Control of Quality of Manufactured Product*. Van Nostrand, Princeton, N.J.

Vedam, H. and V. Venkatasubramanian, Automated interpretation of PCA-based process monitoring and fault diagnosis using signed digraphs. In *Proceedings of 3rd IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries*, IFAC, Lyon, France

Wilcox, N.A., D. M. Himmelblau (1994a). The Possible Cause and Effect Graph (PCEG) Model For Fault Diagnosis-I. Methodology, *Comput.chem.Engng*, **18**, 103-116

Wilcox, N.A., D. M. Himmelblau (1994b). The Possible Cause and Effect Graph (PCEG) Model for Fault Diagnosis-II. Applications, *Comput.chem.Engng.*, **18**, 117-127

Wold, S., P. Geladi, K. Esbensen, J. Ohman (1987). Multiway principal components and PLS analysis. *J. Chemometrics*, **1**, 41-56.

Woodward, R.H., P.L. Goldsmith (1964). *Cumulative Sum Techniques*. Oliver and Boyd. London

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 6.5: Emerging Business Models

J. Köller, T. List, C. Quix, M. Schoop, M. Jarke, P. Edwards & M. Pons

6.5.1 INTRODUCTION

Information technology (IT) continues to be increasingly important as a critical area of business competency. In the US, estimated expenditures for IT are pegged at about 8% of revenue. This reflects the fact that IT expenditures continue their recent increased penetration of business activities. It is also the basis for the latest shift in the balance of competencies for enterprise leadership. These transforming benefits achieved through using IT-based technology will depend heavily on technical factors and on the willingness of a community to work together for the greater benefit of all.

This dependency can be seen in the standards area. Examples of the collaboration issues are illustrated by (1) the area of Java standards (where Microsoft appears to have given up on a contentious standards process - striking out with its own product called C#) and (2) the highly competitive area of XML standards (where Microsoft is providing an XML technology platform called BizTalk Server) and the Rosetta Net XML application standards consortium. The issue here is whether using Microsoft platform technology is a valid standards-based approach. In Microsoft's view, there is an XML language standard and industry groups working with an XML collaborative business network supported by Microsoft can be regarded as being engaged in developing conforming XML-based application standards for their particular industry. RosettaNet supporters don't agree.

An example of a major success of standards that will transform an industry can be found in a specialized area of computing used by the process industries. The particular computing domain is termed Computer Aided Process Engineering (CAPE). The industry had evolved to a point where there were three global major players plus a second group of smaller suppliers that participated in particular regions or in niche products/technologies. Overall success depended on achieving success in technology and collaboration. Technology success was based on standardizing interfaces for component software. This was combined with success in achieving collaboration between (1) the first and second tier of suppliers and (2) the major corporate users of the CAPE products and technologies. Currently, the newly evolved standards are being driven forward by an international coalition called Global CAPE-OPEN (GCO), which is operating under the auspices of the Intelligent Manufacturing Systems (IMS) organization. The EU, Japan, and the US are all working with the GCO project.

Industry breakthroughs such as GCO are being repeated hundredfold in the broader area of enterprise IT. These breakthroughs are based on hard work and vision. They will support the reengineering of inter-business processes that underly an increasingly complex activity of building and evolving IT infrastructure for medium to large enterprises (MLEs). The emerging IT environment is characterized by rapid change in 1) the technology, 2) the roles and relationships of the industry, and 3) expectations of the marketplace and by stakeholders who have invested in providers or consumers of IT technology.

New value-adding tools are being developed to enable processes in this enterprise IT environment, which is characterized by an explosion in complexity. In particular, it addresses the need to manage an amount of information about IT solutions and processes that is growing exponentially. This is true for IT in general and will be also true for high-value, highly specialized areas, as illustrated by technology for computer aided process engineering (CAPE). We distinguish enterprise software from individual productivity products such as Microsoft Office, communications software such as Outlook and Netscape Messenger, and from PDA systems that combine some of these capabilities.

6.5.2 INDUSTRY STRUCTURE

6.5.2.1 The Supply Side

Companies that power the growth of Internet-based business processes and infrastructure—usually termed E-Business—now include disparate organizations, *e.g.* IBM, Siemens, SAP and Microsoft. Other major participants in the growth of IT include the major consultancies such as Accenture (formerly Andersen Consulting), Price Waterhouse Coopers (PCW) and CAP Gemini.

There is overlap in activity, but the first group is clearly identified with the development and sale of hardware and/or software. The second group is more closely identified with consulting for the business application of IT to improve business processes by some combination of automation and innovation. IBM is an example of a company that belongs to both groups. Had HP acquired PCW, it would have ended up in a similar position as IBM.

The history of the impact of IT on business organizations has been intertwined throughout the last decade with an organizational process called *business process improvement (BPI)*. From time to time this general term has been exploited or oversold in terms of the discipline needed for success. This has led on occasions to bad press when failure has occurred through misapplication or poor implementation. The concept of reengineering is valid and is enabling major productivity improvements. It has been seen that the companies that are better able to achieve these potential benefits are the companies that understand the technology and have good implementation skills. Good examples are Cisco, Dell and IBM.

In its recent form, BPI has evolved to encompass applying technology (primarily Internet based) to disintermediation strategies, and/or new business models that are technically enabled or dependent.

The separation of design and construction from operations has a parallel with historic engineering fields. At one point, the design, construction and operations were contained within one organization. Now this is the exception. The operating companies have focused on their operating competencies. They also developed supporting competencies in the acquisition of technology, facilities and infrastructure from companies whose core competencies are in those specialized fields. Good examples are the airline industry and the process industries In the past, the airlines designed and built their own planes. Famous examples are Pan Am and Iberia Air. Famous examples in the process industry are Shell, Exxon, BASF and DuPont. In the World War II era, DuPont had a very large design and construction department that encompassed civil engineering as well as chemical engineering. These industry structures are now gone and the operating companies are focused on operations, marketing and finance.

6.5.2.2 Evolving Business Models in Process Industries (1960-2000)

The process industries have seen an evolution of the industry structures on which business models are based. This is paralleled by a very similar evolution of the CAPE software used.

We might look back 40-50 years to trace these changes. At that point, the manufacturing companies were vertically integrated. They did (1) the product and process research, (2) process development, (3) built the process, (4) operated the process and (5) distributed the product using owned storage and transportation facilities. Over the last half century, we have evolved to the point where about 80% of this value chain is outsourced by the owner of the overall process.

The same thing happened in CAPE. The first versions of CAPE programs were developed over 40 years ago by companies such as DuPont (Chemical Process Evaluation System - CPES), Monsanto (Flowtran), and Dow in the US. These early versions ran on the scientific mainframe computers that were available 50 years ago, e.g. IBM 1620, 7090 and 7094.

Then specialized companies were formed that could leverage economy of scale in what become an increasingly complex and specialized technology. One key event in this history occurred in the US, when the Department of Engery sponsored the development of a process simulator at MIT. This was eventually spun off as a commercial venture called Aspen Technology. Process industry companies were under pressure to justify the specialized skills needed for in-house simulation systems. As this was going on, Monsanto's Flowtran was incorporated into Aspen's products. Also, Simulations Sciences picked up a public domain version of the MIT simulator with an interest in offering a PC system. New players entered the fray, e.g. Hyprotech in North America. Changes in this structure have evolved to the current day position, strongly influenced by the changes in the parent industry structures.

Then came a major new technology—modern IT—which has arguably reached the take-off stage. The two most famous and successful chemical engineers in the world are Jack Welch and Andy Grove (if we evaluate on the Market Cap of GE and Intel, which have been shaped or created in large part as a reflection of their respective leadership). Both take a view that IT and E-Business in the broad sense are going to have a transforming impact on business that will unfold over the next several decades.

What happened to Civil Engineering and Chemical Engineering will happen again in the field of IT. The operating companies will turn to the equivalents of Brown & Root for Civil Engineering or the Kellogg and IFPs of the process industry. Today, the old monolithic companies are still disengaging and building the new supporting competency that exists on other areas.

These structural transformations of organization places new demands on the ability to define, specify and evaluate both the software building blocks of information technology and the management processes for implementing reliable systems. A first step has been the emergence of generalist and specialist IT research organizations. Examples of the former are Gartner, Forrester, Seybold and IDC. Examples of the latter are SANS, ICSA, and Goldburg. The first two are taken from the IT/Internet security domain while the third is in the CRM area. The first group generates an incredible array of research reports and White Papers which are used in part by the final users of IT (the producers of final goods and services that make up GDP), but which are used even more extensively used by the suppliers of IT technology. Goldburg is an interesting example. This company evaluates software used in the CRM field. Each year, from a list of over 200 volunteering software suppliers, sixty are chosen and subjected to an actual testing of their products. From this list, thirty are selected and included in a comprehensive evaluation that rates performance on 52 separate performance criteria.

The current processes that support interactions of IT providers (including COTS software applications, design and building of infrastructure and increasingly the contracted operation of IT systems) is clearly undergoing fast change and will mature over the next one or two decades. Major advances in the software engineering technologies will clearly lead to huge improvements in the productivity and reliability of the IT building process. Donald Hagerling of the US Treasury comments that - in essence - we are on the lower slopes of a technology S curve. He predicts that:

> ...by the year 2050, the world's commerce will be dominated by the first country to have provided an infrastructure trusted to provide transactions and intellectual property. By then, the maturity of formal methods, software engineering practices and 6th generation languages will have brought trust and correctness to all software. All executable code will be generated by a formal model and will not be accepted unless it has been vetted and digitally signed by a recognized third party.

An **important conclusion** is that the business models will continue to evolve as they have been doing for the last 40 years. Grove and Welch say we are at the beginning of an S-curve because computing performance and costs plus advances in methodologies have evolved to a point that changes the rules.

The first step will be better systems for managing information about software applications. This will be utilized across major business areas such as sales automation and CRM and supply chain. It also offers gains for the specialized areas of scientific and engineering software. In the CAPE area, organizations like CO-LaN will support evolution of the next generation of standards-based CAPE software.

For the purposes of this discussion we will segment the new array of enterprise IT software into 1) computing and Internet infrastructure, *e.g.* operating systems and networking, 2) traditional basic accounting and transaction systems, 3) newly established business and application suites, *e.g.* SAP, PeopleSoft, 4) advanced software application platforms, *e.g.* Seibel Systems and BroadVision, and 5) specialized software solution components that provide specialized products or services.

The IT enterprise software encompasses a broad range of functionality and capability. Products will typically vary in the extent to which they support the range of functionality. This means that the best choice for a particular user—going forward we will use the terms user to refer to business enterprises, not individuals—will depend on that user's balance of requirements. For example, the balance of requirements for an American Airline sales automation system will be different from the balance of requirements for a GE Plastics materials supplier's sales automation system. From the software side, we can accomplish field sales automation/CRM with simple systems that cost $5-10k up to systems like Seibel and BroadVision that may cost $100k and up to acquire and startup. The use of software in the process industries is more specialized, but again we see a wide range of products that support differing balances of need in varying ways. One process industry organization lists about 2000 products in its annual catalogue of process industry engineering software. This of course excludes basic commercial systems such as SAP that are outside our current consideration.

6.5.2.3 Emerging Needs for the Enterprise

Communities in the enterprise are grappling with this fast growing management challenge. They are also doing this in a fast changing business environment that is shifting power and responsibility within and between organizations. The rate of change has never been greater. It can only be sustained in a coherent fashion through the adoption of IT enterprise management systems that will support a rational process for the evaluation, selection, purchase and asset accounting. This needs to be a distributed system that can accommodate the needs and tradeoffs that exist in global organizations between local initiative and localization requirements (*e.g.* language and training support) and the direct or perhaps visible economies of standardization and bulk purchasing power.

6.5.2.4 Brief History of IT in the Enterprise

The late 50s and early 60s were the days of the programming by wiring boards, followed by creating digitally stored programs written in machine instructions. Then came the breakthroughs of Jim Backus at IBM and Grace Murray Hopper with the US Navy. These led to FORTRAN and COBOL, which are still used extensively about 40 years after their initial creation. Admittedly, the current versions are very different from the early forms of these pioneering languages.

COBOL was probably responsible for the initial breakthrough in business computing. In the beginning, the COBOL programmers were in a subgroup of a larger organization that was called Accounting and Business Services. Several things happened: (1) computers became faster and less expensive, (2) application languages evolved, (3) SQL and relational database technology were invented by Codd and others, and (4) software engineering methodologies were developed. The scaling issue was addressed as IBM introduced its System 360 with the concept that one could move to a higher capacity, faster machine without re-writing the applications running on the smaller machines. Computer science also advanced on a broader form that included standardization, algorithmic development, creating a consistent, effective compiler technology and the continuing work done to improve software engineering methodology and practices. During this period, IBM was enormously successful in converting mechanical digital systems to modern computer based digital systems. This led to the penetration of computing into business that quickly outpaced the previous base of computing in the scientific and engineering fields.

IT was moving past the point at which IBM's founder Tom Watson declared that the world could probably use five computers. In the beginning scientific computing methods overlapped with the business area through the application of algorithmic approaches like Linear and Nonlinear Programming to solve/optimize transportation and supply chain models, and discrete simulation systems like GPSS were used to solve a wide range of industrial problems extending from tanker scheduling to detailed design of shop floor, manufacturing facilities.

A major transition/disruption for IT had its roots in the 1980s with the advent of PCs, fat clients in modern IT terminology. In the business arena, this led to fast growth of applications that enhance the productivity of individuals. In the beginning, this was focused on the office worker, but there were early experiments with sales force automation in the late eighties as truly portable PC were developed. The PC was responsible for the democratisation of digital computing for the entire organization. This led to initiatives in marketing and manufacturing organizations that were independent from the central IT organization. For the next 15 years, the control of the application and management of non-transactional computing applications has been - to some extent - been an area of contention. During that timeframe, the size and power of IT organizations grew rapidly. The same thing happened in the area of CAPE that we have noted previously.

In the current next phase, the Internet has the empowering impact that PCs did in the mid-eighties. In particular, the IT organizations of today and business organizations in general, face another jolting period of change. Authorities ranging from Peter Drucker to Information Week are making pronouncements about IT. Peter Drucker says that IT is moving to the point where its role can be liked to the role of printing and publishing once that new technology was mastered and reduced to an efficient and well-resourced commodity. Information Week has said the position of Chief Information Officer will go away and the competency of running a business will refocus on the older and more traditional competencies.

There may be an element of cheek in what Peter Drucker and Information Week said, but there is also a reality. What will surely happen is that the design and construction of IT applications and infrastructure will increasingly be outsourced. The new competency will be to manage this enormous outsourcing of what has become a major factor in supporting business operations. What will happen will parallel what happened in the process industries in an earlier era. There are also lessons to be learned from these earlier experiences. It is a fact that modern IT applications are complex. There are many parameters of performance that are critical. Some of the traditional ones are scalability and uptime. But there are many others. In fact, books (*e.g.* [15]) have been written on metrics for E-Commerce. These attempts to write down a set of metrics for E-Commerce or E-Business have been useful, but they only represent a point in time as the concepts and the technology of enterprise IT continue to unfold.

6.5.2.5 IT outside the Enterprise: Application Service Providers

The recent trend of outsourcing as much IT infrastructure and application systems as possible and reduce a company's activities to its core competencies has led to a new model of handling software systems employed in an enterprise. Since current applications are growing more and more complex and their tasks are becoming mission critical for every company, running and maintaining them has become very expensive and resource consuming. Detailed knowledge is necessary to operate those systems and large investments have to be made to provide a suitable hardware infrastructure. According to a study of the Gartner Group [11] one single workspace in an MLE costs up to $10.000. Only one third of these costs originate from investments in hardware or software systems. Two thirds are due to the resources that are needed to install, configure, and maintain this IT infrastructure.

Obviously, there is a large potential for saving money in outsourcing these activities to specialised companies, which have their core competencies in that area. This has given rise to a novel group of IT-services, namely application service providing. An application service provider (ASP) delivers software functionality as external service to a company. Therefore, an ASP can be defined as a company that 1) develops and delivers a service shared by multiple customers; 2) provides these services for a subscription or usage-based fee; and 3) supplies these services from a central location, over the Internet or a private network, as opposed to running on the customer's premises [23,34,35].

Usually the ASP provides and manages the server side soft- and hardware infrastructure thereby leveraging the customers' IT infrastructure. This dramatically reduces the total cost of ownership (TOC) for the software systems that are now hosted at the ASP. This will have a major impact on the handling of software in the industries. Forrester Research estimates that the ASP market will reach a volume of $21 billion in 2001 starting from $1 billion in 1997. Major software vendors such as Microsoft, Oracle, HP, SAP and IBM have adopted this idea and are making progress in implementing this new business model into their products [4].

It is important to distinguish this new class of IT services from the mainframe-oriented services, which were popular some twenty or thirty years ago. In those times powerful hardware was expensive and only large companies could afford their own mainframe systems. Powerful desktop solutions were not available.

Therefore, companies owning data processing centres sold computing time on their machines to their customers. But the applications running on these machines were running more or less stand-alone and were not integrated in the customers' IT infrastructure. Calculations were performed in batch mode often allowing no real interaction. This is the major difference to the ASP model. Modern Internet technology enables the ASP to operate the machines and the software in their location and to fully integrate it into the customers environment. The users on the client side can interact freely with the systems as if the systems were executed in their own environment.

Looking at the ASP landscape, we can identify three main areas where ASP solutions are in use today. The first class of applications making use of this new approach are desktop-based software systems. A good example is Mircrosoft's activities for making their office products ASP-enabled. These activities are mainly focused on delivery and management of desktop applications. The second class of ASP applications are electronic commerce (EC) applications. Many of the EC software vendors offer hosting services for their products. The adoption of the ASP model to EC applications has been rather easy because these systems were designed to be web-enabled (*i.e.* accessible through web-interfaces) from the beginning. This was not the case for the third class of applications, the enterprise resource planning (ERP) systems. Originally ERP systems were designed as corporate wide client-server systems but not as real web-enabled applications. Nevertheless, since these systems are extremely complex, using the ASP approach here is attractive. This is why the ERP vendors are investing lots of resources for web-enabling their applications. Nevertheless, integrating these systems in a corporate environment (which is still necessary) remains a critical task.

Therefore, a new concept for next generation ASP applications is required that facilitates easy integration and flexible combination of different functionalities. The basis for this new generation is the idea of component-based software architectures. Software components are software entities that are executed somewhere in a distributed environment (*e.g.* the Internet) and offer a specific set of services through well-defined interfaces. This technology enables the creation of ASP-hosted web services, which can be used as building blocks for more complex client side applications. Such technology facilitates the seamless integration of internal and external services to create applications that bring together data coming from vendors, partners, and internal sources. This idea has been adopted in the .NET architecture by Microsoft and will be supported by many forthcoming products [25]. Other technologies that have been employed in the CAPE-OPEN context are also suitable for implementing these ideas (see Sections 6.5.4.1 and 6.5.5.1).

Another critical issue when dealing with the ASP approach is security. From the transmission point of view the problem of security is solved. There are various technologies available on the market ensuring that data transmitted over the Internet cannot be spied on. Examples for such technologies are the Secure Socket Layer (SSL) Protocol or Pretty Good Privacy (PGP) for file encryption. A more critical issue is the question what will happen with the user data after transmission on the ASP's machines. The ASP has to access the data in order to perform calculations with it. Usually, this must be done on decrypted data which is a problem when sensitive data is involved. Therefore, the user must either simply trust the ASP or suitable protection mechanisms have to be developed.

6.5.2.6 A brief History of Process Simulation Applications

In the remainder of this chapter, we shall consider specifically emerging business models for process simulation applications. Looking at the historic development of CAPE tools and especially process simulators, we observe a similar evolution compared to IT in general. When flowsheeting tools became widespread some twenty years ago, they were deployed on mainframes, since PCs were just in their infancy. Only very large companies were able to afford the cost of mainframes and process simulators operated on in-house machines. Most end-users relied on service providers and paid solely for the use they were making of both the hardware and the software enabled by these service providers. Interestingly, such service providers were not subsidiaries of flowsheeting tools suppliers but mostly independent companies or subsidiaries from hardware suppliers. Connection to such providers relied on mostly unreliable phone networks and rather crude modems by today's standards. Costs of running the simulation of a distillation column (converging or not) were in the range of several hundred Euros per run.

Progressively, PCs have become the typical hardware on which flowsheeting tools are executed. The price of hardware has been steadily decreasing while performance has increased dramatically. It is more difficult to assess if and how software costs have also been drastically changing. PC-based process simulators have used the classical ways of buying and licensing so far. The software was installed in the customer's network and used traditional licensing models. Licenses have been acquired for each PC, giving an unlimited usage on each single machine (in contrast to the times of mainframes when each simulation had to be paid for separately). Thus there was a direct contact between the simulation tool provider and the end-user rather than going through some intermediary. User support for instance was also progressively given directly by the software supplier.

But the problems concerning TCO of software apply to process simulation software as well. Maintenance of these systems and the required hardware have become expensive so that other models such ASP have become more attractive. The process simulation software faces the same problems as other systems that where not designed with the ASP approach in mind. They were not web-enabled, and web-enabling these systems would mean a fundamental change in their design, which has not been accomplished so far. In process simulation software, not only web-enabling is an interesting problem field. Opening these systems up for integration with other applications is also desirable because designing and simulating a process is a very complex task, which typically involves many different tools with strong interrelations. Most of these tools are highly specialised, expensive, and require much expertise to run and maintained them. Therefore, process simulation can be seen as a good candidate (1) for applying component techniques for mutual integration and data exchange and (2) for using ASP techniques cutting down maintenance and integration costs.

However, the situation in the 1990s *w.r.t.* software for CAPE applications was that no component-based architectures were used, because most existing systems had a FORTRAN core which is not even object-oriented. The tools used for process simulation were closed, monolithic applications, which made it almost impossible to include new components from other vendors or to combine these tools [7]. But this is very desirable, as the manual exchange of data between those applications is tedious and error prone. Additionally, these tools were (and still are) highly heterogeneous because they may run on simple PCs using Windows or mainframes using Unix. To combine these tools, each of them must be divided up into standardised components with defined interfaces. Not only the problem of integrating the simulators is solved by such an approach but by doing so the systems are also moving one big step forward to web-enabled applications. Therefore, they will then be usable in an ASP environment.

Recent announcements made by main suppliers of simulation software lead us to believe that the pay-per-use business model may be back on stream, but using completely different technologies originating from the ASP area. Web-based distribution models are appearing that resemble what occurred twenty years ago but on a much more powerful and flexible level. The CAPE-OPEN standard is one possible means of accomplishing that goal by providing a complete set of standard component interfaces for process simulation software and will open up a new class of business models for process simulation software.

6.5.3 BUSINESS ASPECTS

The market of process simulators is dominated by a few major providers such as AspenTech Inc., AEA Technology Hyprotech and Simulation Sciences Inc., at least for the off-line share of the simulation market. Exact figures are difficult to obtain but 70% to 80% of the market may reasonably be considered as shared by these three companies. Concentration in this market has been going on steadily in the recent years. Or at least the concept of providing a whole range of solutions for an enterprise rather than just simulation tools has been implemented. That can be achieved through a string of well considered acquisitions, a simulation company buying a number of niche suppliers. That can also be done by integrating simulation companies within technological conglomerates providing a large range of software related products.

Typically an industrial end-user is tied up with a single simulation package provider or maybe two at the most when considering the links per business branch within a company. Global supply agreements are often entered where a number of licenses are provided up-front and then costs for additional licenses are fixed. These licenses may be proposed as a bundle, that is not only licenses for basic process simulation tools, but also for additional CAPE tools more specific to a number of applications.

One-to-one relationships between an end-user company and a simulation tool provider are common place. The simulation tool provider shapes the licensing contract to the specificities of each end-user and more so when the client is a large one. So a unique relationship tends to exist which is usually strengthened by a number of years of relationship. But with the change of technology to a more flexible and open environment this relationship could be subject to a fundamental change in the CAPE domain.

One reason for this change is that component technology in connection with industry standards such as CAPE-OPEN will open up the market for many small software vendors offering specialised niche products. Developing such products will become attractive for such vendors because when using the CAPE-OPEN standard they can implement a component, which may then be used in all simulators supporting the CAPE-OPEN standard. As two major simulator vendors (AspenTech and AEA Hyprotech) already support the standard this is likely to happen. A market survey has revealed some 2000 possible component providers.

Customers will then be able to pick the components that optimally suit their needs and employ them in their simulation in a plug-and-play manner. As the next logical step, these components could be made available for download over the internet (after paying a fee) and then be installed on a client's machine. Finally, combining the products of different component vendors, component marketplaces could be created on the web offering a wide variety of simulation software components through a single web-site. We will come back to the idea of a component marketplace later in this chapter discussing possible requirements and business models.

As mentioned above, software components in connection with current internet technology pave the way for ASPs in process simulation. ASP could come in different flavours in that application domain. One option would be the 'classic' ASP model where hosting and executing the simulator with all its components is done on the ASP's machines. But the fully component based CAPE-OPEN standard would also facilitate the ASP model for single components. Therefore, a possible scenario would be a simulation run on your local machine with some components (*e.g.* a powerful numerical solver, a CFD calculation or just a component that is needed very seldom) performed on a remote computer (*e.g.* a massive parallel machine). Finally, these ASP components could be offered in a marketplace together with 'normal' simulation components. In combination with flexible licensing models, this could form a very flexible environment for simulation users enabling them just to buy or rent the software they need and combine them in a plug-and-play manner. We will come back to these ideas later. As an overall consequence of the new technology and standards in the process simulation domain new service providers, besides classic software vendors, will be likely to arise. The basis for one class of services offering computational services will be ASP technology. The other class of services that will enter the market are information services. They could come in form of online simulation databases, component dictionaries with intelligent search mechanisms and many others. Using the technologies mentioned above they can be integrated in modern component based simulation environments. Later we will present more details on these services and possible business models. Additionally, these services might be combined with a marketplace for creating a one-stop full service simulation portal [18,28].

6.5.4 CONTEXT: SOFTWARE COMPONENTS FOR THE CHEMICAL INDUSTRY

New technologies have to be assessed in order to understand their value for business models in the CAPE domain.

6.5.4.1 Software components

Modern software systems, especially large enterprise systems, tend to grow more and more complex but require on the other hand increased flexibility. This flexibility facilitates easy integration of new subsystems or the extraction of functionality of parts of the system to be used elsewhere. Additionally, managing interdependencies between different subsystems in a complex enterprise environment has become a challenging task for software engineers. Therefore, the component-based approach for design and implementation has become popular and has proven useful [13].

Software components can be considered as the next step beyond objects. There are several definitions of a software component which are similar but which each emphasise particular aspects. Some definitions can be found in [12,33]. Based on these definitions the term "software component" is used in this chapter as follows: "A software component is an executable, stand-alone piece of software with a clearly defined interface and behaviour." The component's interface allows other pieces of software (*e.g.* other components) to access its functionality. There are different middleware approaches facilitating the implementation and deployment of software components by providing low level communication infrastructure, component lifecycle management, transaction services, and similar services. The most prominent middleware systems are (D)COM, COM+ and the .NET framework by Microsoft [24,25], CORBA created by the Object Management Group [26], and Enterprise Java Beans by Sun Microsystems [17]. In addition several proprietary middleware systems exist. Additional information on middleware is presented in Chapter 4.1.

The fact that software components are stand-alone pieces of software, which can be delivered and deployed, in a given environment makes them a good candidate for being traded on the web in a component marketplace. In fact several concepts and technical architectures for web-based component marketplaces have been developed [1,5,14,36]. However, all of these marketplaces follow a horizontal approach, i.e. the type of components that can be sold is not limited to specific domains. The horizontal approach can become a problem for various reasons; for example, the more generic a component is, the more technical understanding of the developer and the more customisation for usefulness in a specific domain is necessary.

6.5.4.2 CAPE-OPEN components

The challenges for software engineering concerning integration and flexibility aspects of complex software systems depicted above are also relevant to the CAPE domain, especially to process simulation. Process simulators are tools designed to create mathematical models of manufacturing facilities 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 central for designing new processes; they are also used extensively to predict behaviour of existing or proposed processes. [16] and [20] give an overview of trends and state-of-the-art approaches for simulation-based process engineering.

This problem was addressed by the European CAPE-OPEN (see for example, Chapter 4.3) initiative in which the chemical industries and major vendors of process simulation software have accomplished a standard of open interfaces for process simulation software [2,3]. The overall outcome of CAPE-OPEN was the definition of a conceptual design and interface specifications for simulators, which consist of an assembly of relatively large software components. As illustrated in Figure 1 and discussed in more depth in Chapter 4.3, CAPE-OPEN has identified the following standard components of a process simulator from a conceptual point of view [6].

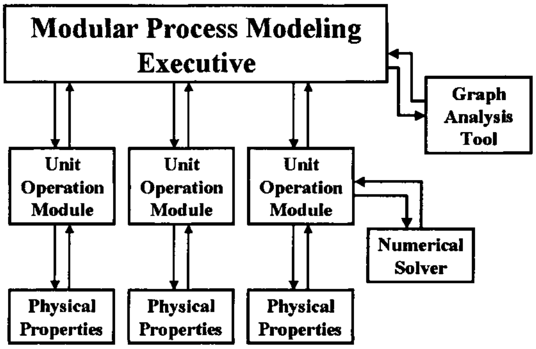


Figure 1. CAPE-OPEN simulator components

- Unit Operation Modules (Units) represent the behaviour of physical process steps or apparatuses (e.g. a mixer or a reactor). They are linked to the simulation flowsheet, which represents an abstraction of the plant structure, and they compute the chemical results of the process steps they represent. The overall process model which represents the plant is assembled from predefined unit libraries.
- Physical Properties (Thermodynamics) Packages: an important functionality of a process simulator is its ability to calculate thermodynamic and physical properties of materials (*e.g.* density or boiling point). Typically, they consist of a database containing many simple properties for a set of chemical species and a set of calculation routines for more complex properties based upon specific mathematical models.
- Numerical Solvers: the mathematical process models of a unit operation or a complete plant are large systems of equations and highly non-linear. An analytical solution is impossible. Therefore, iterative, numerical approaches are either used to solve the equations of a single unit operation module or to solve the overall flowsheet.
- Simulator Executive: the simulator core controls the set-up and execution of the simulation, i.e. analysing the flowsheet and calculate the units. Furthermore, it is responsible for a consistent flowsheet set-up and error checking. The simulator executive itself is not a component but a platform that co-ordinates the actions performed on the various components described above.

The semantics of these components and their interdependencies have been defined in terms of UML diagrams, COM and CORBA interface definitions, and textual descriptions [8].

6.5.5 REQUIREMENTS FOR A CAPE-OPEN COMPONENT MARKETPLACE

As explained above, component technology in combination with internet technology and industry domain standards such as CAPE-OPEN can be used for trading software components over the web. For creating such a marketplace and associated IT services an overview of electronic business transactions as applications of a classic three-phase model for business transactions [28] will be presented. Then the requirements for customers, vendors and intermediaries (*e.g.* a marketplace owner) will be discussed and possible business models will be derived. Some of the services and business models presented here would also be valuable outside a marketplace context. Therefore, possible combinations of these services will be discussed as well.

6.5.5.1 A model of an electronic business transaction

During a commerce process, the involved participants usually go through three phases [28]. Firstly, a party looks for potential business partners. A buyer wants to find relevant suppliers of the product he/she is looking for; a seller might want to find potential customers for the products he/she can supply. After locating potential (new) partners, the second step is to come to an agreement that is acceptable to all partners. Partners might bargain about the price, might find a compromise about the delivery dates, and might negotiate about quality aspects of the products. The aim is to finalise a contract that specifies the business deal. Therefore, this second phase concerns negotiation about details of the agreement [31]. If the negotiation is successful then a business deal is struck and the outcome is a contract which will then have to be processed by the partners in the third phase, e.g. concerning logistics, payment etc. The general model that can be extracted from the above observations is one of three phases, see Figure 2.

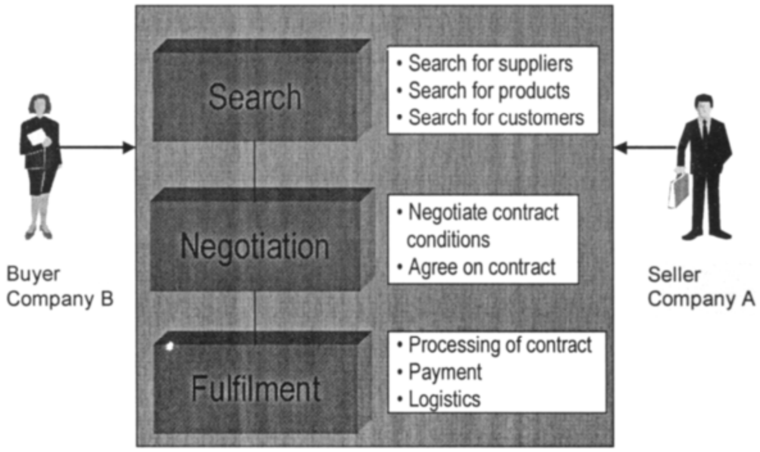


Figure 2. Three Phases of a Commerce Process [28]

The **search phase** is about finding business partners; the **negotiation phase** is about finding agreements leading to a contract; the **fulfilment phase** concerns the execution of the contract. It is important to state that not all of the three phases are present in every commerce process. For example, if it is clear who the negotiation partner is, then no search needs to take place. Furthermore, the three phases do not need to occur in a strictly sequential order. It is possible to jump back, *e.g.* if it becomes clear that a certain negotiation is a dead end, then a business partner might want to come back to the earlier search results and start new negotiations.

The three-phase model is independent of any technological means, *i.e.* it is valid for traditional commerce processes as well as for electronic commerce interactions (see its application in the MEMO–Mediating and Monitoring Electronic Commerce–Project [http://www.abnamro.com/memo/]). For example, a buyer might look for potential suppliers in the yellow pages, in the catalogues of chambers of commerce or on the internet. In this chapter we will concentrate on *electronic marketplaces for business-to-business electronic commerce.* The current practices in such marketplaces can best be discussed using an example of an existing business-to-business marketplace of the chemical industry called *chemUnity* [http://www.chemunity.com]. A buyer's request containing information about the product he/she wants to purchase, its type and concentration, the delivery address and time is transferred via the marketplace to all potential suppliers as specified by the buyer. Suppliers have a fixed amount of time (usually 25 hours) to react. Those who choose to send an offer will be taken into account. The marketplace based on the buyer's selection criteria determines the best offer. If the best offer is within the price range indicated by the buyer, then the transaction is completed and the following obligations exist: The seller must supply the product(s) indicated in the original request whereas the buyer must provide the payment according to the offer received.

Abstracting from the example, we can state general observations concerning the three phases in electronic marketplaces as follows.

The search phase consists of (extended) keyword search based on some classification, *e.g.* a product catalogue, a list of companies in a certain branch etc. Using these kinds of search mechanisms presupposes good knowledge of the search items by the search party and an appropriately structured search domain. For example, if a company would like to find new business contacts or would like to find suppliers of certain products that have different names in different companies, then keyword-based search is clearly insufficient. More intelligent search mechanisms are available [19] but will not be discussed here.

The protocols of electronic negotiations that are usually supported in electronic marketplaces are auctions or electronic catalogues [31]. In the latter case, the option is one of "take it or leave it" – either to order at the price specified in the catalogue or not to enter into the business transaction at all. Auctions can be useful for settings as described above. However, even in the example of *chemUnity* certain problems are obvious. Such a model cannot support complex negotiations. For example, the cheapest supplier might not be the one offering the best quality, the cheapest supplier might not be trustworthy, the third cheapest supplier might be able to deliver much quicker than the cheapest one etc. Furthermore, if negotiations concern frame contracts, then a different negotiation protocol is required. Highly interactive exchanges that occur in traditional commerce can be transferred to electronic commerce where, on the one hand, the potential of information technology can be exploited to offer new functionalities and support effective interactions and, on the other hand, information technology cannot (and indeed should not) replace the human negotiator by an automated software agent but rather support human negotiators in their tasks [30,31].

The fulfilment phase is the one that is usually covered best in any electronic marketplace. Payment models are supported (*e.g.* payment by credit card) and an integration with the companies' logistic systems is achieved. If all goes well after the contract has been finalised then such a model is sufficient. However, if disagreements occur between the parties as to which obligations need to be fulfilled, whether certain duties have been carried out according to the agreements made during the negotiation *etc.*, there is hardly any support to help solving such problems. No history behind an agreement is usually provided that could help the parties themselves or an independent third party to understand why certain agreements have been reached and where the specific problem lies.
To summarise, there are potential problems with respect to current practises for all three phases. Nowadays there exist a number of electronic marketplaces for different branches. Therefore, a (new) marketplace requires additional functionalities for all phases to make it attractive to participants and to distinguish it from its competitors, *e.g.*,

- Search: To capture different relations between concepts, semantic search mechanisms need to be provided so that similar and related information can be found.
- Negotiate: A new negotiation protocol is required that is interaction-based and supports the communication-intensive exchanges in complex negotiations.
- Fulfil: Different payment models should be provided to capture the different needs of various application contexts. Furthermore, a monitoring component could help to observe the interactions and trace them back in case of conflicts.

We have discussed before that software components are good candidates for being sold in a web-based marketplace due to their properties. We have also pointed out that for making a marketplace attractive, additional services should be offered which can be effectively designed for a vertical, *i.e.* domain-specific, marketplace only.

Next we will define the requirements for a CO marketplace from three different viewpoints.

- Firstly, the component users, *i.e.* operating companies and others, will view the marketplace mainly as a platform for buying the components they are interested in without having to search individually for the vendors that can provide them.
- Secondly, the component vendors see the marketplace mainly as a forum for intelligent presentation. They can be found by interested customers and can find prospective clients. Furthermore, vendors will get a notion of the requirements in the market, thereby being able to quickly adapt to changing needs.
- Thirdly, the marketplace provider is interested in an efficient management of the marketplace and the related data. Effective interactions must be ensured to attract participants willing to pay for joining the marketplace.

These different viewpoints lead to different requirements that will be discussed in the following three sections.

6.5.5.2 Requirements of component users

In each marketplace, the customers are the ones that determine whether the market is successful of not. Every business model depends on the customer buying the goods offered, regardless of how the money is actually earned. Thus special attention has to be paid to the customers' needs. The most important issue here is that the CAPE-OPEN standard must be easy to use. There are several aspects considering this:

Suitable components must be easy to find

A typical CAPE-OPEN simulator executive (COSE) will contain a basic set of units and solvers, maybe in a CAPE-OPEN compliant form or as proprietary parts of the simulator (*e.g.* AspenPlus). This means that if a user (probably a process engineer) needs additional components he/she is looking for some very specific features. These features highly depend on the engineer's software environment and the task in work. This can be the required interplay between the COSE and other components already in use or special features of the model the component implements. The component user will need a component dictionary combined with advanced product search facilities that reflect the complicated domain of process simulation.

While the CAPE-OPEN standard offers the possibility for small and medium-sized enterprises to participate in the simulation component market as vendors, the component user will surely not want to search through the product lists of more than a few suppliers. Therefore centralised locations are needed where users can start their search for components.

Therefore, a marketplace should offer a broker functionality so that a company does not have to negotiate with each vendor about similar products. The broker should be able to search through the products of several or all vendors using a given specification for a component. It should then be able to present a list of solutions to the customer or to select a specific product according to the needs of the customer. The engineering company can thereby easily access a file of components without much overhead.

Sometimes a company will not find a component for a given task. The desired component could be a slight variation of an existing component as well as a complex and not yet developed implementation of a model. CAPE-OPEN offers the possibility for vendors and suppliers to tailor these components to the specific need of a customer, which then can be integrated as plug-and-play component. Using the broker functionality or through direct contact with a vendor the customer should be able to negotiate about the tailoring of such a component. Here it will be vital to produce an exact specification of the component to develop. A marketplace has to support these negotiations in a structured way to ensure that the customer gets the required product. The CAPE-OPEN structures have to be reflected within the negotiation. On the other hand, the negotiation support must be flexible enough to specify conditions that are important to the customer but that are not part of CAPE-OPEN.

Flexible licensing models are needed

The CAPE-OPEN standard offers the possibility for a wide range of simulation components – from easy but necessary variations of standard units up to complex and specialised simulation or solver components. These different components should be handled in the same way. The simple ones should be accessible without disturbance of the work process, some components will be often used by a process engineer and should be added to her/his default set of components while others will only be used once in a very specialised situation. Some components might need calculation power that can be offered by an ASP and do not run in the engineer's local network.

The CAPE-OPEN standard offers a flexibility that can be used to realise a huge set of different license models for components. The license models in the process simulation domain have to be more flexible than current license models for software where the use of the software is usually restricted to a certain time and a certain set of individual or concurrent users. To reflect the different application situations, license models are required in which the actual use of the component is paid.

It is often not suitable to explicitly buy each component, in particular concerning standard components. The component user wants to select the necessary component and use it immediately. To support this, components should be *bundled* into groups that can be bought as a whole. Again it will certainly not be useful to create fixed bundles of components–the user should not be forced to buy a set of components of which he only needs some. To offer the greatest flexibility, negotiations for component bundles should be offered. This should include dynamic bundles that develop during the use of components. Here the customer buys a number of components from a given larger set. A number of components from this set can be used so that the customer can choose those components that are useful for him/her and skipping those that are not required. Licensing approaches should support these bundles both for static as well as for dynamic sets.

Component labelling ensures standard compliance

Before buying or renting a component, the customer has to know that it is really CAPE-OPEN compliant. Should it be otherwise, plug-and-play with such a component would not be possible. Additionally, the CAPE-OPEN standard does not cover every aspect of process simulators, *e.g.* performance aspects or suitability for a specific COSE. Therefore, the components should be labelled to guarantee standard compliance and other specific aspects. These labels should contain the information whether the component can work in an arbitrary COSE and whether the component supports additional features of the COSE that are not part of the CAPE-OPEN standard.

Easy and fast deployment of software components

The deployment of the CAPE-OPEN component in the COSE should work without problems. As the standard does not specify how the deployment has to be done, this should be incorporated with the COSE and the marketplace. For the customer, there should be a way to select the necessary components using a web-browser and plug them directly into the COSE in use. Special features of the component according to a specific COSE should be considered.

User support

User support for customers should be available regarding the standard definition. The standard specification documents have to be accessible in an easy way and additional documents have to be supplied. These documents are special collections containing frequently asked questions and answers (FAQs) and "HOW TO"-documents that explain common tasks that arise with the standard.

In addition to buying ready-to-use components from component vendors, some companies might want to develop their own CAPE-OPEN compliant components to incorporate in-house software and knowledge into a COSE. To support this, there should be self-training tools and migration guidelines that supplement training courses and consulting. Other additional services could include online process and property databases or expert forums helping in creating a model for a specific process.

6.5.5.3 Requirements of component vendors

Until now we have looked at customers that buy a CAPE-OPEN component to use it with a COSE. Now we take a look at component vendors. Here vendor means a software company or operating company as the developer of in-house software. A vendor develops CAPE-OPEN compliant components and uses the marketplace to offer these components. Here other requirements for the marketplace arise:

Support for component development

As in the case of the customers the marketplace should offer a wide range of support for the CAPE-OPEN standard. The developer should find all necessary information for development at the marketplace–the marketplace should especially be linked with the CO-LaN.

Make components available and find customers

Small and medium-sized enterprises (SMEs) can use the marketplace to enter the market of CAPE software. Small but specialised companies can use the standard to bring their own expertise to the customer. For example, it has been hard up to now to use a custom tailored model for a unit within the simulation of a whole plant. With CAPE-OPEN, this model can be plugged into the COSE and cooperate with the rest of the structural plant model. For small companies developing such components, it is vital to find customers. A marketplace gives the opportunity to do so.

For SMEs as well as for large software vendors it is vital that the customers find the components offered. Therefore a marketplace can make the components and data about them available so that a customer can search the components according to his or her needs. The vendors therefore have to specify information about the components that can be used in a search.

Large software vendors might be interested to offer their own marketplace of components. Since not even large vendors will offer components for every situation, they will be interested to augment their own set of components with niche products that they cannot/do not want to offer. Here small and large vendors can cooperate to offer a marketplace that is highly attractive to customers.

A marketplace can be used not only to offer products but also to find customers looking for a specific new component. Combined with the above mentioned broker functionality, vendors can be able to find orders for new products that do not yet exist. Again, a detailed negotiation between customer and vendor or between broker and vendor is necessary to specify the exact requirements for the component. Only if a marketplace actively supports these features, smaller software vendors will be able to find new customers. It opens up the market for software companies specialised in developing hand-tailored software in the CAPE-domain.

Additional services

If the marketplace offers the possibility to test a component as CAPE-OPEN compliant, the customer can be sure to buy components with a certain quality. A testing facility therefore makes the marketplace more attractive for the customer. As we have already seen the CO compliance alone might not be enough as a quality indicator. Component labelling according to other quality factors such as performance checks and specifications should therefore be available.

As a marketplace collects a large amount of information about the market of software components the marketplace provider (*e.g.* a large software vendor) can easily and automatically create market surveys that can be used to show up the potential for new products or the modification of existing products.

6.5.5.4 Requirements of marketplace provider

This section will describe the requirements of the marketplace provider. We will present the requirements independent of the provider's original role, i.e. the marketplace could be provided by a large vendor company, by a consortium of users or niche providers, or by an independent organization.

Firstly, the marketplace has to manage data for various aspects. On the one hand, there is static data such as product and company profiles, which will not change very often. The marketplace has to provide a coherent and understandable data model, so that search on the profiles is facilitated and the results are comparable [27]. For example, the data model for products must be able to represent very different but very specific descriptions of the products. On the other hand, dynamic data that is created during a business transaction must also be managed. Dynamic data are, for example, the results of a search, the messages exchanged during a negotiation, and the traces recorded during the fulfilment.

To make the marketplace more competitive the marketplace provider has to offer additional services. One example are trust services that can be provided by the marketplace provider itself or by a trusted third party (TTP) such as a bank, an insurance company, or an umbrella organisation [9,10,21,22]. The TTP will act as the monitor of the marketplace [29]. The TTP can monitor the messages, which are exchanged between business partners during the negotiation or fulfilment phase [29]. In case of a conflict, the TTP has to provide the data which has been recorded for the given business transaction. This data can then be used to resolve the conflict. From a technological point of view, the marketplace needs, therefore, to offer interfaces that allow the TTP to monitor a transaction.

Finally, the marketplace provider can take over a more active role as an intermediary between customers and suppliers of components. As the provider knows what the customers are looking for, the provider can provide this information about current user demands to the vendors. This can also be done in the opposite direction, i.e. new products offered by vendors can be advertised to potential customers who have recently been looking for a similar component. These functionalities can be supported technically by data mining applications. Data mining is a mechanism to detect patterns in large data sets. In this context, it can be used to detect repeating demands of customers for similar products. Furthermore, the marketplace provider can support the customer during the negotiation phase by conducting pre-negotiations with potential suppliers, so that the customer does not have to deal with many vendors simultaneously when he/she is looking for one component.

All of the three types of requirements can be fulfilled in different ways. Based on the requirements, we can derive new (future/emerging) business models, which are described in the next section.

6.5.6 BUSINESS MODELS

Based on requirements, a number of services can be derived. These services can become the basis for various business models. Depending on the type of service these business models could be applied by three kinds of companies: component vendors, marketplace owners, or independent service providers. However, there could be intersections between these groups. A marketplace owner could possibly be a completely independent company. But it seems more likely that a large software vendor(s) or a user group would operate such a marketplace. The same holds for the independent service providers, which could offer their services to be plugged into a marketplace or to be used directly by the software vendors and users.

6.5.6.1 Component dictionaries

As we have seen for the requirements of component users as well as those of suppliers, the most important thing is to have a platform for offering and finding components. Therefore, the most important service that has to be offered is a *component dictionary*, that includes information about components and its vendors. This service will be a part of the CO-LaN that will not offer features such as advanced search mechanisms. In particular the advanced search will be the added value of a component dictionary, the complex field of CAPE components needs more than a search based on 5 categories (*e.g.* vendor, operating system, unit type) to reflect the often fine-grained differences between units or mathematical models.

The business models related to component dictionaries are similar to those of internet search engines: The search engine is an ideal advertising platform or an integrated feature to another service. Another option would be to charge fees for entering or retrieving component information into the dictionary. Thus it is possible to have the dictionary as a stand-alone service or as part of a marketplace. While the stand-alone service will have to concentrate on offering a most complete list of components, the integrated service will have to support the vendors that participate in the marketplace.

6.5.6.2 ASP services

The ASP can fulfil different roles each of them offering different business models. Mostly, an ASP will offer several of activities mentioned here and make its money from a combination of the business models presented here [18]. The ASP could be a software vendors itself, the marketplace owner or an independent provider.

The ASP may own and maintain the software (component) and charge licence fees to the users. The charges can be base on various license models depending on his relation to the customer. The customer can hold a permanent licence for unlimited access (maybe for a limited time) to the software as the standard model. More flexible approaches such as pay-per-time, pay-per-use combined with a monthly base fee or software bundling are more advanced options.

The ASP may host the software (*i.e.* execute it on his machines) and provide access via the Internet or a private network for the customer. Additionally, the ASP takes care of maintaining hardware, software, and other services, e.g. doing backups. Depending on the security and availability, the ASP can guarantee the customer has to pay a monthly fee.

The ASP may host and/or own data-related services such as the component directory mentioned above or any other kind of online database. Access to such a service may be charged on a transaction base or as by monthly fee. A variant of this model would be data push services where the ASP automatically provides data to the customer without letting the client explicitly ask for it (*e.g.* stock quotes). Charging models are similar to the latter case.

If the ASP has programming competency it can also offer customisation and integration services. This means that it is responsible for writing software that connects the ASP's server site application with the ones running locally in the clients network.

As the ASP is able collect user data it can use this information to make money by selling it (*e.g.* by using data mining techniques, see below)

6.5.6.3 Community platform

A business opportunity that has already been taken up in some of the existing marketplaces is that of providing community-building services [32]. Participants in a marketplace share similar interests. For example, customers want to buy software components for the chemical industry that adhere to a certain standard and quality. By setting up and maintaining such a community platform, the marketplace provider can supply services to a large group of users (be it vendors, customers, or other participants) with similar demands. The main service is that of providing information. In the present context, that could be, for example, information about the CO standard, about the participants in the marketplace, about specific components, about usage and set-ups of such components. A set of frequently asked questions could be provided which prevents the provider from having to answer similar questions by different parties while being able to broadcast relevant information in an efficient way. The information provider could, therefore, enable consulting via such a community platform. Furthermore, individual consulting could be offered, targeted at specific groups of participants. Access to the community could be restricted by charging a membership fee to all companies or individuals wanting to participate thereby generating revenue. This model has been applied by various business-to-business marketplaces.

6.5.6.4 Certification Authority

One of the key arguments for using software components in combination with an software standard such as CAPE-OPEN is that plug-and-play interoperability is possible. This facilitates easy and inexpensive integration of third party software into the customers system. However, for plug-and-play to work, standard compliance of the software components must be ensured. This can be accomplished by performing tests for standard compliance on the software and granting a label or certificate if the test is successfully passed.

In the case of CAPE-OPEN, labelling is first performed by the CO-LaN organisation but it is planned that this activity will later be performed by other companies (*e.g.* a large software vendor). The business model in this case will be to charge a component supplier for performing and evaluating a compliance test for the software component. This basic model is already implemented by the CO-LaN organization. Additional tests with regard to suitability for running in specific simulators or performance issues may also be included in more advanced (and more expensive) test suites.

For the component vendor, going through these tests will be attractive because the label will guarantee a certain level of quality of his software. Additionally, the labelling company may maintain a detailed dictionary compliant components (see above), which can be seen as marketing instrument for the vendor. The certification authority could generate additional revenue by charging the vendor for entering his data in a dictionary or by providing certification data to a dictionary owner.

6.5.6.5 Broker platform

Since the interactions between vendors and customers as described before are not direct but take place via a marketplace, an intermediary mediates between the parties. This mediating role can be fulfilled by a broker system. Such broker system would then support the three phases of a business transaction, namely search, negotiate, fulfil [27]. The broker could also play an active role since it acquires a large amount of knowledge about the market exchanges. For example, the broker could find out business needs, bundle the requirements and sell this information to the vendors. Pre-negotiations could take place about standard requests that could be offered to the customers. Therefore, customers would not have to negotiate about all requests themselves with many different vendors. Apart from mediating, the broker could also act as a monitor. Conflict resolution could be offered to the participants by monitoring the contract fulfilment. Traceability can be used as a tool to find out what happened before, i.e. to see the history behind agreements, which is helpful for all participants involved in the business interactions [29,31]. Similar to the community platform access fees for the broker platform could be charged. Another option would be a transactions based fee scheme here based on the value of the business transactions.

6.5.6.6 Data mining

Another business opportunity is offered by the data gathered at the marketplace. It contains valuable information about the behaviour of suppliers and customers in the marketplace. New demands or trends can be detected by analysing this data. Furthermore, the credibility of a vendor or a customer can be judged on the traces of previous business transactions. For example, a customer could be marked as less credible if he/she has often refused to pay for the delivered components. On the other hand, the quality of the components of a specific vendor may be considered low if customers often complain about them. All this information is gathered during the operation of the marketplace in the negotiation and fulfilment phases and can be offered to the participants as an additional service. Of course, all parties have to agree to allow the collection of this kind of information and to present it to the participants of the marketplace.

Vendors and customers will certainly agree to it, as not only negative statements as given in the examples before can be deduced from the data. Positive statements about the quality of a vendor or the credibility of customer can be made as well.

The marketplace provider can also gather information about the efficiency of specific licence models or negotiation strategies. The provider as an intermediary between vendors and customers knows which models are used more frequently than others. The information about the most successful licence models can be offered to vendor companies that then may result in the change of licence models.

6.5.7 CONCLUSION

The discussion in this chapter shows that trends in computer-aided process engineering software are parallel to trends in general IT. This also applies to the most recent trend, the one towards opening up global markets through electronic business including eMarkets for CAPE software components to include more vendors, and Application Service Providing to enable more customers.

However, the complexity of CAPE software requires additional steps towards making such innovative and beneficial solutions possible. One of the most crucial requirements, addressed in our case by the CAPE-OPEN standard and the related methods and software tools, is the management, method and tool support for a broadly accepted and learnable standard. In particular, a global web-based service can be developed this way.

On the solid ground of such a standard, the marketplace provider can be a vendor, a coalition of customers, an independent organisation (*e.g.* a Trusted Third Party), or various combinations of the above. Most importantly for continuous innovation, SME's can enter the market previously dominated by few vendors, and the large vendors can offer innovative combinations of their own and of third-party products. In this way, a well-designed eMarket-Place for CAPE software can play a critical role in the fast-moving process industries once it has gained an initial critical mass to be broadly used and accepted. The Global CAPE OPEN consortium is actively pursuing this avenue.

6.5.8 REFERENCES

[1] A. Behle: *An Internet-based Information System for Cooperative Software Reuse.* In Proc. of 5th IEEE International Conference on Software Reuse, pp 236-245, 1998.

[2] B. Braunschweig , M. Jarke, A. Becks, J. Köller and C. Tresp: *Designing Standards for Open Simulation Environments in the Chemical Industries: A Computer-Supported Use-Case Approach.* In Proc. of the 9th Annual Int. Symposium of the Int. Council on Systems Engineering, Brighton, England, June, 1999.

[3] B. Braunschweig, M. Jarke, J. Köller, W. Marquardt and L .v. Wedel: *CAPE-OPEN - experiences from a standardization effort in chemical industries.* In Proc. Intl. Conf. Standardization and Innovation in Information Technology (SIIT99), Aachen 1999.

[4] Business Software Alliance: *Building an Information Economy*, Software Industry Positions U.S. for New Digital Era, 1999.

[5] P. Buxmann, W. König and F. Rose: *The Java Repository – An Electronic Intermediary for Java Resources.* In Proc. of the 7th Int'l Conference of the International Information Management Assoc., 1996.

[6] CAPE-OPEN, Conceptual Design Document 2, http://www. global-cape-open.org/CAPE-OPEN_standard.html, 1998.

[7] CAPE-OPEN, Project Programme, Annex I, Brite/EuRam Be 3512, 1996.

[8] CAPE-OPEN, Web-site, www.global-cape-open.org, 2001.

[9] A.M. Chircu, G.B. Davis and R.J. Kauffman: *Trust, Expertise, and E-Commerce Intermediary Adoptions.* In Proceedings of the 2000 Americas Conference on Information Systems, Long Beach, CA, 2000.

[10] T.H. Clark and H.G. Lee: *Electronic Intermediaries: Trust Building and Market Differentiation.* In Proceedings of Proceedings of the Thirty-Second Annual Hawaii International Conference on System Science, IEEE Computer Society, 1999.

[11] L. Cohen et al.: *External Service Providers (ESP): Delivering on the Promise of Value.* PC Week, June 1999.

[12] D. D'Souza and A.C. Wills: *Objects, Components and Frameworks with UML: The Catalysis Approach.* Addison Wesley, Reading, MA, 1999.

[13] J. Hopkins: *Component Primer.* Communications of the ACM, **43**(10), October 2000.

[14] H.-A. Jacobsen, O. Günther and G. Riessen: *Component Leasing on the World Wide Web.* In Proc. of the 1st ACM Conf. Electronic Commerce, ACM Press, 1999.

[15] S. Jagannathan, J. Srinivasan, J. Kalman, *Internet Commerce Metrics and Models in the New Era of Accountability*, Prentice Hall, 2001

[16] M. Jarke and W. Marquardt: *Design and evaluation of computer-aided process modeling tools.* In Proc of the Intl. Conference on Intelligent Systems in Process Engineering (Snowmass, Co, July 1995), AIChE Symposium Series, vol. 92, 1996, 97-109.

[17] Java 2 Platform Enterprise Edition http://java.sun.com/ j2ee, 2000.

[18] J. Köller, T. List and M. Jarke: *Designing a Component Store for Chemical Engineering Software Solutions.* In Proc. 34th Hawaiian Intl. Conf. On System Sciences (HICSS-34), Maui, 2001.

[19] C.J. Leune: *Final document on searching, querying and discovering mechanisms.* MEMO Deliverable 1.4, July 2000 (http://www.abnamro.com/memo/).

[20] W. Marquardt: *Trends in Computer-Aided Process Modeling.* In Computers and Chemical Engineering, 1995.

[21] R.C. Mayer, J.H.Davis and F.D. Schoormann: *An Integrative Model of Organizational Trust.* Academy of Management Review, **20**(3), 1995, 709-734.

[22] M. Merz, W. Lamersdorf and K. Müller: *Trusted Third-party Services in COSM.* Electronic Markets, 4(2), 1994, 7-8.

[23] Microsoft Corporation: *Application Service Providers: Evolution and Resources.* White Paper, 2000.

[24] Microsoft COM web-site www.microsoft.com/com, 2001.

[25] Microsoft .NET web-site, www.microsoft.com/net, 2001.

[26] OMG CORBA web site: www.omg.org/corba, 2001.

[27] C. Quix and M. Schoop: *Facilitating Business-to-Business Electronic Commerce for Small and Medium-Sized Enterprises.* In Proc. First Intl. Conference on Electronic Commerce and Web Technologies, EC-Web 2000, Greenwich, UK, September 2000, pp 442-451, Springer Verlag.

[28] M. Schoop, J. Köller, T. List and C. Quix: *A Three-Phase Model of Electronic Marketplaces for Software Components in Chemical Engineering.* Proc. of First IFIP Conference on E-Commerce, E-Government, E-Business, Zurich, Switzerland, October 2001 (to appear).

[29] M. Schoop and T. List. To Monitor or not to Monitor – The Role of Trusted Third Parties in Electronic Marketplaces. In Proc. Wirtschaftsinformatik 2001, Augsburg, Germany, September 2001 (to appear).

[30] M. Schoop, C. Quix: *Towards Effective Negotiation Support in Electronic Marketplaces.* In Proc. Tenth Annual Workshop on Information Technologies & Systems, WITS 2000, pp 1-6, Brisbane, Australia, 2000.

[31] M. Schoop and C. Quix: *DOC.COM: Combining document and communication management for negotiation support in business-to-business electronic commerce.* In Proc. 34th Hawaiian Intl. Conf. On System Sciences (HICSS-34), Maui, 2001.

[32] K. Stanoevska-Slabeva and B. Schmid. Requirements Analysis for Community Supporting Platforms based on the Media Reference Model. Electronic Markets 10(4): 250-257, 2000.

[33] C. Szyperski: *Component Software: Beyond Object-Oriented Programming.* Addison Wesley Longman Ltd, 1998.

[34] G. Tamm and O. Günther: *Business models for ASP marketplaces.* In Proc. 12th European Conference on Information Systems (ECIS 2000), Vienna, 2000.

[35] G. Tamm and O. Günther: *On-Demand Application Integration: Business Concepts and Strategies for the ASP Market.* In Proc. 4th Workshop on Federated Databases, Shaker, 1999.

[36] E. Whitehead, J. Robbins, N. Medvidovic and N. Taylor: *Software Architecture: Foundation of a Software Component Marketplace.* In Proc. 1st International Workshop on Architectures for Software Systems, Seattle WA, April 24-25, 1995, 276-282.

Part VII: Case Studies

The chapters of this part present the modern use of CAPE tools and architectures on representative process examples. The designers and developers of these examples from the Global CAPE Open project have also written these chapters.

Chapter 7.1 (Bayer et al) highlights the need for a number of CAPE methods and tools during the lifecycle stages of conceptual process design. Two case studies are presented. For each of these case studies, the activity model, the workflow and the dataflow are presented, highlighting the need for different tools and how they can be used efficiently through appropriate interfaces. The two case studies deal with the conceptual design of processes for the production of methyl acetate and production of polyamide6.

Chapter 7.2 (Siepmann et al.) highlights the need for open simulation environments with a prototype describing two alternative COM CORBA bridges. The prototype highlights, for example, wrapping around an external third party software (in this case, a Fluent CFD simulation package) for multilevel modelling purposes.

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)

Chapter 7.1: Case Studies in Design & Analysis

B. Bayer, M. Eggersmann, R. Gani & R. Schneider

7.1.1 INTRODUCTION

The objective of this chapter is to highlight the need for interfaces and standards through the description of the activities during the conceptual design stage of the process design lifecycle involving the design of two processes. Two case studies have been used to highlight this. The case studies do not provide numerical results but point out the work-flow and data-flow associated with the solving of the conceptual design problems related to these case studies. In this way, the reader can get an idea of the need for standard interfaces when solving practical design problems.

The first process (case study) involves conceptual design for the production of methyl acetate while the second process (case study) involves the production of polyamide-6. Both case studies employ a number of commercial software as well as software developed in-house (in academic institutions). More details on these software can be found in chapters of parts III, IV and V.

A description of the various stages of the process design lifecycle is described in chapter 2.3.

7.1.2 CASE STUDY: METHYL ACETATE

7.1.2.1 Introduction

Methyl acetate is an ester of some industrial importance. It is used mainly as a solvent for fast-drying lacquers and as solvent for cellulose nitrate and other cellulose derivatives. In 1985 the yearly production of Methyl Acetate was 17,350 tons and one production technique is esterification of methanol and acetic acid with water as by-product.

$$CH_3OH + CH_3COOH \Leftrightarrow CH_3COOCH_3 + H_2O$$

The principal side-reaction leads to small amounts of di-methyl Ether (DME) by the elimination reaction of two methanol molecules:

$$2\ CH_3OH \Leftrightarrow CH_3OCH_3 + H_2O$$

Reaction information can be obtained from Kirk-Othmer [1], Rönnback *et al.*[2], Pöpken *et al.* [3] and Song *et al.*[4]. Additional equilibrium mixture data is available in the Dortmund database. A search of the literature and in databases is necessary to collect all the relevant information. From the literature, at least six alternative flowsheets can be found (Siirola [5], Jaksland [6], Pilavachi & Sawistowski [7]). Examples of industrially used Mass Separation Agents for different separations can be found in Perry *et al.* [8]. A conventional flowsheet of the process is illustrated in Figure 1. In the separation step it involves extractive distillation, azeotropic distillation, a decanter and extraction. Based on an algorithm using thermodynamic insights Jaksland [6] proposed the alternative flowsheet shown in Figure 2 involving only four columns. The analysis involves the use of several computer aided tools. A tool is used for mixture analysis to identify binary and ternary azeotropes, liquid-liquid phase splits and distillation boundaries. Moreover a tool is used for solvent selection using Computer Aided Molecular Design (CAMD) technology. Several alternatives have been proposed applying both azeotropic, extractive and pressure swing distillation for the different separation tasks.

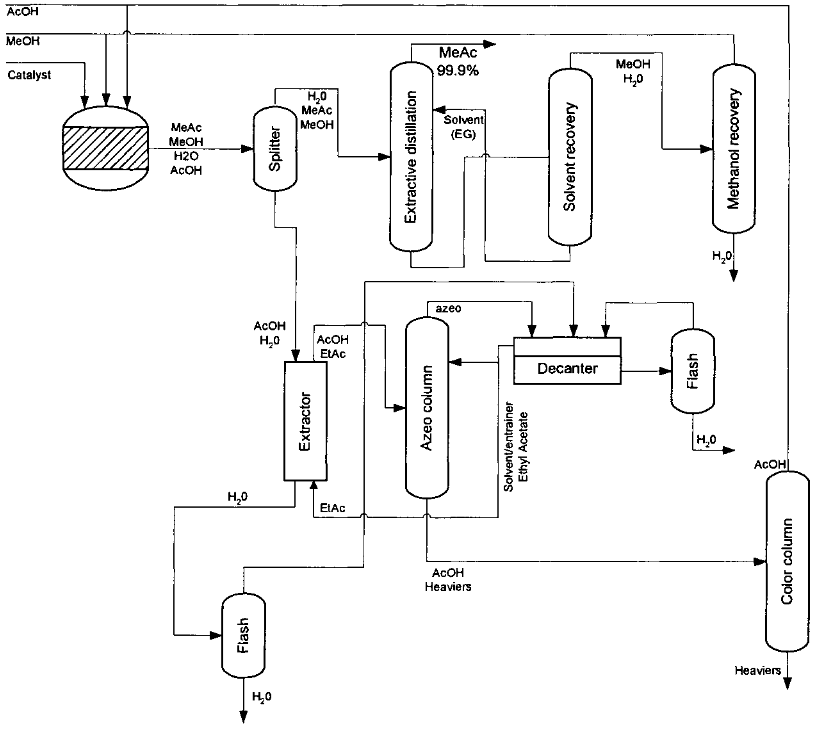


Figure 1: Conventional flowsheet for the methyl acetate process, Siirola [5]

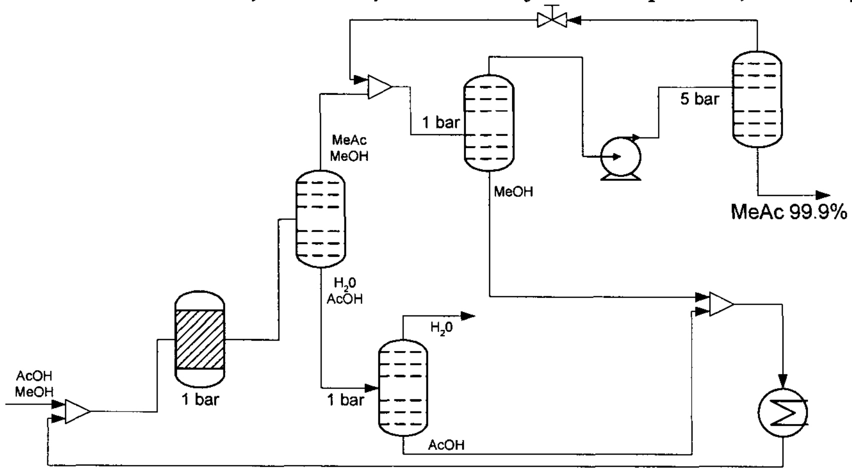


Figure 2: Alternative flowsheet for methyl acetate production proposed by Jaksland [6]

It is feasible to combine the reaction step and the separation step in a single reactive distillation column. An equilibrium-based reactive distillation column can be set up and simulated. It has been possible to obtain approximately 90%

pure methyl acetate as a product. When the reactive distillation column is combined with a conventional distillation column, high purity methyl acetate product can be obtained. The flowsheet in this case is just two distillation columns, one reactive and one non-reactive.

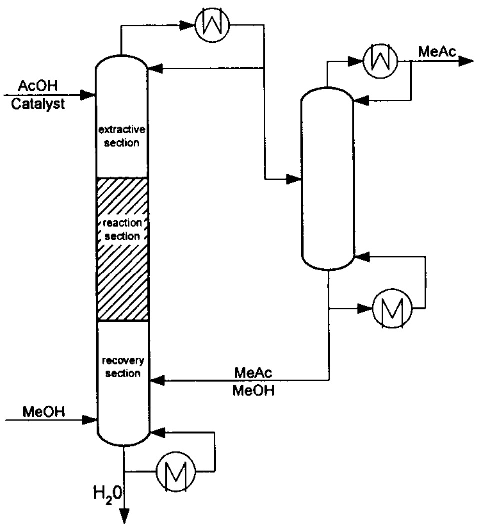


Figure 3: Methyl Acetate produced by reactive distillation

This case study is chosen for the following reasons:

- Many of the steps involved in the early phase of the conceptual design can be highlighted through this case study.
- Esterification processes are of industrial significance.
- This process has been well studied (at least six flowsheets proposed in the open literature).
- Kinetic reaction data is available.
- The system has complex physical behaviour (*e.g.* azeotropic mixtures).
- Several CAPE tools are required for the analysis.
- Both traditional and hybrid separation processes (reactive distillation) are involved.
-

7.1.2.2 Stages of the process design lifecycle

The activities, the scenarios, the data-flow, work-flow and the interface requirements for selected stages of the process design lifecycle related to conceptual design are highlighted in this section. A detailed description of other important stages of the process lifecycle, such as, studies related to operation and controllability, safety & hazards, environmental assessment, *etc.*, can be found in Bayer *et al.* [9]

Modelling of kinetic reactor

The main activity here is to prepare a kinetic model of a liquid phase reactor, which would be needed in process simulation for design and analysis. Figure 4a and 4b highlight the sequence of the main activities related to the development of a kinetic reactor.

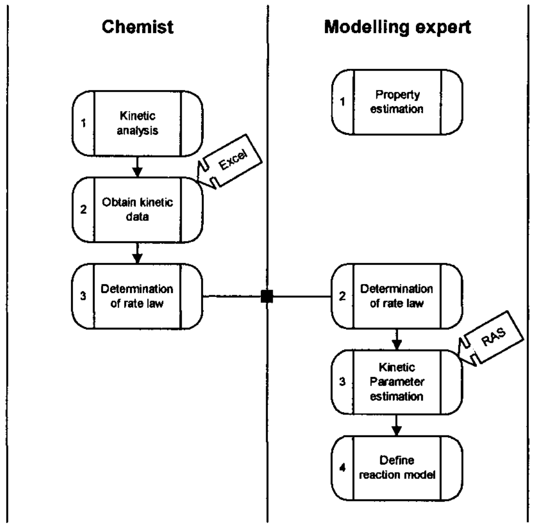


Figure 4a Work-flow for modelling of reaction kinetics

The starting point for this activity is the selection of the reaction path. Once the reaction path has been selected the kinetics must be studied. A kinetic analysis may be performed in the laboratory to produce kinetic data or the data might be obtained from literature. From data and physical insights a suitable rate law is determined and through kinetic parameter estimations the reaction model is defined. The reactor model can then be generated, for example, in a modelling testbed and used for simulations of the reactor operation. When a suitable model has been fixed it can be exported to a process simulator.

If a reaction model cannot be found in literature such a model may be generated based on experimental work. The Chemist needs access to the fundamental physical data of the involved compounds. These may be supplied from literature or from the modelling expert. The Chemist plans the experimental work and collects the data in a Spreadsheet together with date, experimental conditions, batch numbers, *etc.* From the behaviour with respect to concentration and temperature dependence, models are chosen for fitting the data. The parameters in the model are fitted using the experimental data, *i.e.* an interface is needed between the spreadsheet (or other data collection program) and the parameter estimation tool.

Having a kinetic model with fitted parameters we can predict the reaction behaviour within certain operational limits. This forms the basis of the reactor design. Possible reactor designs are investigated in co-operation with a reactor expert who provides the fundamental models for the different reactor types. The modelling expert combines the reactor models and the kinetic model (and possibly the physical properties model) to generate the reactor model. An interface is thus needed between the kinetic parameter estimation tool and the reactor modelling tool.

The reactor model is tested in the modelling tool by the reactor expert. The results from different runs should be collected in *e.g.* a spreadsheet, *i.e.* the modelling should be interfaced with a data collection tool. When the model is documented and has been tested and the reactor expert is satisfied with it, it should be exported as a GCO compliant unit to a process simulator. The modelling tool should therefore be interfaced with a process simulator (see Figure 4b). Table 1 summarizes the interface between different types of tools that may be used development of the kinetic model.

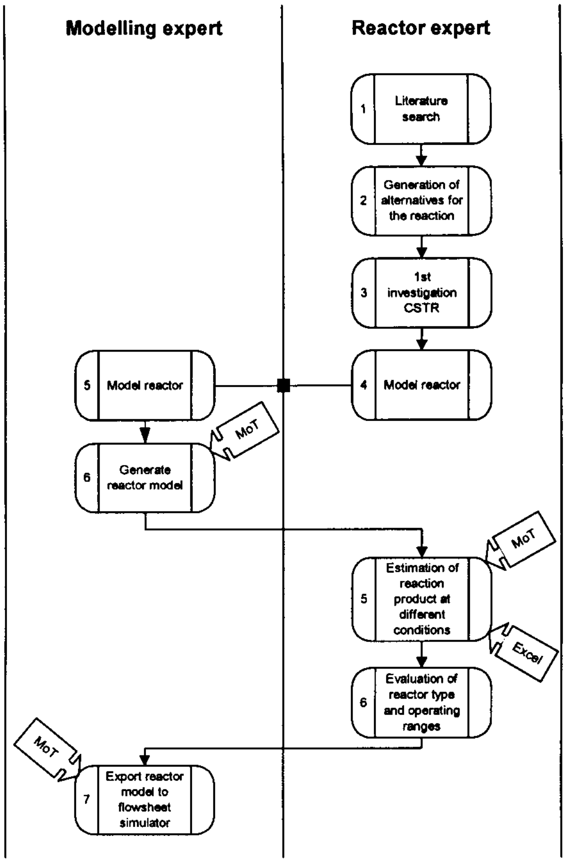

Modelling expert
Reactor expert
1 Literature search
2 Generation of alternatives for the reaction
3 1st investigation CSTR
4 Model reactor
5 Model reactor
6 Generate reactor model
MoT
5 Estimation of reaction product at different conditions
MoT
Excel
6 Evaluation of reactor type and operating ranges
MoT
7 Export reactor model to flowsheet simulator

Figure 4c Simulation and evaluation of reactor

Table 1. Interface between different types of tools

Tool A	**Tool B**	**A → B**	**B → A**
Excel	Kinetic parameter estimation tool	Experimental data	
User	Kinetic parameter estimation tool	Kinetic model	
Kinetic parameter estimation tool	Modelling tool	Kinetic model	-
User	Modelling tool	Reactor equations	Results of reactor simulation
Modelling tool	Spreadsheet	Results of reactor simulation	
Modelling tool	Process simulator	Unit model	

Definition of thermodynamic model

The main activity here is to set up a thermodynamic model, which can predict the necessary physical properties of the individual pure components and their mixtures. The activity diagram and thesummary of interface requirements are highlighted through Figure 5 and Table 2, respectively.

In order to model the process an accurate model of the physical behaviours is essential. Pure component and mixture properties are taken either from literature or experimental data. If literature data cannot be found and experimental work is constrained by time and/or money, physical properties can be estimated using CAPE tools. From an analysis of the process type and operating conditions a suitable properties model package can be selected. Relevant property model parameters can be fitted and/or tuned from experimental data using a tool for parameter estimation.

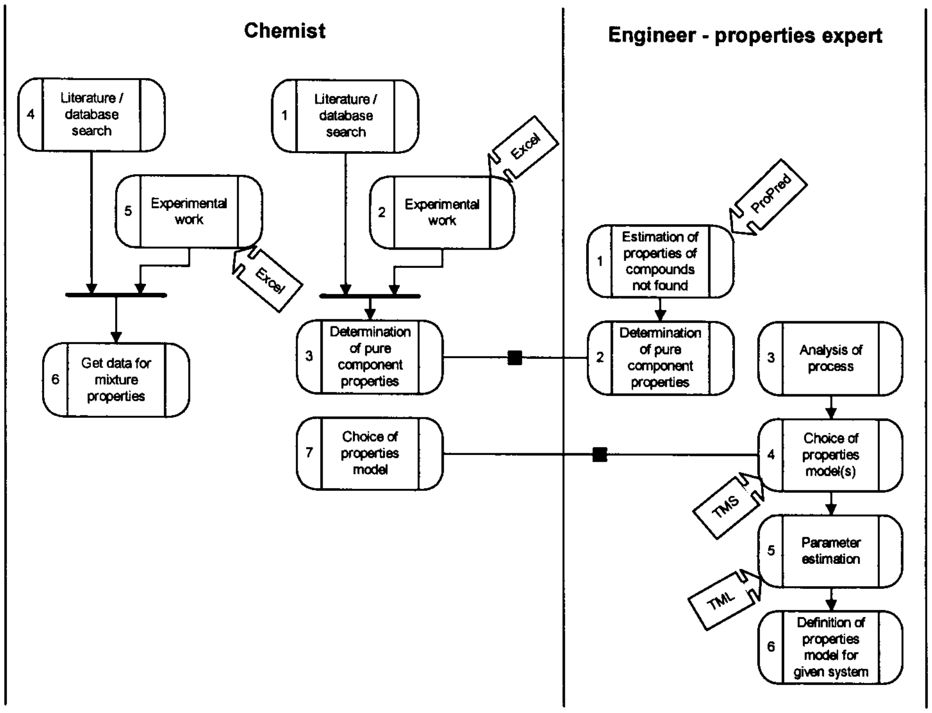

Chemist
Engineer - properties expert
4 Literature / database search
5 Experimental work
6 Get data for mixture properties
1 Literature / database search
2 Experimental work
3 Determination of pure component properties
7 Choice of properties model
Excel
Excel
1 Estimation of properties of compounds not found
ProPred
2 Determination of pure component properties
3 Analysis of process
4 Choice of properties model(s)
TMS
5 Parameter estimation
TML
6 Definition of properties model for given system

Figure 5. Selection of Thermodynamic Model

Many pure components are documented in the literature and the work will therefore start with a literature/database search. If the properties of a compound are not known, experimental work can be planned and performed. Data will often be collected in a spreadsheet together with details of date, batch numbers, conditions, *etc.* The experimental data can be fitted to the needed correlations using a parameter estimation program. Thus, an interface is needed between the spreadsheet (process model) and the parameter estimation program. Furthermore an interface is needed between the parameter estimation program and the compound database. Estimates of pure component properties can be obtained by using suitable CAPE tools. An interface is therefore also needed between the property prediction tool and the compound database.

Table 2: Interface requirements for property prediction tools

Tool A	**Tool B**	**A → B**	**B → A**
User	Excel	Experimental data	
User	TMS (Thermodynamic Model Selection)	Compounds Mixture composition Type of operation Temperature and	Suggested thermodynamic models Literature references

Tool A	Tool B	A → B	B → A
		pressure range	
TMS	TML (Thermodynamic Model Library)	Choice of properties model	
Excel	TML	Tables with experimental data	
User	TML	Objective function Weights and scaling factors Parameters to be fitted Optimisation method	Interaction parameters Residual plots
TML	Compound database	Pure component properties Mixture interaction parameters	Mixture interaction parameters Vapour pressure parameters
User	ProPred (Pure component property prediction)	Molecular structure Estimation method	Primary properties Secondary properties (functions of more than one property) Functional (functions of T and P)
ProPred	Compound database	Pure component properties	

The next important step is to describe the properties of the mixture. First of all a literature search takes place in order to determine if interaction parameters are already available. If not experimental work needs to be planned and performed. A chemist and the properties expert discuss the choice of properties model from the knowledge of mixture behaviour and which types of calculations are involved, for example, phase diagrams. An expert system can assist in the choice of properties model. The model parameters can then be fitted to the experimental data using a parameter estimation tool. An interface between the expert system and the parameter estimation program is useful. Mixture interaction parameters are transferred to the compound database, and an interface is also needed for this.

Selection of separation sequence

The principal activity here is to determine the separation sequence. The activity diagram and the summary of the interface requirements are highlighted through Figure 6 and Table 3, respectively.

A driving force based technique for process synthesis that determines a near optimal separation sequence is determined. First mixture data is calculated from a properties model. From this data the driving force (or secondary separation efficiency) curves are generated for each adjacent pair of key components at uniform pressure. This identifies feasible separation techniques and conditions of operation. Mixture data is analysed using ternary phase diagrams and residue curves. Scaling factors are applied to penalise non-sharp separations and compensate for diluted feed in the driving force curves. New driving force curves are calculated and the separation sequence is determined on this background.

Initial information for the separation expert is the compounds present, an appropriate properties model and an estimated pressure throughout the system. Necessary data can be generated using information from a compound database and a tool for validation of generated process alternatives, for example, a process simulator. An interface is required between the compound database and the simulation tool. The phase composition data can be transformed into a composition driving force diagram. An interface is required between the simulation tool and a graphical tool. For many systems ternary phase diagrams with residue curves are also generated. An interface is therefore also required to transfer compound and property model information from the process simulator to this tool.

The separation expert analyses the composition driving force diagrams and ternary phase diagrams in order to determine possible products, phase boundaries, ternary azeotropes, feasible separation techniques, feasible conditions of operation and evaluates the sizes of composition driving forces. From this information the separation sequence (including hybrid separation techniques) is determined. Both distillation columns and other unit operations can be included. In case of a distillation column sequence, the first column is analysed using a tool for Process Design (PDS). From information of compounds, property model, pressure, equilibrium data, column feed composition, desired product composition, the reflux ratio and the minimum reflux ratio, PDS generates scaling factors, determines near optimal feed plate location, generates operating lines and estimates the number of stages required. Thus, an interface is required between PDS and the process simulator for the validation phase of the design.

The remaining columns in the separation sequence are then investigated in the same manner using PDS. When all the columns in the separation sequence have been designed, all the design data can be transferred to the process simulator through a suitable interface. The separation system is then verified by simulation using appropriate process models.

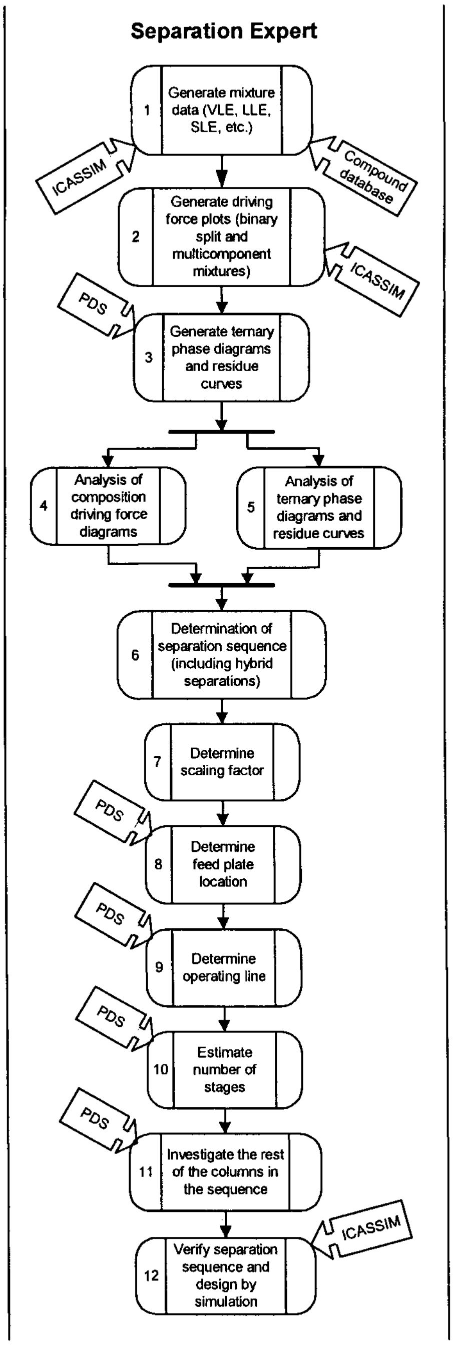

Separation Expert
1 Generate mixture data (VLE, LLE, SLE, etc.)
ICASSIM
Compound database
2 Generate driving force plots (binary split and multicomponent mixtures)
ICASSIM
PDS
3 Generate ternary phase diagrams and residue curves
4 Analysis of composition driving force diagrams
5 Analysis of ternary phase diagrams and residue curves
6 Determination of separation sequence (including hybrid separations)
7 Determine scaling factor
PDS
8 Determine feed plate location
PDS
9 Determine operating line
PDS
10 Estimate number of stages
PDS
11 Investigate the rest of the columns in the sequence
ICASSIM
12 Verify separation sequence and design by simulation

Figure 6. Separation sequence

Table 3. Interface requirements for software tools for separation process design

Tool A	Tool B	A → B	B → A
User	ICASSIM (Process Simulator)	Compounds Properties model Pressure	Phase composition data Driving force plots
Compound database	ICASSIM	Pure component properties Mixture interaction parameters	
User	PDS (Process Design Studio)		Ternary phase diagrams Azeotropes (binary and ternary) Heterogeneous liquid boiling surface Residue curves Distillation boundaries Liquid miscibility
ICASSIM	PDS	Compounds Properties model Pressure	
User	PDS	Column feed composition Desired product composition R / R_{min}	Feed plate location Operating lines Estimated number of stages
PDS	ICASSIM	Topology of column sequence Distillation column designs: Feed plate location Number of stages Reflux ratio	

Design of an extractive distillation system

The main activity here is to design an extractive distillation system for separation of Methyl Acetate from Methanol. A suitable entrainer also needs to be found (selected) and the optimal design of the column system is to be determined. Figures 8a – 8c show the activity diagrams for this step. The interfaces between tools are summarized in the Table 4.

From a mixture analysis a list of candidate entrainers is found. Feasibility is checked by binary and ternary mixture analysis. An initial column design is proposed by a distillation design expert and a suitable steady state model for the column is determined. The extractive column is first modelled in a steady state simulator. An optimisation problem is formulated and the extractive column is optimised. Then the solvent regeneration column is added to the simulation and the system re-optimised. By using the different candidate entrainers, the most suitable entrainer can be chosen.

For extractive distillation a suitable solvent is needed. From the problem definition and the compounds, which need to be separated, a literature study is performed by the properties expert to identify already used solvents. A number of constraints on physical properties are then formulated, *e.g.* boiling point range, azeotrope behaviour, *etc.* Constraints can either be formulated in order to find a solvent with equivalent properties to an already known solvent or in order to fulfil some desired properties, *e.g.*, high selectivity and high solvent power. A list of feasible solvents is generated using a Computer Aided Molecular Design (CAMD) tool. This list is then screened with respect to availability, health and safety aspects, operational aspects and solvent price. From this analysis a list of feasible candidate solvents are generated for further analysis.

The candidate solvents are analysed with respect to their mixture behaviour. Binary VLE diagrams are generated for the different combinations of solvents and mixture components. Ternary phase diagrams are generated to identify binary and ternary azeotropes, liquid-liquid phase splits, distillation boundaries and residue curves. An interface is required between the CAMD tool and the binary and ternary phase diagram analysis tool. From the analysis of the mixture behaviour of the different solvents the most promising candidate solvents are chosen.

The separation objective and the feed flow conditions are used by the distillation expert in order to determine column type and approximate column parameters (number of plates, feed plate locations) and a starting point for column parameters (reflux ratio, vapour boil-up, etc.). This information is used by the simulation and optimisation expert to set up and simulate a steady state column model.

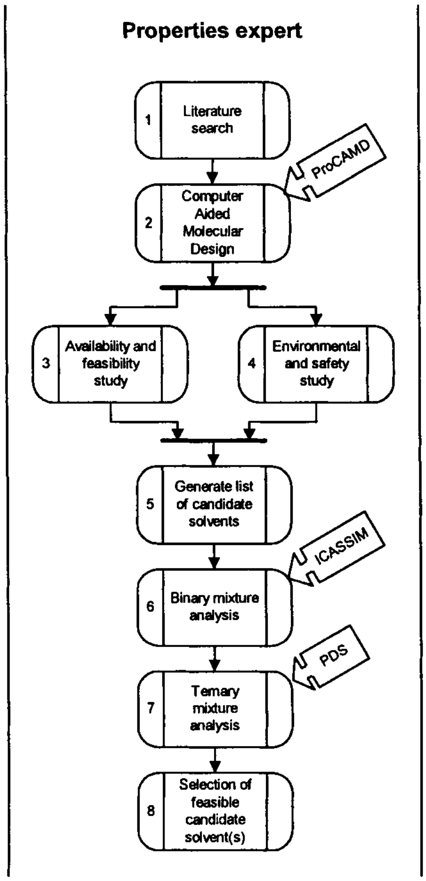

Properties expert
1 Literature search
2 Computer Aided Molecular Design
ProCAMD
3 Availability and feasibility study
4 Environmental and safety study
5 Generate list of candidate solvents
ICASSIM
6 Binary mixture analysis
PDS
7 Ternary mixture analysis
8 Selection of feasible candidate solvent(s)

Figure 8a. Extractive distillation -- properties expert

The simulation and optimisation expert formulates and solves a steady state simulation of the extractive column using the information from the distillation expert. An interface between the CAMD tool and the process simulator is useful. An optimisation problem is formulated, which can include capital cost models, utility cost, solvent price and constraints on operation and purities. From knowledge of feasible control variables, appropriate optimisation variables can be selected. The optimisation is formulated mathematically in the optimisation solver. An interface between the process simulator and the optimisation solver is required in order to include the column model in the optimisation problem. The column model operating under 'optimal' conditions is then extended with a recovery column, which is used to recycle the solvent back to the extractive column. The simulation model is updated and the optimisation problem is extended to incorporate the new column. The outcome is a steady state simulation model of an extractive distillation system with optimal operating conditions.

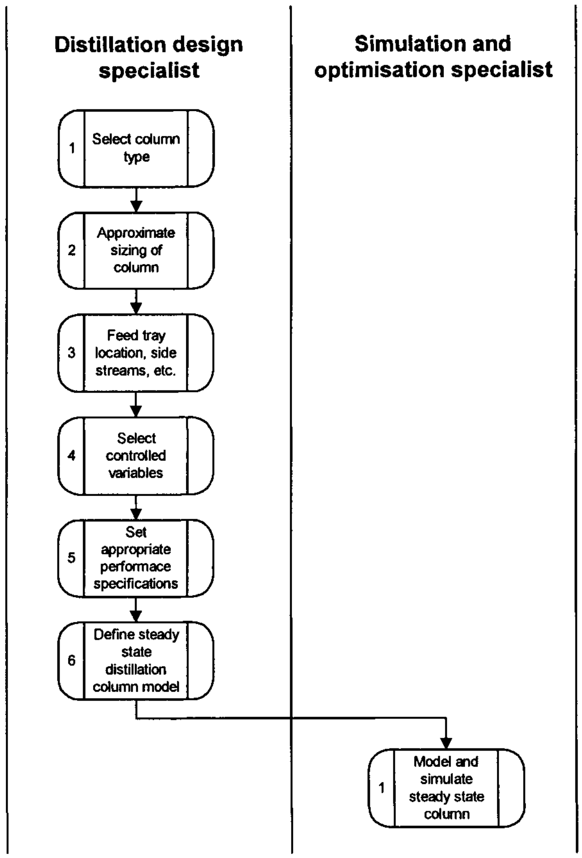

Distillation design specialist
Simulation and optimisation specialist
1 Select column type
2 Approximate sizing of column
3 Feed tray location, side streams, etc.
4 Select controlled variables
5 Set appropriate performace specifications
6 Define steady state distillation column model
1 Model and simulate steady state column

Figure 8b. Extractive distillation -- distillation design specialist

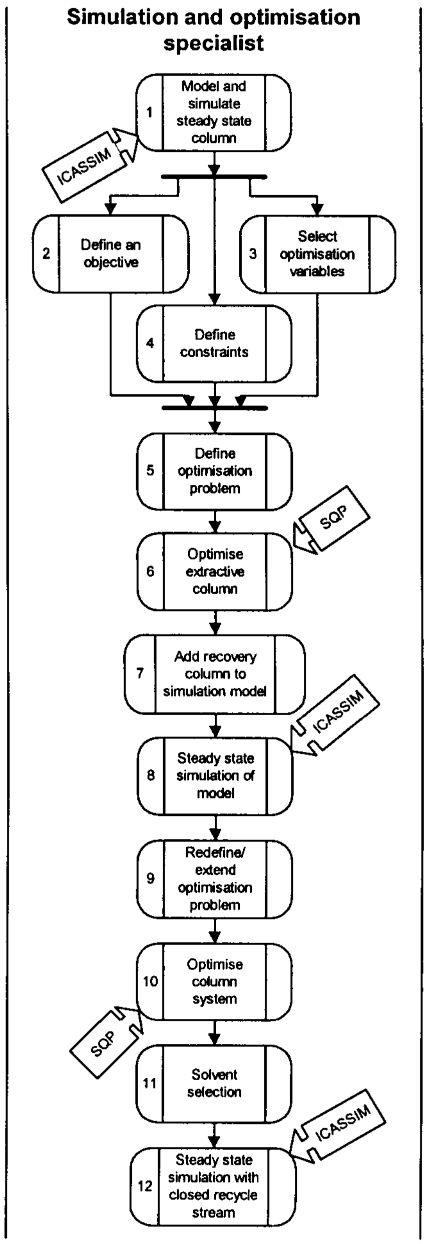

Simulation and optimisation specialist
1 Model and simulate steady state column
ICASSIM
2 Define an objective
3 Select optimisation variables
4 Define constraints
5 Define optimisation problem
SQP
6 Optimise extractive column
7 Add recovery column to simulation model
ICASSIM
8 Steady state simulation of model
9 Redefine/ extend optimisation problem
10 Optimise column system
SQP
11 Solvent selection
ICASSIM
12 Steady state simulation with closed recycle stream

Figure 8c. Extractive distillation - Simulation and optimisation expert

Table 4. Interface requirements for integrated process design

Tool A	Tool B	A → B	B → A
User	Process simulator	Components Reaction models Separation models Properties model Simulation strategy Numerical solver Units of measurements Flowsheet topology Feed streams Unit specifications	Unit and stream summary Convergence history
User	ProCAMD	Functional groups in molecule Thermodynamic model Solute Constraints:Normal boiling point;Selectivity; Solvent power Solvent loss	List of feasible solvents; Compound Database entry
ProCAMD	Process simulator	Feasible solvents	Mixture information
User	PDS	Thermodynamic model	Ternary phase diagrams Azeotropes (binary and ternary) Heterogeneous liquid boiling surface Residue curves Distillation boundaries Liquid miscibility
ProCAMD	PDS	Feasible solvents	
Compound database	PDS	UNIFAC parameters	
Process simulator	PDS	Mixture information Pressure	
User	SQP (optimiser)	Objective function Constraints Optimisation variables Solver settings Solution strategy	Optimal values of optimisation variables Value of objective function Values of constraint equations Lagrange multipliers Convergence history
Process simulator	SQP (optimiser)	Unit and stream information	Set points Simulation strategy

Simulation and optimisation of flowsheet

The main activities here are to set up and simulate a generated process flowsheet including the reactor, the separation system and the recycle streams and then to determine the optimal process conditions through a steady state optimisation. See Figure 9 and Table 5 for the activity diagram and the summary of the interface requirements, respectively.

A flowsheet topology has been suggested in the process synthesis stage and steady state simulation of the process will be performed. The flowsheet is set up using GCO compliant models (where these exist) and custom models. When steady state behaviour of the flowsheet has been determined we optimise the operating parameters to meet the objective within the given constraints. First we need to formulate an appropriate objective function and formalise the process constraints. Next we need to identify appropriate optimisation variables. The process can then be formulated in the simulation environment and optimised using a static optimisation algorithm. Finally, the result of the optimisation algorithm can be analysed by doing a sensitivity analysis and by analysing the effect of the active constrains (Lagrange multipliers).

A flowsheet is built in the process simulation environment by the simulation and optimisation expert. Steady state simulations are performed to investigate static behaviour. An optimisation problem is defined, which can include capital cost functions, operating cost functions and revenue functions, plus a number of constraints. The optimisation problem is integrated with the simulation problem, thus an interface is required between the flowsheet simulator and the optimiser. The optimiser will return the optimal set of optimisation variables and the value of the objective functions. Furthermore the Lagrange variables are given for sensitivity analysis.

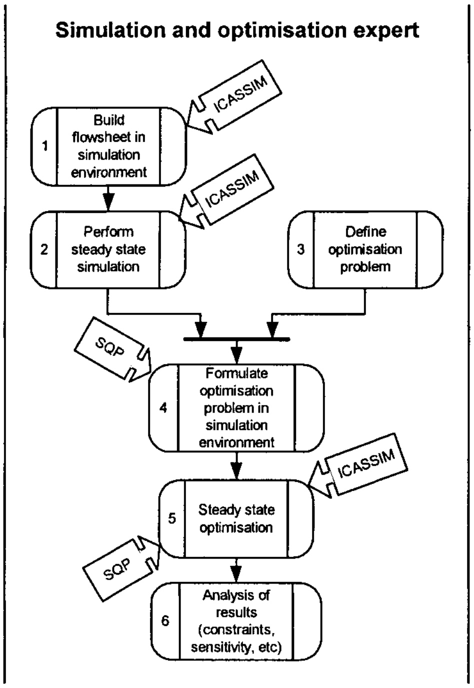

Simulation and optimisation expert
1
Build flowsheet in simulation environment
ICASSIM
2
Perform steady state simulation
ICASSIM
3
Define optimisation problem
SQP
4
Formulate optimisation problem in simulation environment
ICASSIM
5
Steady state optimisation
SQP
6
Analysis of results (constraints, sensitivity, etc)

Figure 9. Information flows during simulation and optimization

Table 5. Interface requirements for process simulation & optimisation

Tool A	Tool B	A → B	B → A
User	Process simulator	Components Reaction models Separation models Properties model Simulation strategy Numerical solver Units of measurements Flowsheet topology Feed streams Unit specifications	Unit and stream summary Convergence history
Process simulator	Optimiser	Simulation results (state variables)	Set-points (controlled variables)
User	SQP (optimiser)	Objective function Constraints Optimisation variables Solver settings Solution strategy	Optimal values of optimisation variables Value of objective function Values of constraint equations Lagrange multipliers Convergence history

7.1.3 CASE STUDY: POLYAMIDE6

7.1.3.1 Introduction

This case study describes the design process of a chemical plant for the production of polyamide6 including the polymer compounding and post-processing. It focuses on the workflow, the involved people and the used tools together with their interactions. This case study also forms the basis for Chapter 5.1, where it is employed to demonstrate the use of various mathematical models during process design.

The design task given in this case study is to produce 40 000 tons polyamide6 per year with a given product quality and specification.. Polyamide6 is produced by the polymerisation of •-caprolactam. There are two possible reaction mechanisms, the hydrolytic and the anionic polymerisation [10]. Since the anionic polymerisation is mainly used for special polymers [11], this case study looks at the hydrolytic polymerisation, which is also more often applied industrially. This polymerisation consists of three single reaction steps:

Ring opening of •-caprolactam:
$C_6H_{11}NO + H_2O \Leftrightarrow C_6H_{13}NO_2$

Poly-condensation:
$$H[NH-(CH_2)_5-C=O]_m OH + H[NH-(CH_2)_5-C=O]_n OH$$
$$\Leftrightarrow H[NH-(CH_2)_5-C=O]_{m+n} OH + H_2O$$

Poly-addition:
$$H[NH-(CH_2)_5-C=O]_n OH + C_6H_{12}NO \Leftrightarrow H[NH-(CH_2)_5-C=O]_{n+1} OH$$

There are different kinds of reactors that can be used for the polymerisation: sequences of two or more tank reactors or plug flow reactors [12] and a special reactor developed for the polyamide6 production, the VK-tube [13]. The polymer melt at the outlet of the reactor contains monomer, oligomers and water, which need to be removed in order to meet the required product quality. Therefore, a separation is needed. Two separation mechanisms can be used here: The evaporation in a wiped-film-evaporator (WFE) or extraction with water to remove •–caprolactam with a successive drying step [14]. Polymer post-processing is done within an extruder: Additives, fillers and fibres are added in order to meet the specified product qualities. Within the extruder, an additional polymerisation of the melt and the degassing of volatile components are possible.

These different alternatives for reaction, separation and extrusion lead to several alternative processes for the polyamide6 production. In Figure 10, a flowsheet of one process alternative is given: the reaction takes place within two reactors;

separation is done by leaching and drying of polymer pellets. The cleaned pellets are re-melted in the extruder so that additives can be added.

The case study covers the design of the reaction and separation system as well as the extrusion. For all process units, mathematical models are developed and used for simulation within different simulators. Because the design requires very specific knowledge, each unit is studied in detail by different experts.

This case study was chosen for the following reasons:

- The design process covers many steps performed by different roles within the conceptual process design phase.
- Computer-aided process engineering for polymer processes is still a novel research area with many potentials for improvements.
- Through the integration of the compounding and post-processing cooperation between actors from different companies and backgrounds can be examined.
- The polyamide6 production is a well studied process.
- Numerous data on the reaction mechanisms is available.

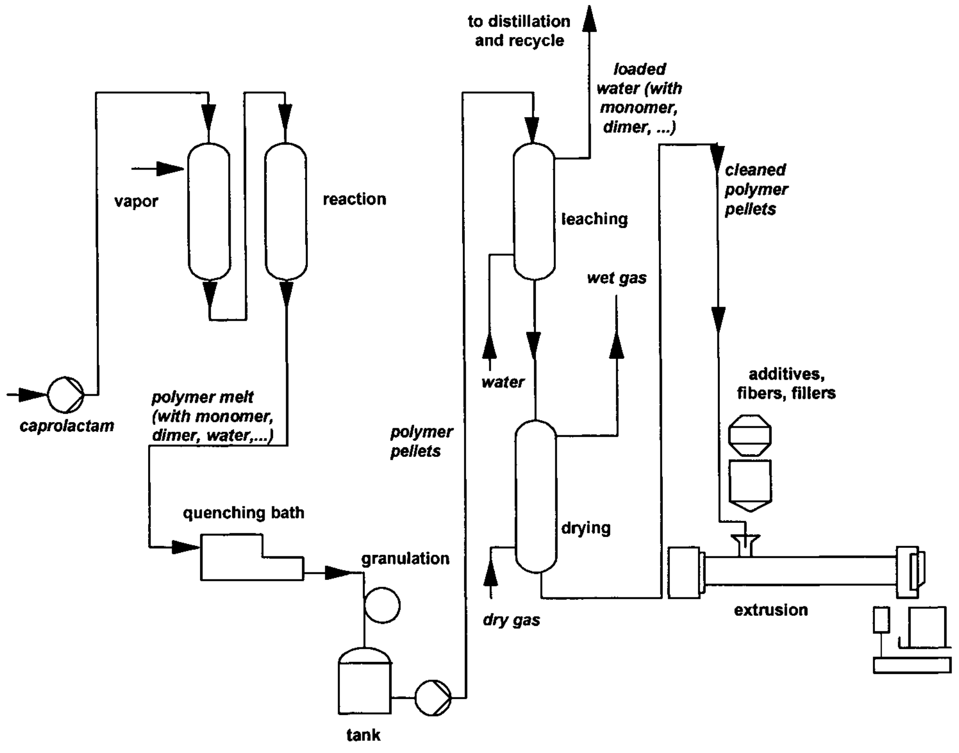

to distillation and recycle
loaded water (with monomer, dimer, ...)
cleaned polymer pellets
vapor
reaction
leaching
wet gas
water
additives, fibers, fillers
caprolactam
polymer melt (with monomer, dimer, water,...)
polymer pellets
quenching bath
drying
granulation
dry gas
extrusion
tank

Figure 10. Flowsheet for polyamide6 production.

7.1.3.2 Stages of the process design lifecycle

This case study covers mainly the phases of conceptual process design. Based on the design task, the overall structure of the process and the operation mode are set. Several design alternatives are synthesized, analysed, and evaluated. Finally, decisions for the different process sections are taken and the plant concept is set comprising a process description, process flow diagrams together with mass and energy balances and data sheets for major units.

The different stages of this case study from the project start to the decision for a plant concept are described in some detail in the following. The focus is set on technical activities, roles, used tools, and information flows; management activities are mainly unconsidered. This design process is presented in Chapter 5.1 from a modelling perspective.

Literature survey

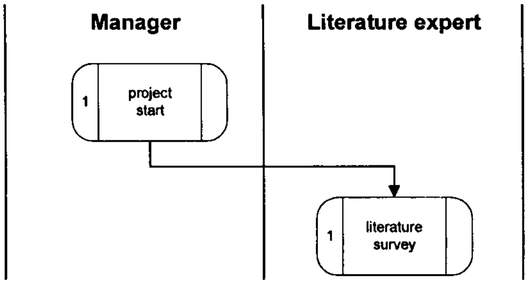


Figure 11. Literature survey.

The manager defines the design task; he places a team responsible for the project and sets a first project schedule. The first step after this project start is to gather information about the process, which is going to be developed, its underlying chemistry, and the occurring components. This is done by a literature expert (not a librarian, but a person with a technical background working on literature surveys and research) within a literature survey (see Figure 12). The information collected is available to all members of the design team.

The output information of that survey is a collection of books, articles, and patents. It covers information like:

- Physical properties of the occurring substances: •–caprolactam, water, polyamide6, and others;
- existing industrial processes (process flow diagrams);
- reactions and their kinetics;
- occurring side reactions; and
- relation between viscosity and molecular weight.

Preliminary flowsheet synthesis

Based on the results of the literature survey, some preliminary flowsheets are created. At this stage of the design process, simple block flow diagram are used. For their creation information about the occurring components, reactions including stoichiometry, reaction conditions as well as information about the product distribution and conversion is needed. This information is part of the results of the literature survey. Furthermore, the information about the desired production amount and the product quality is needed. Based on that information, the technical project leader creates preliminary flowsheets for the two alternatives for the process within a flowsheet editor (FE): one for continuous operation mode and one for batch mode (Figure 13).

The next step in the scenario is the evaluation of these two alternatives. The economic potential for both preliminary flowsheets is calculated using a simple model based on the actual material prices, the product distribution within the different reactors and the assumption of complete separations, which is implemented in EXCEL. A comparison shows that for the required amount of 40000 t/a the continuous mode is more economic. Further information that lead to that decision are heuristics about the pros and cons of batch and continuous processes.

In the following, the reaction and the separation section are studied in parallel by a reaction and a separation expert.

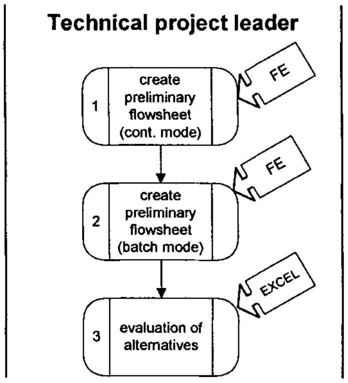

Technical project leader
1
create preliminary flowsheet (cont. mode)
FE
2
create preliminary flowsheet (batch mode)
FE
3
evaluation of alternatives
EXCEL

Figure 13. Preliminary flowsheet synthesis.

Table 6. Information flows during preliminary flowsheet synthesis.

A	B	A → B	B → A
User	Flowsheet editor	Flowsheets (structure, elements)	
Flowsheet editor	EXCEL	Flowsheets (structure, elements)	
User	EXCEL	Product amount Product distribution Separation factors Material prices Model for economic potential	Economic potential

Synthesis of reactor alternatives

The reaction expert starts with the generation of some possible realizations of the polymerisation reaction. His activities during this synthesis step are shown in Figure 14. For the generation of reasonable alternatives and their analysis, the he needs some additional information about the hydrolytic polymerisation of polyamide6. He formulates a demand for some additional literature to the literature expert, who performs a corresponding survey (see Figure 4). This literature contains the information that there are two principle alternatives for the realization of a polymer reaction [15] and that industrial polyamide6 productions are mainly carried out in a sequence of two or more reactors[16].

The reaction expert creates four alternatives for the reaction section of the preliminary flowsheet using the flowsheet editor: a CSTR, a PFR, and two reactor sequences (CSTR-CSTR and CSTR-PFR). The reactor sequence consisting of a CSTR and a PFR corresponds to the reaction sequence conducted in a VK-tube. The input information for this activity are the preliminary flowsheet, the results from the reaction literature survey and the expert's knowledge about reaction processes.

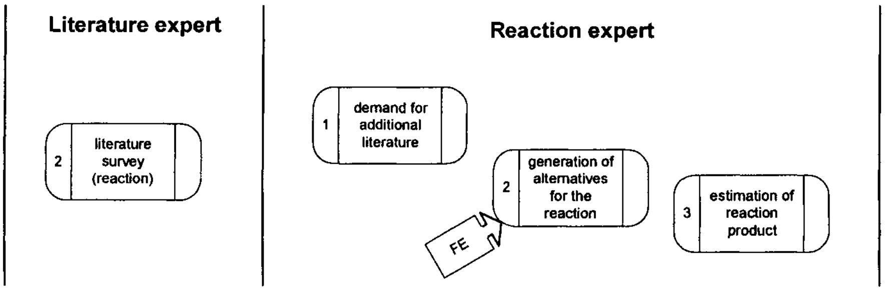

Literature expert
Reaction expert
1
demand for additional literature
2
literature survey (reaction)
2
generation of alternatives for the reaction
FE
3
estimation of reaction product

Figure 14. Synthesis of reactor alternatives.

The separation expert needs some information about the material stream at the reactor outlet. Therefore, based on the knowledge about occurring components and reactions, stoichiometry and conversions (obtained from the two literature surveys), the reaction expert makes an estimation of the reaction product (stream information), so that the separation expert can start working immediately.

Table 7. Information flows during synthesis of reactor alternatives.

A	B	A → B	B → A
User	Flowsheet editor	Flowsheets with 4 reactor alternatives (structure, elements)	

Analysis of reactor alternatives with one reactor

First simulations only involve one reactor to understand the basic effects and to compare the different reactor types. The reaction expert starts with the simulation of the CSTR (1st investigation CSTR, Figure 15) because these simulations are easier to realize. For the simulation of the reactor Polymers Plus from Aspen Tech is used [17]. The input for specifying the simulation model comprises knowledge about the occurring reactions, reaction conditions, and kinetic data. Thermodynamic data can be obtained from the Polymers Plus properties database; the user adds data for one specific component, the cyclic dimer of •-caprolactam, since for this component no data is available in Polymers Plus. The reaction expert makes some assumptions to simplify the calculations (reactor volume, neglecting the vapour phase).

Doing some simulation studies, he tries to meet the given product specification. Results are operating conditions for the CSTR, reactor size, stream results, some graphs obtained from sensitivity analyses and the simulation files themselves. Based on the stream results, the reaction expert updates the information about the reaction product, which builds the basis for the separation design.

As simulations for polymers are not very sophisticated yet, they have to be validated by experiments. When the 1st investigation of the CSTR is finished, the reaction expert contacts the laboratory director to plan some experiments on the polymerisation reaction. They discuss which experiments should be performed to obtain some information about the polymerisation of •-caprolactam in a CSTR as well as a schedule. The laboratory director uses his knowledge about experimental techniques for the characterization of polymers during that discussion. These experiments are performed at laboratory-scale and the results are passed to the reactor expert. He compares them with the simulations and recognizes significant differences. In a discussion with the laboratory director the reaction expert finds out, that it was a mistake during the simulation not to consider the vapour phase in the polymerisation reactor. A second investigation of the CSTR within Polymers Plus follows.

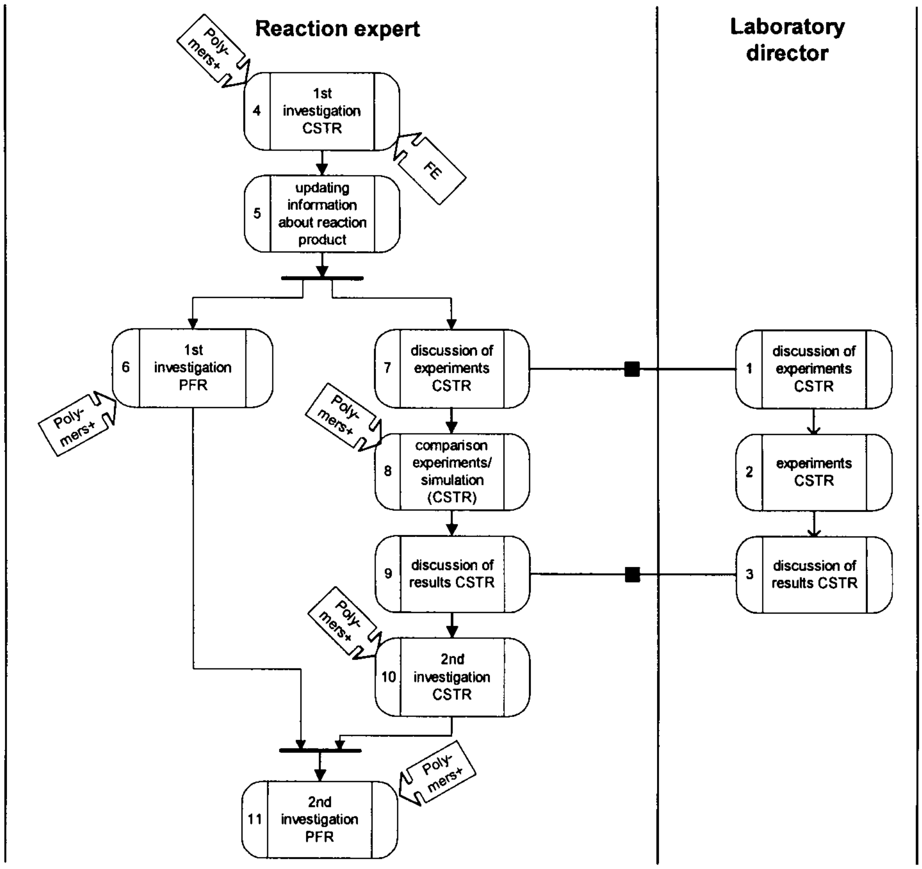

Reaction expert
Laboratory director
Poly-mers+
4 1st investigation CSTR
FE
5 updating information about reaction product
6 1st investigation PFR
7 discussion of experiments CSTR
8 comparison experiments/ simulation (CSTR)
9 discussion of results CSTR
10 2nd investigation CSTR
11 2nd investigation PFR
1 discussion of experiments CSTR
2 experiments CSTR
3 discussion of results CSTR

Figure 15. Analysis of reactor alternatives I.

In parallel to the experiments with a CSTR, the reaction expert has started with the first investigation of the PFR within Polymers Plus based on the results obtained from the 1st investigation of the CSTR (which leads to an "information flow" from Polymers Plus to Polymers Plus as given in Table 8). Again, thermodynamic data and the given product specification are used. Results are operating conditions for the PFR, reactor size, stream results, some graphs obtained from sensitivity analysis and the simulation files themselves. Since the reaction expert made the same assumption as in the CSTR simulation (no vapour phase), a second investigation of the PFR is also necessary, where the experimental results of the CSTR are taken into consideration.

Based on the results of the first investigations of the CSTR and the PFR and the experimental results, the reaction expert performs the 2nd investigations. The results are new operating conditions for the CSTR and the PFR, respectively, reactor sizes, stream results, new graphs from sensitivity analyses and modified simulation files.

Table 8. Information flows during analysis of reactor alternatives I.

A	B	A → B	B → A
Flowsheet editor	Polymers Plus	Flowsheets (structure, elements)	
User	Polymers Plus	Occurring reactions Reaction conditions Kinetic data Product specification Assumptions	Operating conditions Reactor size Stream information Graphs (sensitivity analyses) Input and output files
Properties data base (*e.g.* Polymers Plus)	Polymers Plus	Thermodynamic data	
Polymers Plus	Polymers Plus	Input and output files	Input and output files

Analysis of reactor sequences

Based on the simulations with one reactor, the reaction expert wants to perform some investigations on the two reactor alternatives consisting of two reactors. The investigations of the CSTR and PFR showed, that the molecular weight of the polymer depends on the water content within the reactor. Therefore, a separation between the reactors seems to be reasonable in order to attain the desired molecular weight. The reaction expert first discusses possible separations with the separation expert (see Figure 16). Based on the problem formulation (occurring components, operating conditions, separation task) and his knowledge about the separation of volatiles from polymer melts, the separation expert thinks that a separation of the water from the melt should be possible by flashing. According to that discussion result, the reaction expert creates two new flowsheet alternatives for the reaction in the flowsheet editor with a separation between the two reactors in the alternatives with the CSTR-CSTR and the CSTR-PFR sequence.

He performs some investigations of all reactor sequence alternatives (CSTR-CSTR, CSTR-PFR, CSTR-Sep-CSTR, CSTR-Sep-PFR); he uses his results, experiences and the files from the investigations of the CSTR and the PFR. For the Polymers Plus model of the separation simple splitter blocks are used as well as a flash model. The results are operating conditions for the CSTRs and the PFRs in the sequences, reactor sizes, stream results, new graphs from sensitivity analysis and new simulation files.

Based on the results of the investigation of all four reactor alternatives, the reaction expert estimates the costs of each alternative (capital investment costs as well as operating costs) using the ICARUS Process Evaluator [18]. The costs obtained from that estimation are major criteria for the decision for one reactor alternative.

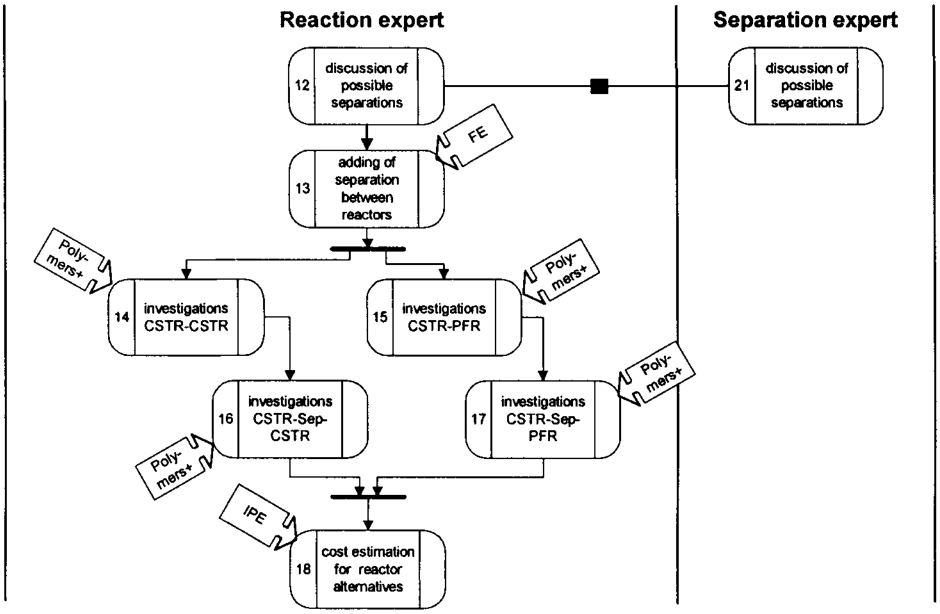

Reaction expert
Separation expert
12 discussion of possible separations
21 discussion of possible separations
13 adding of separation between reactors
FE
14 investigations CSTR-CSTR
15 investigations CSTR-PFR
16 investigations CSTR-Sep-CSTR
17 investigations CSTR-Sep-PFR
18 cost estimation for reactor alternatives
Poly-mers+
IPE

Figure 16. Analysis of reactor alternatives II.

Table 9. Information flows during analysis of reactor alternatives II.

A	B	A → B	B → A
User	Flowsheet Editor	Flowsheets (structure, elements)	
Flowsheet editor	Polymers Plus	Flowsheets (structure, elements)	
User	Polymers Plus	Reaction conditions Product specification	Operating conditions Reactor size Stream information Graphs (sensitivity analyses) Input and output files
Properties data base (*e.g.* Polymers Plus)	Polymers Plus	Thermodynamic data	
Polymers Plus	Polymers Plus	Input and output files	Input and output files
Polymers Plus	IPE	Operating conditions Reactor sizes Output files	
User	IPE		Costs for 4 reactor alternatives (operating, capital investment)

Synthesis of separation alternatives

In order to be able to synthesize and analyse some possible alternatives for the separation of •- caprolactam, oligomers and water out of the polymer melt, the separation expert needs additional information about polyamide6, water, monomer and oligomers and their physical properties (thermodynamic data, transport data). He demands some additional literature from the literature expert, who performs a corresponding literature survey and submits the results back (Figure 17).

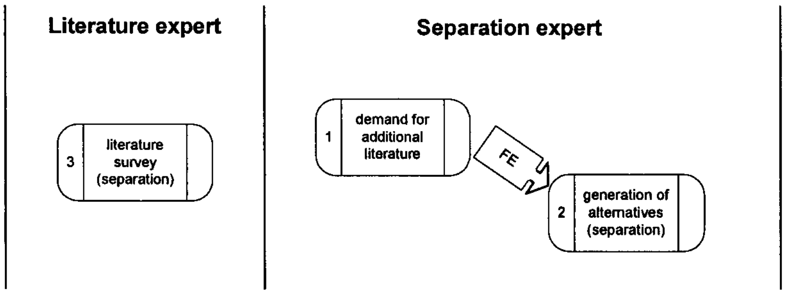

Literature expert
Separation expert
3 literature survey (separation)
1 demand for additional literature
FE
2 generation of alternatives (separation)

Figure 17. Separation alternatives.

Based on the information obtained from the additional literature survey and general knowledge about separation processes (flashing, evaporation, absorption, *etc.*), the separation expert synthesizes two alternatives for the process step of the removal of water, monomer and oligomers from the polymer within the preliminary flowsheet: the evaporation in a wiped-film-evaporator (WFE) and the extraction with water with a successive drying step.

Table 10. Information flow during synthesis of separation alternatives.

Tool A	Tool B	A → B	B → A
User	Flowsheet editor	Flowsheets with 2 separation alternatives (structure, elements)	

Analysis of separation in a WFE

The separation expert starts the investigation of the degassing with a simulation of that separation alternative using Polymers Plus. As an input stream into the separation system, he uses the estimated reaction product stream from the reaction expert; product amount and quality are constraints he has to regard. Since no predefined model for a WFE is available within Polymers Plus, the separation expert uses a flash simulation in which mass transfer limitations are neglected as a first approximation (1st investigation degassing; Figure 18). Product amount and quality, thermodynamic data, and the stream information about the reactor outlet from the reaction expert are the input data for that simulation. After this investigation, the separation expert comes to the end, that the flash model within Polymers Plus is not suitable, since the polymer melt is a very viscous fluid and the assumption of negligible mass transfer resistance is not tolerable.

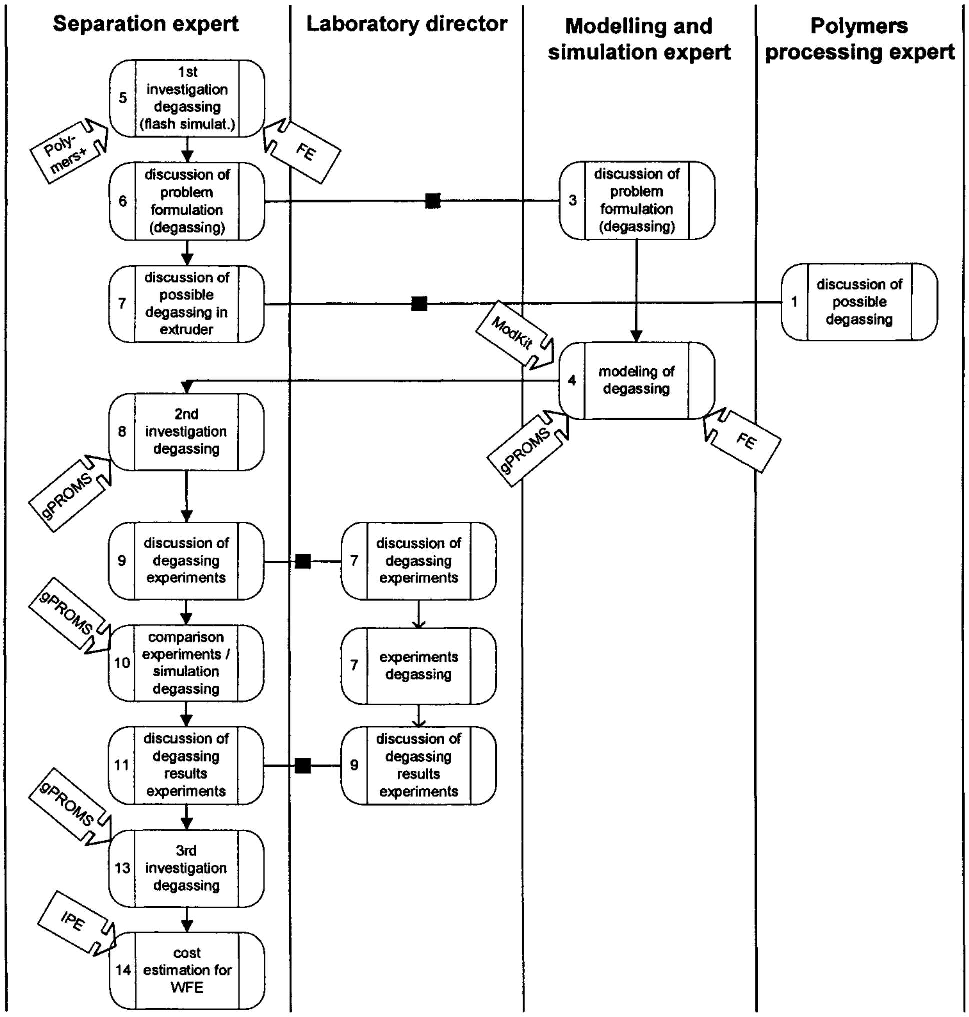

Separation expert
Laboratory director
Modelling and simulation expert
Polymers processing expert
5 1st investigation degassing (flash simulat.)
Poly-mers+
FE
6 discussion of problem formulation (degassing)
3 discussion of problem formulation (degassing)
7 discussion of possible degassing in extruder
1 discussion of possible degassing
ModKit
4 modeling of degassing
gPROMS
FE
8 2nd investigation degassing
gPROMS
9 discussion of degassing experiments
7 discussion of degassing experiments
gPROMS
10 comparison experiments / simulation degassing
7 experiments degassing
11 discussion of degassing results experiments
9 discussion of degassing results experiments
gPROMS
13 3rd investigation degassing
IPE
14 cost estimation for WFE

Figure 18. Analysis of the wiped film evaporator.

The separation expert discusses with the modelling and simulation expert the characteristics of the WFE and the requirements for a model. Based on an agreed problem formulation and all required inputs (flowsheet, production amount and quality, stream information, thermodynamic data), the modelling and simulation expert develops the model within the modelling tool ModKit [19]. From this, he generates a model that can be simulated with gPROMS [12].

During the first investigation of the degassing the separation expert realizes that it is difficult to reach the required purity of the polyamide6. He knows that degassing in the polymer extruder is a means to remove a certain content of volatiles. Therefore, this option should be considered already during the separation study. The separation expert discusses with the polymers processing expert, if it is possible to have a combination of the WFE and an extruder in which the polymer is degassed. They decide to take this alternative into consideration.

Using the ModKit model and the gPROMS file, the separation expert performs the second investigation of the WFE. He obtains simulation information about the operating conditions and equipment sizes of the WFE as well as sensitivities. The gPROMS model needs to be validated with experiments. The separation expert discusses with the laboratory director which experiments should be performed to obtain some information about that specific process step. They agree on a schedule; the experiments are performed at laboratory-scale. When the results are available, the separation expert compares them with the simulations. After a discussion with the laboratory director he adjusts the parameters of the gPROMS model in order to fit them to the experimental results. A third investigation of the WFE with the gPROMS model follows. Again, the simulation gives information about the WFE (operating conditions, equipment sizes); the information about the flow rate coming from the WFE is needed by the polymers processing expert.

Using all operating conditions, equipment data and other simulation results that can be obtained from the output files, the separation expert estimates the capital investment costs and the operating costs of the wiped film evaporator within the ICARUS Process Evaluator.

Table 12. Information flows during analysis of the WFE.

A	**B**	**A → B**	**B → A**
Flowsheet Editor	Polymers Plus	Flowsheets (structure, elements)	
User	Polymers Plus	Product amount and quality	
Properties data base (*e.g.* Polymers Plus)	Polymers Plus	Thermodynamic data	
Polymers Plus	Polymers Plus	Stream information reactor outlet	
Flowsheet Editor	ModKit	Flowsheets (structure, elements)	
User	ModKit	Stream information reactor outlet Product amount and quality	Model specification
Properties data base (*e.g.* Polymers Plus)	ModKit	Thermodynamic data	
ModKit	gPROMS	Model specification	
User	gPROMS	experimental results degassing	Operating conditions Equipment data Stream information Graphs (sensitivity analyses) Input and output files
gPROMS	IPE	Operating conditions Equipment data Output files	
User	IPE		Costs for WFE (capital investment, operating)

Analysis of separation with leaching

The separation expert wants to perform a first investigation of the leacher and the dryer, but for this alternative of the separation no model is available within Polymers Plus. Therefore, the separation expert discusses with the modelling and simulation expert the characteristics of the leaching and the requirements for the model. Based on an agreed problem formulation and all required inputs (flowsheet, production amount and quality, stream information, thermodynamic data), the modelling and simulation expert develops the model using the modelling tool ModKit. Using this ModKit model, which is described in detail in chapter 5.1.6, and the gPROMS file generated from it, the separation expert performs the second investigation of the leacher. These simulations bring information about the operating conditions and equipment sizes of the extraction. The gPROMS model of the leacher needs to be validated with experiments. The separation expert discusses with the laboratory director which experiments should be performed to obtain some information about that specific process step. They agree on a schedule. The experiments are performed at laboratory-scale. When the results are available the separation expert compares them with the simulation results. After a discussion with the laboratory director, he adjusts the parameters of the gPROMS model in order to fit them to the experimental results. A third investigation of the leacher with the gPROMS model follows. Again, the simulation gives information about the leacher (operating conditions, equipment sizes). These results are used for the estimation of capital investment costs and operating costs with the ICARUS Process Evaluator (IPE).

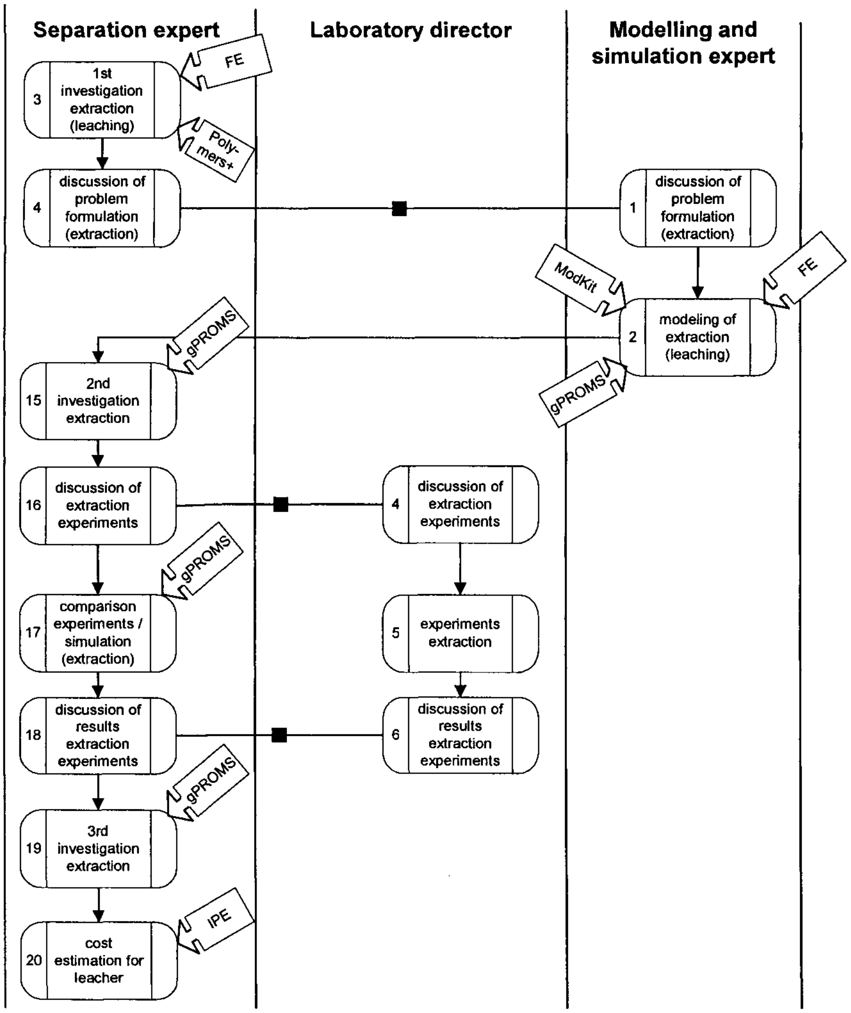

Separation expert
Laboratory director
Modelling and simulation expert
3 1st investigation extraction (leaching)
FE
Poly-mers+
4 discussion of problem formulation (extraction)
1 discussion of problem formulation (extraction)
ModKit
FE
2 modeling of extraction (leaching)
gPROMS
gPROMS
15 2nd investigation extraction
16 discussion of extraction experiments
4 discussion of extraction experiments
gPROMS
17 comparison experiments / simulation (extraction)
5 experiments extraction
18 discussion of results extraction experiments
6 discussion of results extraction experiments
gPROMS
19 3rd investigation extraction
IPE
20 cost estimation for leacher

Figure 19. Analysis of the leacher.

Table 13. Information flows during analysis of the leacher.

A	B	A → B	B → A
Flowsheet Editor	ModKit	Flowsheets (structure, elements)	
User	ModKit	Stream information reactor outlet Product amount and quality	Model specification
Properties data base (*e.g.* Polymers Plus)	ModKit	Thermodynamic data	
ModKit	GPROMS	Model specification	
User	GPROMS	Experimental results degassing	Operating conditions Equipment data Stream information Graphs (sensitivity analyses) Input and output files
gPROMS	IPE	Operating conditions Equipment data Output files	
User	IPE		Costs for leacher (capital investment, operating)

Extruder design

The polymers processing expert starts with extruder design after a discussion with the separation expert about the possibility of having a combination of the WFE and an extruder in which the polymer is degassed. Therefore, the study of the plastics processing parts is performed in parallel to the other studies. Conventionally, studies like this are carried out after the design of the reaction and separation sections is finished.

The polymers processing expert demands additional literature regarding material parameters of polyamide6 in order to be able to do some material planning. The literature expert performs the corresponding literature survey and passes the results back. Additionally, the polymers processing expert searches for information about additives, fillers and fibres that can be used for polyamide6 compounding.

The extruder design starts with an estimation of the impacts of the extrusion on the material (*e.g.* molecular weight) and determination of the necessary amount of additives and fibres (19, 22). Based on the flow rate of the polymer melt obtained from the WFE simulation, thermodynamic data, and the estimated fibre content, the process parameters of the extruder are calculated and an appropriate screw geometry is determined using MOREX, a tool for simulating the flow within twin screw extruders based on mathematical and physical models 15. Those are the input for the calculation of the degassing performance and for the simulation of the extruder with consideration of the degassing leading to operation conditions and equipment data for the extruder.

The results of the calculations need to be validated by experiments within an extruder in lab-scale. The required experiments are discussed between the polymers processing expert and the laboratory director. The experiments are performed. When the experimental results are available and discussed in order to understand them and the experimental conditions under which they were obtained, the polymers processing expert uses them to optimise the extrusion process and the screw geometry using MOREX.

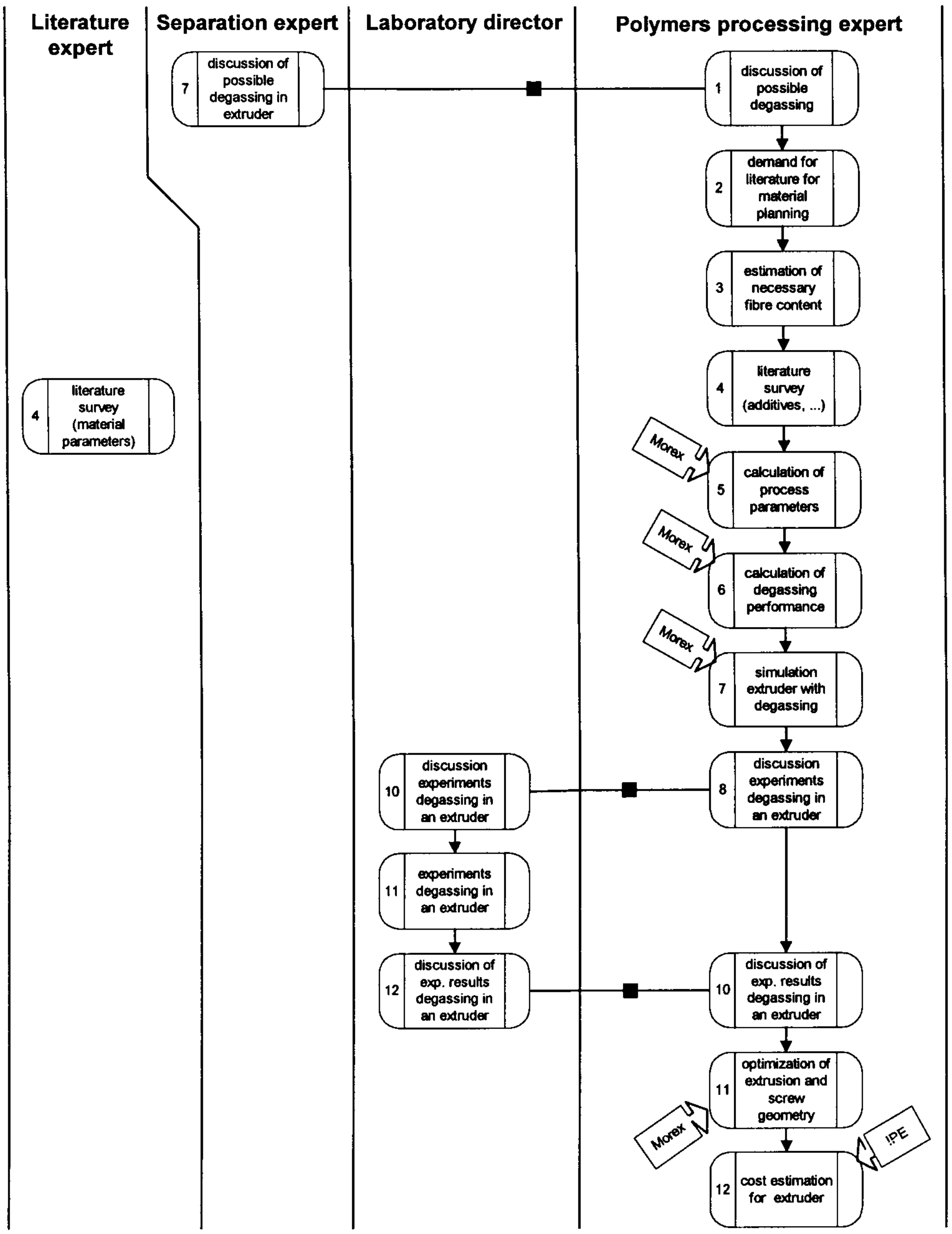

Literature expert
Separation expert
Laboratory director
Polymers processing expert
7 discussion of possible degassing in extruder
1 discussion of possible degassing
2 demand for literature for material planning
3 estimation of necessary fibre content
4 literature survey (material parameters)
4 literature survey (additives, ...)
Morex
5 calculation of process parameters
Morex
6 calculation of degassing performance
Morex
7 simulation extruder with degassing
10 discussion experiments degassing in an extruder
8 discussion experiments degassing in an extruder
11 experiments degassing in an extruder
12 discussion of exp. results degassing in an extruder
10 discussion of exp. results degassing in an extruder
11 optimization of extrusion and screw geometry
Morex
IPE
12 cost estimation for extruder

Figure 2. Extruder design.

Again, a simulation gives information about extruder design (operating conditions, extruder data). Based on this information and the output files, the polymers processing expert estimates the capital investment costs and the operating costs of the extruder.

Table 14. Information flows during extruder design.

A	B	A → B	B → A
Properties data base (*e.g.* Polymers Plus)	MOREX	Thermodynamic data	
User	MOREX	Product amount and quality Stream information reactor outlet Experimental results degassing Est. fibre content Information about additives, ...	Process parameters Operating conditions Extruder data Input and output files
MOREX	IPE	Operating conditions Extruder data	
User	IPE		Costs for extruder (capital investment, operating)

Decision for plant concept

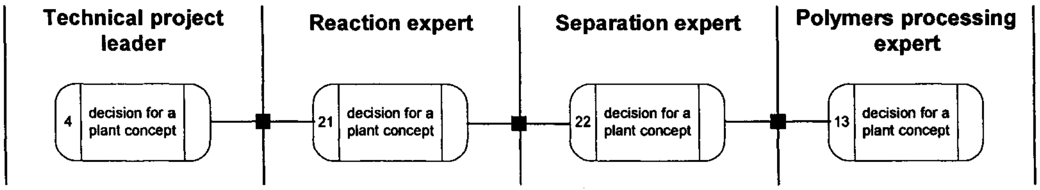

Technical project leader
Reaction expert
Separation expert
Polymers processing expert
4
decision for a plant concept
21
decision for a plant concept
22
decision for a plant concept
13
decision for a plant concept

Figure 20. Decision for plant concept.

The final decisions for the reactor and the separation system are taken by all designers together in a meeting where the influences between the different process parts can be discussed. All results of the different experts (reaction, separation, and polymers processing) need to be considered. The plant concept comprises a process description of that process which will be investigated in more detail in the following phases of the design lifecycle. PFD and pre-P&ID together with mass and energy balances and data sheets for major units are the main results and output of this case study (Figure 20).

7.1.4 DISCUSSION AND CONCLUSIONS

Two case studies for design processes in chemical engineering have been presented. In the Methyl Acetate case study a process is designed for a basic chemical intermediate by looking at major process steps in reaction and separation in detail and by including elements of the basic engineering phase such as cost estimation, safety and environmental studies. The scope of the Polyamide6 is the process chain from the monomer, *i.e.* a chemical intermediate, to the final customer product. The process design takes place in different departments; this leads to a high number of interactions and dependencies between co-workers highlighting the workflow.

Therefore, the two case studies complement each other in the way that the Methyl Acetate study has a higher emphasis on the use of many smaller specialized tools in specific parts of the conceptual design and detailed engineering phase, whereas the Polyamide6 study describes in more details the overall workflow in the conceptual design phase and focuses on the use of larger more generalized tools.

Process design involves the application of different design approaches that require different methods and tools. Process design can simply be a scale-up of a laboratory experiment, supported by the use of heuristics, design equation, shortcuts and/or graphical methods. On the other hand, in model-based design mathematical models are used at different levels of detail for prediction of process and plant behaviour during various stages of the process lifecycle 18. Within most real design processes these two approaches are combined. Both case studies rely heavily on the use of models.

From case studies like this, it is possible to derive several conclusions, for example, needs for tool support; needs for interfaces can be identified from the information flows between tools; dependencies between different activities may imply possible changes and improvements in the workflow such as better communication between the designers; and, bottlenecks and weaknesses in existing design processes can be also identified. This discussion focuses exemplarily on the need and identification of tool interfaces.

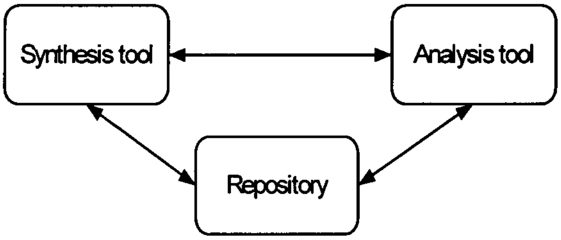


Figure 21. Classes of tools and their interactions.

During a design process different alternatives are synthesized and analysed. Based on the analysis one design alternative is chosen. It is possible to identify a structure among the methods and tools used in the design phase and can be broadly classified as shown in Figure 21.

Tools for analysis tools are used for evaluation of the chemical process of interest or parts of it under different scenarios, such as physical-chemical behaviour, safety, environmental, or economic consideration. Within analysis tools different kinds of models at various levels of detail and granularity are used and solved.

Example for analysis tools are all kinds of process simulators, tools used for parameter estimation, analysis of physical behaviour, analysis of the safety or economic potential of some design alternative, *etc.*. Repositories, where all kind of information and data is collected and stored, like physical property databases, model libraries, and design databases.

Examples of synthesis tools are tools for solvent design, flowsheet design, modelling, *etc.*. This classification of synthesis and analysis corresponds to the structure of the design process where synthesis and analysis activities are performed: several design alternatives are synthesized and analysed regarding different scenarios. Based on the results of this analysis, the designers choose one (or a small number of) design alternatives, which are considered for further investigation. The repositories are needed for storing the information.

The data handled in the different tool classes depend on each other: data that are created or changed within one tool is needed in another tool or has an influence on the data created there. A summary of information that needs to be exchanged between the different tool classes is given in Table 15. It was obtained from abstracting the data flows between the actual tools within the two case studies presented in this chapter. Thus, this table does not give a complete overview of all possible information flows occurring between different tool classes, but it gives a good impression of needed tool interfaces.

Table 15. Information exchange between tool classes in the two case studies.

Tool class A	Tool class B	A → B	B → A
Analysis	Analysis	Flowsheets Mathematical models Kinetic reaction model Property model information Design specification Process parameters Simulation results Equipment data Stream information Energy requirements	
Analysis	Repositories		Experimental data Component properties Thermodynamic data Impact factors
Analysis	Synthesis	Model information Impact factors	Flowsheets Equipment data Process parameters Unit models
Repositories	Repositories		
Repositories	Synthesis	Component properties Thermodynamic data	
Synthesis	Synthesis	Flowsheets	

7.1.4 ACKNOWLEDGEMENTS

This work is partially supported by the European Union within the project Global CAPE-OPEN and by the DFG, Deutsche Forschungsgemeinschaft, in the CRC 476 'IMPROVE'.

7.1.5 REFERENCES

1. Kirk-Othmaer Encyclopedia of Chemical technology, 1980, 3ed., Vol 9, John Wiley & Sons, New York.
2. Ronnback, R., Salmi, T., Vuori, A., Haario, H., Lehtonen, J., Sundqvist, A., Tirronen, E., Chem Eng Sci., 52, (1997), 3369-3381.
3. Popken, T., Geisler, R., Gotze, L., Brehm, A., Moritz, P., Gmehling, J., Chem Eng Technol, 21 (1999), 401-404.
4. Song, W., Venimadhavan, G., Manning, J. M., Malone, M. F., Doherty, M. F., I & EC Research, 37, (1998), 1917-1928.
5. Sirolla, J., *An industrial perspective on process synthesis*, AIChE Symposium Series, 91 (304), (1999), 222-234.
6. Jaksland, C., 1996, *Separation process design and synthesis based on thermodynamic insights*, PhD-thesis, CAPEC-DTU, Lyngby, Denmark.
7. Sawistowski, H., Pilavakis, P. A., Chem Eng Sci., 43 (1978), 355-360.
8. Perry, R. H., Green, D. W., Maloney, J. O., 1997, Perry's Chemical Engineers' Handbook, 7ed., McGraw-Hill, USA.
9. Bayer, B., Henriksen, J. P., R. Schneider, R., 2000, *Aspects of lifecyle modelling I & II*, GCO-project report, RWTH-Aachen & CAPEC-DTU.
10. Aspen Technology, 2001. *Aspen Products* [online]. Available from: http://www.aspentech.com/index.asp?menuchoice=appolymerplus. Accessed July 18, 2001.
11. Aspen Technology, 2001. *Aspen Products* [online]. Available from: http://www.aspentech.com/index.asp?menuchoice=ap5ipe. Accessed July 18, 2001.
12. Bogusch, R., Lohmann, B., Marquardt, W., Computer-Aided Process Modeling with MODKIT, in print: *Computers and Chemical Engineering* (2001).
13. Deutsches Patentamt, Verfahren und Vorrichtung zum kontinuierlichen Polykondensieren von Lactamen, patent number P 14 95 198.5 B78577 (1969).
14. Gerrens, H., On Selection of Polymerization Reactors, *Ger. Chem. Eng.* 4 (1981) 1-13.
15. Haberstroh, E., Schlüter, M., Moderne Technologien bei der Entwicklung von Simulationswerkzeugen. In: *Tagungshandbuch 20. Kunststofftechnisches Kolloquium des IKV*, Verlag Kunststoffinformation, Bad Homburg (2000) Part 7, 4-7.
16. Hornsby, P.R., Tung, J.F., Tarverdi, K., Characterization of polyamide 6 made by reactive extrusion. I. Synthesis and characterization of properties, *Journal of Applied Polymer Science* **53** (1994), 891-897.
17. Kohan, M.I. (Ed.), *Nylon Plastics Handbook*, Carl Hanser Verlag, München (1995).
18. Marquardt, W., von Wedel, L., Bayer, B., Perspectives on lifecycle process modeling, in: Malone, M.F., Trainham, J.A., Carnahan, B. (eds.): Foundations of Computer-Aided Process Design, AIChE Ser. 323, vol. 96 (2000) 192-214
19. Michaeli, W., Grevenstein, A., Engineering Analysis and Design of Twin-Screw Extruders for Reactive Extrusion, *Advances in Polymer Technology* **14** (1995) 263-276

20. Process Systems Enterprise (2001). *PSE: gPROMS Overview* [online]. Available from: http://www.psenterprise.com/products_gproms.html. Accessed July 18, 2001.
21. Tai, G., Tagawa, T., Simulation of Hydrolytic Polymerisation of ε-Caprolactam in Various Reactors. A Review on Recent Advances in Reaction Engineering of Polymerization, *Industrial & Engineering Chemistry Product Research and Development* **22** (1983), 192-206.
22. Verein Deutscher Ingenieure (Ed.), *Der Doppelschneckenextruder, Grundlagen und Anwendungsgebiete*, VDI-Verlag (1998).

Software Architectures and Tools for Computer Aided Process Engineering
B. Braunschweig and R. Gani (Editors)
© 2002 Elsevier Science B.V. All rights reserved.

Chapter 7.2: A Prototype for Open and Distributed Simulation with COM and CORBA

V. Siepmann, K.W. Mathisen & T.I. Eikaas

7.2.1 INTRODUCTION

The main part of this work was done as part of the Global CAPE Open project [1, 2, 3] where open standards for process modelling components (PMC) have been developed and integrated in process modelling environments (PME). Main PMC's include unit operation, thermodynamics and solvers and main CAPE Open compliant PME's include AspenPlus [1], Hysys [9] and gPROMS [12], see chapter 4.4 for more details. A CAPE Open PME can also be abbreviated COSE (CAPE Open Simulation Environment).

In order to really develop open simulation environments it is important to be able to integrate Windows COM [5] modules and software components on other platforms, *e.g.* Unix based on CORBA [4]. Windows is the dominating operating systems for CAPE Open PME. To establish communication between PME in COM and PMC in CORBA, some sort of a bridging module has to be developed. Furthermore a standard or procedure for developing CAPE Open CORBA PMC must be defined.

This prototype describes two alternative COM CORBA bridges, one developed in C++ [8, 15], and one in Python [7,13, 14]. Secondly, two alternative CORBA unit modules are described, both based on a CORBA C++ Unit skeleton. One is a in-house unit model typically used for proprietary unit operations and the other is a wrapping around another third party CAPE application such as a Fluent CFD simulation. For these reasons a prototype exploiting the Cape-Open interfaces for PMC in CORBA was developed. This technology is available on most widely used platforms, *e.g.* Windows, Linux, Unix, Solaris, Free-BSD, open-VMS, and it is especially suitable for heterogeneous setups.

7.2.2 SOFTWARE ARCHITECTURE

7.2.2.1 Overview

An overview of software components for the prototype is shown in Figure 1.

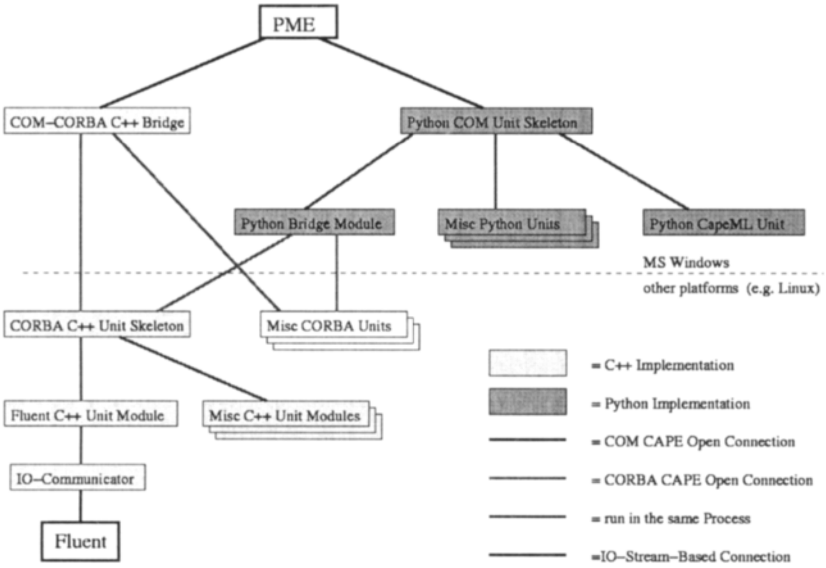


Figure 1 Software architecture overview

The modules running on Windows are shown in the upper part of the figure (above the dashed line). These are either developed in C++ (using Visual C++) or Python (using ActivePython,[2]). The modules that are not required to be running on Windows platforms are shown in the lower part of the figure. The modules to the left are used when wrapping a third party module or application, in this case a Fluent CFD simulation. One could also use the CORBA C++ unit skeleton to implement standalone unit modules. Finally is possible to develop miscellaneous CORBA units in other programming languages.

However, it was experienced that it can be quite difficult to develop COM-CORBA bridges in C++, and an alternative approach is to use Python. Python is a high level programming supporting both COM and CORBA technologies satisfactory in one single distribution. The Python approach includes (see Figure 1) a COM CAPE Open Unit Operation skeleton in Python and a bridge instantiation created by wrapping a CORBA CAPE Open Unit Operation server. The first step of this approach makes it possible to include a CapeML unit operation. CapeML is XML-based [16] and tailor-made for process modeling tasks.

Note that there are different types of module connections in Figure 1. Some connections (*i.e.*, blue and cyan ones) are based on the CAPE Open standards while red connections mean that the connected modules run in the same process.

7.2.2 Software Development Environment

The prototype is developed on Redhat Linux 7.0 and SuSE Linux 7.1 using gcc 2.95, and Windows NT 4.0 Servicepack 6. Coding has been done with the aid of Visual C++ 6.0 [15] and ActivePython [13]. Workspace definitions for Visual C++ 6.0 and GNU-style makefiles exist for the CORBA unit skeleton. The unit skeleton is represented by a pair of static / dynamic libraries for use on UNIX-like platforms. For compatibility reasons, only a static library is provided for Windows platforms. OmniORB v3.0.2 [11] is used as the object request broker, since it supplies high performance CORBA 2.3 [4] compliant code.

7.2.2.3 Process Modelling Environment

The Process Modelling Environment (PME) must be a CAPE Open Simulation Environment (COSE). Possible COSE's include AspenPlus, Hysys and gPROMS. For further details on COSE's, please see chapters 3.3 and 4.4.

7.2.2.4 COM-CORBA C++ Bridge

In order to connect CORBA CAPE Open unit operations to currently available simulation environments, a bridging tool is necessary. This tool converts the COM-calls from the executive to CORBA-calls to the external unit operation and, the other way, converts CORBA-calls from the external unit operation to compatible COM-calls to the appropriate modules located inside the simulation executive. Note that the described module is not a general COM-CORBA bridge, but specific for CAPE Open unit operations. It could be possible to split the bridge in two related modules, the first one being a COM Unit skeleton and the second the actual bridge implementation. Such a split is proposed for the Python realisation and there are historical reasons only for not splitting in the same way for the C++ implementation.

7.2.2.5 CORBA C++ Unit Skeleton

The CORBA unit operation skeleton in C++ enables programmers with little knowledge on CORBA and CAPE Open to develop a new CAPE Open compliant unit model or wrap an existing unit model so that it becomes CAPE Open compliant. This wrapping technique can be extended so that existing FORTRAN unit models can be made CAPE Open compliant.

In summary, the main requirements for the CORBA C++ unit operation skeleton are:

- It must provide an easy to use C++ unit operation framework hiding CORBA from the implementing programmer.
- It must cover most possible facilities available through the CAPE Open specifications.
- There must be a simple way to define parameters and ports as well as to access those.
- The software architecture should be flexible to allow non-standard implementations.

Figure 2 shows how the unit skeleton works in a CAPE Open context.

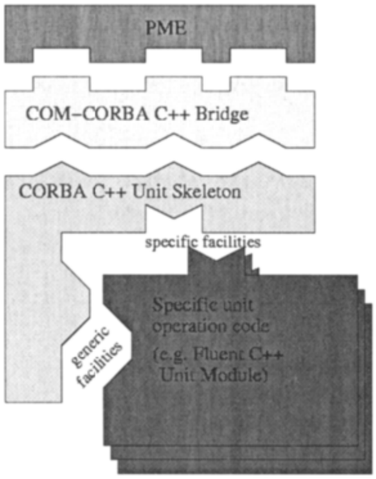


Figure 2 Illustration of history of process design

The skeleton provides an implementation of the CORBA CAPE Open interfaces, which is used by the COM-CORBA bridge to supply a PME with implementations of the COM CAPE Open interface. At the same time, it supports *generic facilities* to the specific unit operation, including a generic unit report and helper functions to create and access parameters and ports. These facilities allow the implementing programmer of a specific unit operation to concentrate on the code for initialising, calculating and terminating a unit operation. Basically these features are all to be provided from the specific implementation to the skeleton.

To avoid unnecessary effort and loss of performance and information through the skeleton, most interfaces are directly implemented one to one as classes. Inheritance relations of interfaces are therefore reflected by corresponding inheritances of the implementing classes. Figure 3 shows this at the example of the *ICapeUnitReport* interface.

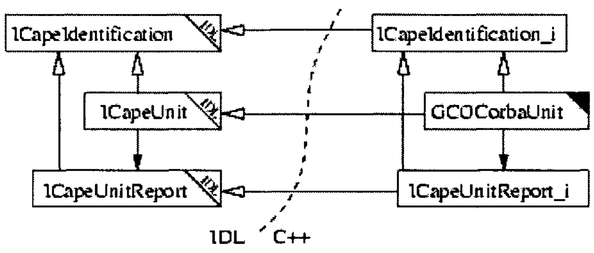


Figure 3 Illustration of the general design to implement Cape Open IDL interfaces in C++

In some cases, additional classes are introduced. Furthermore, the abstract class *GCOCorbaUnit*, which directly implements *ICapeUnit*, is equipped with C++ internal methods described as *generic facilities* above. The abstract methods *Initialize()* and *Compute()* have to be implemented by a sub-class, while *Terminate()*, *Restore()* and *Save()* are defined virtual and provide by default an empty implementation. Hence, they can be over-implemented on demand. For future versions, default persistence facilities might be needed, e.g. saving the unit parameters in an XML file.

7.2.2.6 Fluent C++ Unit Module

This module is a specific implementation of the CORBA C++ Unit Skeleton. In order to operate as intended methods for initialising, calculating and terminating the unit operation must be included. During initialisation ports and parameters are declared and some additional resources are allocated. The communication to Fluent via the IO-Communicator is initialised and validated.

During calculation messages from PME via the bridge is sent to Fluent and back. First the input port data is read, interpreted and converted to Fluent surface definition commands. Typically the input port data represents material stream vectors, *e.g.* in T, p, x and N_{total} coordinates. Second the surface definitions are sent to Fluent via the IO-communicator. Third the Fluent calculation is invoked. Finally, when the Fluent results are available the surface data for the output ports are converted to stream vectors and saved. The run is terminated by closing the connection to Fluent. Resources allocated during initialisation are freed.

7.2.2.7 IO-Communicator

This module is a C++ class that is compiled into the Fluent C++ unit module. It is capable of starting a process and handling the standard input, output and error channels.

7.2.2.8 Fluent

The Fluent module can be any Fluent simulation case, from one small piece of process equipment to a complicated set of process units or even plant sections. The Fluent case (instance) is built up using the Fluent built-in graphical capabilities, but during run-time the Fluent is called, started and run from the PME. With other words Fluent is run a batch mode. Only the minimum set of commands to manage a Fluent simulation case can be used at this stage. On a longer term the following capabilities is desired:

- Defining and reading boundary conditions;
- Obtaining different kinds of mean values for quantities on the surfaces (boundaries);
- (Re-) Initialisation of quantities in the simulation space.
- Perform steady-state iterations on the problem and check convergence;
- Integrating a defined time interval in a dynamic context

The last item emphasises that it is desired to extend the scope of the current CAPE-OPEN interfaces to enable dynamic simulations in a dynamic-modular context.

7.2.2.9 Miscellaneous C++ Unit Modules

The CORBA unit operation skeleton in C++ can also be used to develop a general CAPE Open Unit module. Unit is one of the main process modeling components (PMC) in the CAPE Open standard. However, most active participants in the CAPE Open work have focused on developing COM, not CORBA based unit modules. Experiences on developing CAPE Open units in CORBA have been gained by developing a model of a heating tank. This module can run on a separate machine from the PME and the rest of the simulation modules, and illustrates how (geographically) distributed simulation capabilities are built-in when using CORBA.

7.2.2.10 Miscellaneous CORBA Units

The main purpose of this hybrid (COM and CORBA) prototype is to demonstrate open simulation as facilitated by the CAPE Open standards. Our main approach is to use C++ in the prototype development. However, the CAPE Open standard is independent of the programming language. This means that standalone units based on CORBA that follows the CAPE Open standards can be implemented in a number of different programming languages, *e.g.* JAVA, can be used.

7.2.2.11 Python COM Unit Skeleton

The Python COM unit skeleton is developed by defining a Python class with the required CAPE Open interface methods and parameters. The methods and parameters have been described earlier in the C++ implementation. They have also been presented in chapter 4.1 where a COM implementation of the CAPE Open Unit interface standard is described.

7.2.2.12 Python Bridge Module

The Python Bridge module converts the COM calls to the appropriate CORBA syntax.

7.2.2.13 Miscellaneous Python Units

The proposed architecture may also be used to include COM CAPE Open units developed in Python. The Python COM unit skeleton will hide many technical details and make the development possible without expert's skill on COM programming.

7.2.2.14 Python CapeML Unit

A very interesting capability of the prototype architecture is that it is possible to include units specified in CapeML. CapeML is an extension of XML and designed for process modeling tasks

7.2.3 REFERENCES

1. Aspen Technology, Inc (AspenTech) (2001). *AspenPlus (Sequential modular steady state process simulator) v10.2,* http://www.aspentech.com
2. Braunschweig, B. L., Pantelides, C. C., Britt, H. I. & Sama, S. (2000). *Process modeling: The promise of open software architectures*, Chemical Engineering Progress, **96**(9) 65-76.
3. Belaud, J. P., Bernier, J., Halloran, M., Köller, J., Piñol, D. & Roux. R. (2000). *Error handling strategy: Error common interface.* Technical report, Global CAPE-OPEN and project partners.
4. CORBA (2001). *Common Object Request Broker Architecture,* http://www.omg.org
5. COM (2001), *Component Object Model,* http://www.microsoft.com
6. Greenfield, L. (1996). *The LINUX Users' Guide,* http://www.linux.org
7. Hammond, M. & Robinson, A. (2000). *Python Programming on Win32.* O'Reilly, 1st edition
8. Henning M., & Vinoski. S. (1999). *Advanced CORBA Programming with C++.* Addison-Wesley

9. Hyprotech (subsidiary of AEA Technology plc) (2001). *Hysys.Process (Sequential modular steady state process simulator) v2.2,* http://www.hyprotech.com
10. Object Management Group (OMG) (1999). *C++ Language Mapping Specification,* http://www.OMG.org
11. OmniORB (2001). OmniORB, freeware, Object Request Broker for C++ and Python, http://www.uk.research.att.com/omniORB
12. PSE (Process Systems Enterprise Ltd) (2001). *gPROMS v2.0 (Equation–oriented process simulator),* http://www.psenterprise.com
13. ActivePython (2001). ActivePython freeware development tool, http://www.python.org
14. Rossum, G. v. (1999). Python Library Reference, release 1.5.2, CNRI, http://www.python.org
15. Stroustrup, B. (1997). *The C++ Programming Language.* Addison-Wesley, 3rd edition
16. XML (2001). Extensible Markup Language, w3 consortium, http://www.w3.org

Glossary of Terms[1]

ACL	Agent Communication Languages (see chapter 6.1)
ACSL	Advanced Continuous Simulation Language
AD	Automatic differentiation
AIC	Akaike Information Criterion (see chapter 3.4)
ANS	Agent Name Server
API	Abbreviation of application program interface, a set of routines, protocols, and tools for building software applications. A good API makes it easier to develop a program by providing all the building blocks. A programmer puts the blocks together. (w)
Architecture	Defines how a computer program is broken down into its constituent parts. In the domain of OO applications, these parts are software components or objects. (c)
ARM	Application Resource Model
ASP	Application Service Provider
AspenPlus	Aspen-Commercial process simulation package (http://www.aspentech.com)
Batch process	By definition, a batch process occurs in a finite time span, having an initial start condition, a path of transformation (usually involving material and energy transfers) and a final state. This is a definition based on type of processing and does not necessarily reveal the classification of the product itself.
BDF	Backward differentiation formula (see chapter 3.2)
BFD	Block Flow Diagram
BLAS	Basic linear algebra subroutine library – A collection of FORTRAN subroutines and functions for performing basic linear algebra tasks
BPI	Business Process Improvement (see chapter 6.5)
C#	A hybrid of C and C++, it is Microsoft's newest programming language developed to compete with Sun's Java language. C# is an object-oriented programming language used with XML -based Web services on the .NET platform and designed for improving productivity in the development of Web applications.(w)
C++	A high-level programming language developed by Bjarne Stroustrup at Bell Labs. C++ adds object-oriented features to its predecessor, C. C++ is one of the most popular programming language for graphical applications, such as those that run in Windows and Macintosh environments. (w)
CAMD	Computer Aided Molecular Design
CAPE	Computer Aided Process Engineering
CapeML.Model	See chapter 3.1
CAS Number	Chemical Abstracts Service has a Chemical Abstracts Registry Number (CAS-number) for known chemicals.
CGI	CGI: Abbreviation of Common Gateway Interface, a specification for transferring information between a World Wide Web server and a CGI program. A CGI program is any program designed to accept and return data that conforms to the CGI specification. The program could be written in any programming language, including C, Perl, Java, or Visual Basic. CGI programs are the most common way

[1] Definitions in this annex are borrowed from websites with minor rewriting or simplifications. The sources are indicated: (w) for www.pcwebopedia.com , an online computer dictionary for internet terms; (o) for www.omg.org, OMG's web site. (m) for Microsoft's web site, www.microsoft.com, (c) for www.colan.org, the CAPE-OPEN Laboratories network website.

	for Web servers to interact dynamically with users. Many HTML pages that contain forms, for example, use a CGI program to process the form's data once it's submitted. (w)
CO-LaN	CAPE-OPEN Laboratories Network (CO-LaN), a not-for-profit user-driven organisation for the testing and management of the CAPE-OPEN standard.
COM/DCOM/COM+	Abbreviation of Component Object Model, a model for binary code developed by Microsoft for its implementation of middleware. COM enables programmers to develop objects that can be accessed by any COM-compliant application. COM+ builds on COM's integrated services and features, making it easier for developers to create and use software components in any language, using any tool. (m) & (c)
CO	CAPE OPEN project (see chapter 4.4)
COPS	Context Oriented Process Support System
CORBA	Short for Common Object Request Broker Architecture, an architecture that enables pieces of programs, called objects, to communicate with one another regardless of what programming language they were written in or what operating system they're running on. CORBA was developed by an industry consortium known as the Object Management Group (OMG). (w)
COSE	CAPE OPEN Simulation Executive
DAE	Differential – Algebraic Equations
Data Reconciliation	Data reconciliation is a minimization of measurement errors subject to satisfying the constraints of the process model
DCS	Distributed Control System
DIPPR	Design Institute for Physical Properties
DO	Dynamic Optimisation. Nonlinear algebraic objective function with possible integral terms. Nonlinear differential and algebraic constraints. All variables are continuous valued.
Domain oriented language	Domain-oriented (process) *modelling* languages provide modelling concepts, which correspond to domain-specific concepts.
DSL	Digital Subscriber Line
DTD	Document Type Definition
Dynamic simulation	Dynamic simulation defines the behaviour of a process over time. (c)
EAI	Acronym for enterprise application integration. EAI is the unrestricted sharing of data and business processes throughout the networked applications or data sources in an organization. (w)
EJB	EJB: short for Enterprise JavaBeans,(EJB) is a Java API developed by Sun Microsystems that defines a component architecture for multi-tier client/server systems. EJB systems allow developers to focus on the actual business architecture of the model, rather than worry about endless amounts of programming and coding needed to connect all the working parts. This task is left to EJB server vendors. Developers just design (or purchase) the needed EJB components and arrange them on the server.(w)
EPC	Engineering Procurement and Construction
ERP	Enterprise Resource Planning
ESO	Equation Set Object
EVM	Errors in Variables Measured (see chapter 3.4)
Flowsheet	A symbolic representation of process: a set of simulation blocks with connections. (c)
GAMS	General Algebraic Modelling System
GCO	Global CAPE OPEN project (see chapter 4.4)
Generic	Generic *modelling* languages are those that do not provide domain-specific

Language	concepts but concentrate on a mathematical (or systems) level.
gProms	Commercial process simulation package (http://www.psenterprise.com)
GRID computing	The growing popularity of the Internet along with the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we do computing. These new technologies enable the clustering of a wide variety of geographically distributed resources, such as supercomputers, storage systems, data sources, and special devices, that can then be used as a unified resource and thus form what are popularly known as "Computational Grids". A Grid is analogous to the electrical power grid and aims to couple distributed resources and offer consistent and inexpensive access to resources irrespective of their physical location.
GUI	Graphical User Interface. A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. (w)
HLS	Harwell subroutine library – a large collection of FORTRAN subroutines for performing tasks from linear sparse algebra to nonlinear optimisation
HTML	HyperText Markup Language, the authoring language used to create documents on the World Wide Web. HTML is similar to SGML, although it is not a strict subset. HTML defines the structure and layout of a Web document by using a variety of tags and attributes. (w)
HYSYS	Commercial process simulation package (http://www.hyprotech.com)
ICAS	Integrated Computer Aided System (http://www.capec.kt.dtu.dk)
IDL	Interface Definition Language. IDL is used to formally define the interface of a Software Component. It is programming language neutral, but its syntax is very similar to C. (c)
IIOP	Internet Inter-ORB Protocol, a protocol developed by the Object Management Group (OMG) to implement CORBA solutions over the World Wide Web. IIOP enables browsers and servers to exchange integers, arrays, and more complex objects. (w)
Interface	An interface represents a communication gateway to an implementation of the functionality of a class or Software Component. It does not imply an implementation, but defines what the implementation provides. A Software Component may employ several interfaces, which act as different aspects or ways of working with the component. Multiple interfaces are used to group together levels or kinds of functionality.(c)
IRK	Implicit Runge-Kutta method (see chapter 3.2)
IT	Information Technology
J2EE	Java 2 Platform Enterprise Edition. J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitiered, Web-based applications.(w)
Java	A high-level programming language developed by Sun Microsystems. Java was designed to take advantage of the World Wide Web. Java is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. (w)
Javascript	A scripting language developed by Netscape to enable Web authors to design interactive sites. Although it shares many of the features and structures of the full Java language, it was developed independently. Javascript can interact with HTML source code, enabling Web authors to spice up their sites with dynamic content.

	(w)
KQML	Knowledge Query Manipulation Language
Lifecycle process modelling	Where not only the different facets of process models (in the sense of data) are considered, but also their relationships and their evolution in the sense of a work process.
Linux	A freely-distributable open source implementation of UNIX that runs on a number of hardware platforms, including Intel and Motorola microprocessors. It was developed mainly by Linus Torvalds. Because it's free, and because it runs on many platforms, including PCs, Macintoshes and Amigas, Linux has become extremely popular over the last couple years. (w)
LP	Linear Program. Linear algebraic objective function and constraints. All variables are continuous valued.
MathML	MathML (see chapter 3.1)
Middleware	Middleware provides a mechanism for communication between separately compiled and linked Software Components that can be written in different programming languages and run on different processors and platforms. It thus provides a means of communicating between Software Components prepared by different authors and organizations. Middleware provides binary level standards to permit this communication. It allows software systems to be assembled from well-tested, independently developed, Software Components obtained from a variety of sources. (c)
MIDO	Mixed integer dynamic optimisation. Nonlinbear algebraic objective function with possible integral terms. Nonlinear differential and algebraic constraints. Both discrete and continuous valued variables
MILP	Mixed integer linear program. Linear algebraic objective function and constraints. Both discrete and continuous valued variables.
MINLP	Mixed integer nonlinear program. Nonlinear algebraic objective function and constraints. Both discrete and continuous valued variables
ModDev	A computer aided tool for model generation (see chapter 3.1)
MODEL.LA	A computer aided modelling tool (see chapter 3.1)
Modelica	Modelica: Modelica is the result of a standardisation effort in domain-independent (also called multi-domain) modelling languages, which aims at providing a standard language that is based on object-oriented concepts to simplify model development and reuse.
ModKit	See MODEL.LA (see chapter 3.1)
MOM	Message oriented middleware
MoT	Computer aided modelling tool-box (see chapter 3.1)
MPC	Model predictive control
NET	A Microsoft operating system platform that incorporates applications, a suite of tools and services and a change in the infrastructure of the company's Web strategy. The objective of .NET is to bring users into the next generation of the Internet by conquering the deficiencies of the first generation and giving users a more enriched experience in using the Web for both personal and business applications. (w)
NLP	Nonlinear Program. Nonlinear algebraic objective function and constraints. All variables are continuous valued
Object	An object is a software entity, which combines both properties (data or variables) and methods (procedures). Objects are created from an abstraction called a class. One can have many objects of the same class or type, each representing different instances (for example, several different distillation columns within a simulation). The term encapsulation is commonly used with objects. It means that the an

	object's internal properties and methods are not visible except in the controlled way that is defined by the class interface. (c)
Object Orientation	Object Orientation (OO) is a way of building software based on objects and the relationships between them. Objects have defined behaviours and states. Objects are defined by classes that can be grouped hierarchically. OO promotes efficient code re-use and creating programs that are easier to read. Key attributes of an object-oriented language are encapsulation, inheritance and polymorphism. These benefits are provided through a class structure, which is a feature central to all OO-languages. It is claimed that OO handles complexity better because it provides a more natural correspondence between the software and real-world entities. (c)
Object Oriented Languages	Object-Oriented languages provide classes, and mechanisms of inheritance and polymorphism. The OO languages include C++, Java, Smalltalk, C# and others. Visual Basic is often included because it supports most OO concepts and is widely used for business and engineering applications. (c)
ODE	Ordinary Differential Equations
OMG	Object Management Group, a consortium with a membership of more than 700 companies. The organization's goal is to provide a common framework for developing applications using object-oriented programming techniques. OMG is responsible for the CORBA specification. (w)
ORB	Object Request Broker
OTS	Operator Training System
Parameter Estimation	Parameter estimation is the step after data reconciliation in which the reconciled values of the process variables are used to set values for the model parameters.
PDAE	Partial Differential Algebraic Equations
PFD	Process Flow Diagram
P&ID	Piping & Instrument Diagram
PlantData XML	PlantData XML is an effort originally conceived in 1999, with work now starting in 2001, to combine existing process industry international data exchange standards and XML Internet standards to create a practical and easy to use electronic data exchange standard for the process industry.
PMC	Process Modelling Component
PME	Process Modelling Environment
POSC	Pterotechnical Open Software Corporation
PPDB	Physical Properties DataBase
PRO-II	Commercial process simulation package (http://www.simsci.com)
Programming language	*Modelling* language where a model is directly implemented into the solver, e.g. as a subroutine.
PSE	Process Systems Engineering
PUP	Public Unit Parameters
RTO	Real Time Optimisation
Sequential/ Modular Approach	In this approach each unit operation is represented by a set of equations grouped into a block (or module) and the whole flowsheet is solved one a module-by-module basis (in sequential way)
Simulation	See chapter 3.3 for definition
Simultaneous Solution or Equation oriented	The main idea of this approach is to collect all the equations and solve them as a large system of non-linear algebraic equations.
Simultaneous/ Modular or two-tier approach	The basic idea of this approach is based on solving a simplified, linearised set of equations for each unit operation in an inner-loop thereby allowing the solution of interconnected unit operations simultaneously. In the outer-loop, a rigorous model

	is used to update the linearised model.
Smalltalk	An object-oriented operating system and programming language developed at Xerox Corporation's Palo Alto Research Center. Smalltalk was the first object-oriented programming language. Although it never achieved the commercial success of other languages such as C++ and Java, Smalltalk is considered by many to be the only true object-oriented programming environment, and the one against which all others must be compared. (w)
SOAP	Simple Object Access Protocol provides a way for applications to communicate with each other over the Internet, independent of platform. Unlike OMG's IIOP, SOAP piggybacks a DOM onto HTTP in order to penetrate server firewalls. SOAP relies on XML to define the format of the information and then adds the necessary HTTP headers to send it. (w)
Software Component	A software component is a defined set of functionality provided by some implementation of compiled software. The software component functionality is defined through a specified interfaces. Access to the interface is by the middleware technology. A software component is capable of being used and re-used in different software applications. (c)
Speciality products	The word 'specialty' refers to the type of product being manufactured irrespective of process used to make it. The name, 'specialty' arises from the special functional properties that these products bring when added to other 'bulk' materials. Examples are chemicals that impart special fragrances or flavours to their substrates or modify the rheological behaviour of (bulk) fuel (e.g. diesel).
SpeedUP	Commercial simulation package (see chapters 3.1 and 5.3)
SQP	Sequential (or successive) quadratic programming (see chapter 3.4)
STEP	See chapter 4.1 for definition (also visit http://www.STEPlib.com)
MOF	The MOF, or MetaObject Facility, standardizes a meta-model - the concepts that are used to build application models. UML models, which are built from the MOF metamodel, interchange easily because of this commonality. Another part of the MOF specification defines a repository for metamodels and models. (o)
Three-tier	Three-tier: a special type of client/server architecture consisting of three well-defined and separate processes, each running on a different platform: 1. The user interface, which runs on the user's computer (the client). 2. The functional modules that actually process data. This middle tier runs on a server and is often called the application server.3. A database management system (DBMS) that stores the data required by the middle tier. This tier runs on a second server called the database server. (w)
Two-tier	Two-tier architecture: A simple client-server relationship is referred to as a two-tier architecture.
UDDI	Universal Description, Discovery and Integration. A Web-based distributed directory that enables business to list themselves on the Internet and discover each other, similar to a traditional phone book's yellow and white pages. (w)
UML	Unified Modelling Language (UML) is the software notation that emerged from collaborative efforts of Rational Software and two other leading companies in analysis of object based software design. UML merges the methods formerly known as Objectory (Ivar Jacobson), OMT (James Rumbaugh), and the Booch Method (Grady Booch).It provides different views of planned and proposed software systems. It has been accepted as a standard notation by OMG. (c)
VeDa	See chapter 3.1 for details
Visual Studio	Microsoft Visual Studio is a complete enterprise-class development system including tools for Basic, C++ and J++ development and Web applications creation. (m)

Windows	A family of operating systems for personal computers. Windows dominates the personal computer world, running, by some estimates, on 90% of all personal computers. The remaining 10% are mostly Macintosh computers. Like the Macintosh operating environment, Windows provides a graphical user interface (GUI), virtual memory management, multitasking, and support for many peripheral devices. (w)
WSDL	Web Services Description Language, an XML-formatted language used to describe a Web service's capabilities as collections of communication endpoints capable of exchanging messages. WSDL is an integral part of UDDI, an XML-based worldwide business registry. WSDL is the language that UDDI uses. WSDL was developed jointly by Microsoft and IBM. (w)
WWW	World Wide Web
XMI	XMI: short for XML Metadata Interchange, is a stream format for interchange of metadata including UML models. It's useful for transferring the model from one step to the next as your design and coding progress, or for transferring from one design tool to another. Because XMI streams models into XML datasets, it also serves as a mapping from UML to XML. (o)
XML	xXtensible Markup Language, a specification developed by the W3C. XML is a pared-down version of SGML, designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. (w)

Subject Index

Author Index

CORRIGENDUM

The chapter “3.5: Frameworks for Discrete/Hybrid Production Systems” by L. Puigjaner, A. Espuña & G.V. Reklaitis belongs to Part III, unfortunately it could not be placed in sequence, and has been placed at the end of the book, pages 663-700.

We apologize for any inconvenience this may cause.

Elsevier Science B.V.

Chapter 3.5: Frameworks for Discrete/Hybrid Production Systems

L. Puigjaner, A. Espuña & G.V. Reklaitis

3.5.1 INTRODUCTION

In a recent review (Antsaklis, 2000) the term "Hybrid" has been identified as meaning in general, heterogeneous in nature or composition, and the term "hybrid systems" meaning systems with behaviour defined by entities or involving a combination of continuous and discrete parts. Thus, a "hybrid dynamical system" is considered to be a dynamical system where the behaviour of interest is determined by interacting continuous and discrete dynamics. Such systems generate variables that are combinations of continuos or discrete values and through them cause interaction with other systems and environments. Furthermore, this discrete or continuous-valued information is time dependent, where the time may also take on continuous or discrete values. One could also distinguish between information, which is time-driven in some cases, and event-driven in others. The consideration of all these elements embraces a variety of disciplines that include process and control engineering, mathematics and computer science.

The coexistence of continuous and discrete systems requires the development of models that accurately describe the behaviour of such hybrid systems. Hybrid models may be used to significant advantage in the context of manufacturing, communication networks, computer synchronisation, traffic control and chemical processes among others. Here, we will concentrate in those related to flexible chemical manufacturing systems.

In chemical processes, although hybrid systems have a central role in embedded control systems that interact with the physical plant, they also arise in complex flexible manufacturing structures by virtue of the hierarchical organization of the whole enterprise management system. In these systems, a hierarchical framework is needed to manage complexity. Generally, higher levels in the hierarchy require less detailed models of the behavior occurring at the lower levels, thus creating the need for interaction of discrete and continuous components. In general, there will be different types of models and at different level of detail, as it will be described next.

This chapter discusses the characteristics of frameworks for discrete/hybrid production systems. First existing standards and generic models are indicated. Next sources of information and information flow are described. Then, existing modeling frameworks embodying different approaches are presented. The exploitation of information for model improvement is discussed in the next section. The last section outlines the use of models for decision-making and optimization.

3.5.2 EXISTING STANDARDS

Chemical plants constitute in part large hybrid systems. A rigorous model of a single unit of a chemical plant may easily encompass several hundred of nonlinear differential equations and several thousand algebraic equations. The associated modeling effort requires open structures, interoperability and strict adherence to existing standards. A summary of those standards is given below.

3.5.2.1 Generic models

Generic models related to the integration of manufacturing entities can be classified in two types of architecture (Sararaga *et al.*, 1999):

- Enterprise reference architectures: these are used to manage the development and the implementation of an integration system for the whole enterprise.
- Architectures for designing physical systems: these are associated with a specific system. The most interesting architectures for our study are the CIM architectures and the functional control architectures:
    - The CIM architectures are hierarchical and deterministic models of scheduling and control.
    - The functional architectures for control identify the components, specify the integration rules and present standard interfaces for developing interoperable components.

An enterprise reference architecture models the whole cycle of an integration project at the enterprise level, from its conception through its definition, functional design, detail design, implementation and operation, to its obsolescence.

Most of the architectures and methodologies developed in the Enterprise Engineering area are summarized as follows.

- **ARIS** (Architecture for Information Systems) by IDS Scheer. ARIS focuses on enterprise modeling room, on operative and information technologies, with the objective to construct the information system of the enterprise.
- **CIM-OSA** (CIM Open System Architecture) by ESPRIT program. This open architecture supplies: a general definition of CIM field, an implementation guide, a description of the system components and subsystems, and a modular structure matching the international standards. The CIMOSA reference model has been validated in different projects: VOICE in the automation sector, TRAUB in the machine-tool sector and VOICE II in the aluminum smelting sector.
- **GRAI-GIM** (GRAI Integrated methodology) by GRAI Laboratory. Initially this architecture was developed for modeling the decision structure of a manufacturing enterprise with focus on strategy, tactical and operational scheduling. Subsequently, it was extended to support the design of CIM systems, becoming an integrated methodology for modeling of business processes, that establishes a structured and clear approach for analyzing and designing this type of systems.
- **IEM** (Integrated Enterprise Modeling) by IPK Fraunhofer Institute. IEM supports the construction of enterprise models for process reengineering, and so, it also allows modeling the process dynamic to evaluate the operative options.
- **PERA** (Purdue Enterprise Reference Architecture) by Purdue Applied Control Laboratory. This model is the most complete life cycle for developing CIM systems. It involves explicitly the organization and the staff in the model, searching these critical factors in the integration of the whole system (Figure1).

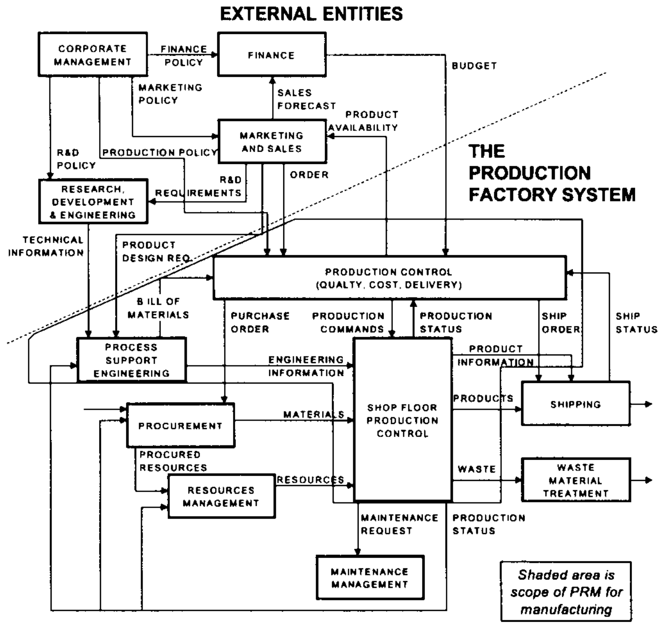


Figure 1. The Purdue Reference Model (PRM) architecture.

- **The wheel of CASA/SME** (Computer and Automated Systems Association/Society. The new wheel of the manufacturing enterprise is a graphic index of the knowledge and management aspects needed in the current manufacturing environment. It describes the main six elements to obtain a competitive manufacturing: the clients, the staff and management resources, the systems and the shared knowledge, the processes, the resources and the responsibilities, and the manufacturing infrastructure.
- **GERAM** (Generalized Enterprise Reference Architecture and Methodology) by IFAC/IFIP (International Federation of Automatic Control / International Federation for Information Processing) Task Force. GERAM defines those methods, models and tools required for developing, designing, constructing and maintaining an integrated enterprise in a changing environment It is not a new proposal. It organizes all the knowledge of enterprise integration instead of redefines it. In this way, the previous architectures maintain their identity, but they can contrast the overlapped and complementary profits with the other architectures through GERAM.

3.5.2.2 CIM Architectures

The main characteristics of CIM architectures are summarized in the following descriptions.

- **FAM** (Factory Automation Model) by ISO (International Standards Organization). It establishes a hierarchical structure with six levels (Enterprise, Factory/shop floor, Section/Area, Cell, Station and Equipment) for modeling a production shop floor. In each level, the upper level objectives are subdivided into simple tasks, and the lower level information is evaluated.
- **AMRF** (Advanced Manufacturing Research Facility) by NBS (National Bureau of Standards). This hierarchical structure with five levels (Factory, Area, Cell, Station, Equipment) has been developed with hardware and software of different manufacturers. The upper level tasks are broken down through the hierarchy, and in the lowest level those tasks are a set of simple executable actions.
- **ICAM** (International Computer Aerospace Manufacturing) program model by U.S. Air Force. This model is a functional structure with five levels, that follows structured method development for applying computer-aided technologies to manufacturing, as a manner of improving productivity.
- **COPICS** (Communication-Oriented Production Information and Control System) by IBM. COPICS involves the main activities in relation with manufacturing scheduling and control, with special attention to communications, database management and presentations.
- **Digital Equipment Corporation model**. This approach defines a control system as a set of functional modules with their corresponding data flows. All this information allows the establishment of the shop floor layout. As a result, functional and physical models for a manufacturing system and its subsystems are obtained.
- **Siemens model**. This model integrates the main technical and management functions for manufacturing through the information flow. It distinguishes a vertical information flow for interconnecting the different levels of the hierarchical model, and a horizontal information flow for controlling and synchronizing the activities of each level. Moreover, it incorporates the Computer Aided Organization (CAO) concept that involves accounting, staff and industrial accounting areas, and Computer Aided Industry (CAI) that involves CIM and CAO.
- **UPC CIM** model. This approach solves the problem of co-ordination of the different activities, considering hierarchical structures which are built under a modular philosophy (Puigjaner and Espuña, 1998). Figure 2 shows the hierarchy flowchart between the different levels. A large Real-time Data base Management system is accessible to any level, feeds the knowledge base supporting the expert systems at two levels and provides information to the optimization module.

Figure 3 shows the CIM architecture against a multi-layer optimisation framework that follows ISA SP88 and SP95 standards.

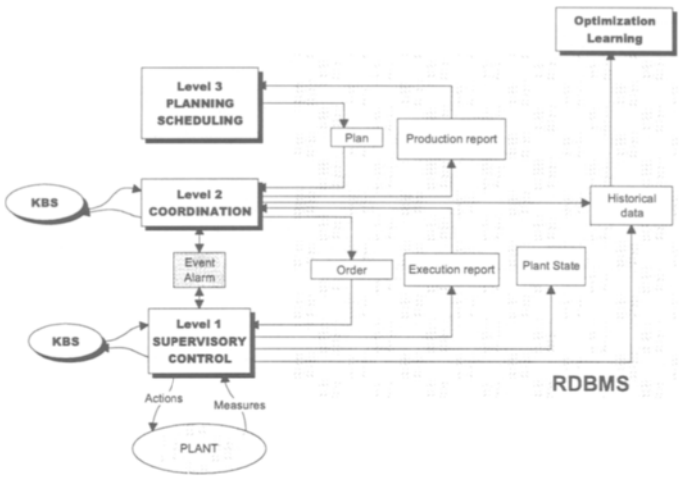


Figure 2. Flowchart of the hierarchical decision-making

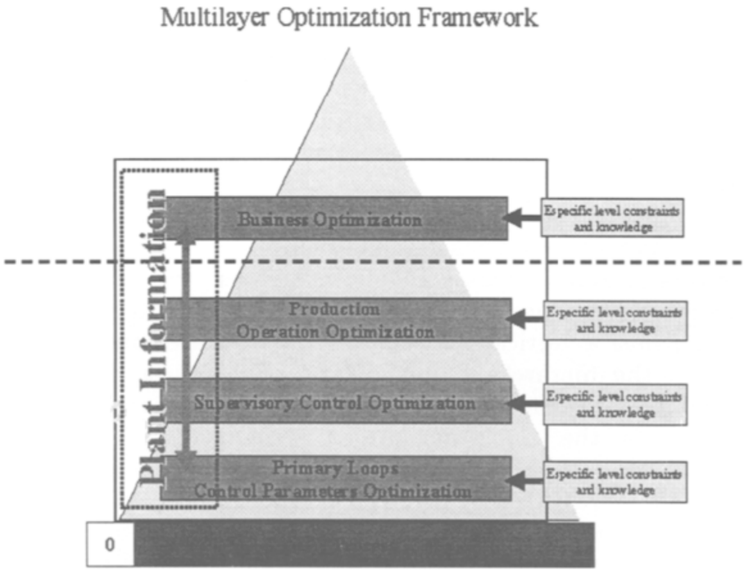

Multilayer Optimization Framework
Plant Information
Business Optimization
Especific level constraints and knowledge
Production Operation Optimization
Especific level constraints and knowledge
Supervisory Control Optimization
Especific level constraints and knowledge
Primary Loops Control Parameters Optimization
Especific level constraints and knowledge
0

Figure 3. Multi-layer Optimization Framework.

3.5.2.3 Information structure: ISA SP88

Processes have always been made up of physical objects, vessels, pumps, valves, transmitters, and controllers represent physical objects joined together by pipe and wire. Normally the physical plant objects constitute a natural hierarchical structure. The object vessel can, for example, have the sub-objects agitator and heater. The heater can have its own sub-objects such as a temperature sensor and control valve.

In the batch industries the methods to describe objects and procedures have differed from industry to industry, and sometimes even from project to project. That has served as incentive to standardize how batch control procedures must be implemented (NAMUR NE33, ASTRID, ISA S88.01, ISA S88.02, GAMP guidelines).

The ISA S88.01 introduced an object-oriented methodology to modeling the batch processes and a common set of definitions. S88.01 offers opportunities to define process control applications in a modular fashion. The application of ISA S88 in batch control will be presented later in detail.

3.5.2.4 Information management: ISA SP95

Open simulation and optimization in a real time environment requires specification of the data flows between the simulation models and applications at levels 2, 3 and 4 in the hierarchical computer system structure (ISA-dS95.01-1999) shown in Figure 4.

Data flows should lead to a reference model. The ISA model is mainly logical and does not focus on the software architecture. Interoperability requires the extensive and detailed identification of data flows so that formal software interface supporting these flows can be defined. The following modules are contemplated:

- Coordination (Supervisory Control at level 2)
- Data reconciliation (Supervisory Control at level 2)
- Model predictive control (Supervisory Control at level 2, sub-scenario of what one may call Advanced control).
- Scheduling and planing (Plant Production Scheduling and Operational Management Level 4A).

The following modules will also be described:
Fault detection and diagnosis (Supervisory Control at level 2).

It is not intended to describe some supervisory control subtasks as standalone (independent) scenarios. For example:

- Parameter estimation (although it may be part of Data Reconciliation, Fault Detection and Diagnosis, or possibly even Model predictive control)
- State estimation (although it may be part of Model predictive control, Fault Detection and Diagnosis, or possibly even part of Data Reconciliation).
- Real-time optimization (although Model Predictive Control includes optimization).
- Optimization (a term that is really too broad to be used in this context)

- Data acquisition.

The interaction can take place between simulators (or optimizers) and any other external software (not just real-time control systems) exchanging information with each other during the simulation (or optimization). So the important bit as far as GCO is concerned is what interfaces the simulator (or optimizer) needs to provide to allow this type of interaction –(not the real-time issues, which have already been considered by many other standardization efforts, *e.g.* OPC).

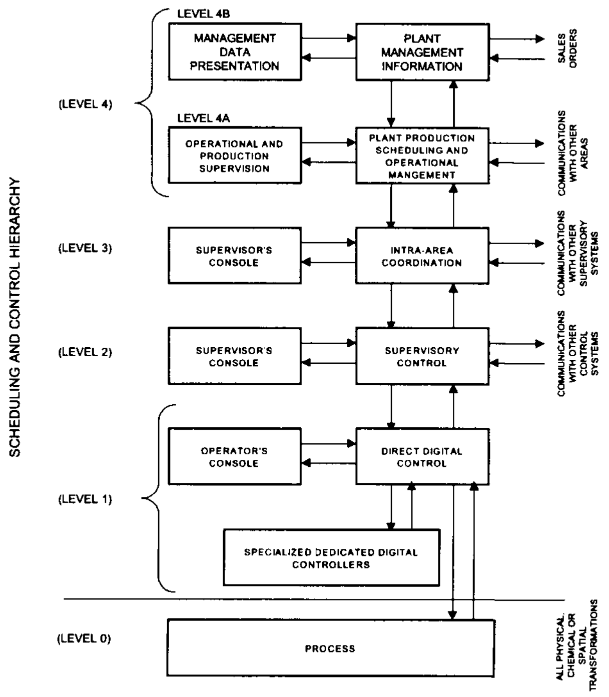

SCHEDULING AND CONTROL HIERARCHY
LEVEL 4B
(LEVEL 4)
LEVEL 4A
MANAGEMENT DATA PRESENTATION
PLANT MANAGEMENT INFORMATION
SALES ORDERS
OPERATIONAL AND PRODUCTION SUPERVISION
PLANT PRODUCTION SCHEDULING AND OPERATIONAL MANGEMENT
COMMUNICATIONS WITH OTHER AREAS
(LEVEL 3)
SUPERVISOR'S CONSOLE
INTRA-AREA COORDINATION
COMMUNICATIONS WITH OTHER SUPERVISORY SYSTEMS
(LEVEL 2)
SUPERVISOR'S CONSOLE
SUPERVISORY CONTROL
COMMUNICATIONS WITH OTHER CONTROL SYSTEMS
(LEVEL 1)
OPERATOR'S CONSOLE
DIRECT DIGITAL CONTROL
SPECIALIZED DEDICATED DIGITAL CONTROLLERS
(LEVEL 0)
PROCESS
ALL PHYSICAL, CHEMICAL OR SPATIAL TRANSFORMATIONS

Figure 4. A hierarchical computer control system structure for an industrial plant. From ISA-dS95.01-1999, Enterprise-Control System Integration. Part 1: Models and Terminology.

The "communication" with other external software is realized with OPC and utilized in real-time issues. These specifications are contemplated at a general level. Our purpose is to specify the type of "specific" information in the Process Industry that is exchanged through these standards. In this way, OPC will be tool for the interaction.

There are two main general modes of external software interacting with process simulators:

- The simulator replaces the plant and it is utilized for validation purposes and for operator training.
- The simulator runs in parallel with the plant and aids the other systems to make decisions or to evaluate actions (decision support).

The information that is exchanged in each one of the above interaction modes is as follows:

- In the operators' training case, the simulation is dynamic. The external software sends the simulator time-varying control actions and receives "plant" measurements. There are several levels at which simulators can interact with operators. Depending on the level, the time requirement change, and also the operation dynamics. That is, at some levels it is possible that the simulation could be in a steady-state mode.
- In a decision support tool context, the simulation can be either steady state or dynamic.

3.5.3 SOURCES OF INFORMATION

3.5.3.1 Process Monitoring

The decision-making process concerning possible modification of a system requires knowledge about its actual current state. This requirement is even more important in hybrid manufacturing structures. The relevant information is obtained by collecting a data set, improving its accuracy and refining it. The refined information is used for *process monitoring* which may initiate further analysis, diagnosis and control actions, optimization and general management planning.

Data collected on industrial plants is typically subject to variation in quality. This is inherent in all industrial data collection because the sampling and testing equipment, schedules and techniques are exposed to a wide range of influences. Such influences will give rise to a raw data set containing missing data points, gross errors, and outliers, all of which need to be treated to obtain a useable data set (Bagajewicz, 2001).

Gross errors are caused by an instrument or testing procedure that gives an incorrect response for an extended length of time. The results are therefore not representative of the process state. If the errors are large enough to distort the real trend, they should be excluded from data analysis. Gross errors cause problems when the variable is being used in a data information extraction procedure such as principal component analysis (PCA) or any other data compression technique. Gross errors in even one variable can therefore be of critical importance to the usefulness of an entire data set if advanced data analysis tools are to be used. The data collection systems need to be designed to detect the onset of gross errors and alert the operators. *Outliers* in data occur even in the best data collection systems. They need to be detected and eliminated so as not to distort statistical tests conducted in the data. This distortion occurs because the outliers increase the apparent measuring error (residual variance) and consequently affects the sensitivity of confidence tests. Missing data disrupts the regularly spaced nature of sampled process data and prevents the use of various data analysis techniques that assume regular sampling. Consequently, missing data should be replaced with realistic estimates of process readings before using the data in analysis.

Process monitoring and Trend Detection focuses on identifying the trends in the behavior of a process in real-time. Various methods based on *multivariate statistics* (Principal Component Analysis, Partial Least Squares, Canonical Variates Analysis), *system Theory* (Canonical Variates State Space Modeling, Observers, State Estimators, Hidden Markov Models, Wavelets), *artificial intelligence* (Knowledge-Based Systems, Neural Networks, Fuzzy Logic) have been reported for process monitoring and trend analysis. As the process moves from normal operation to abnormal operation, it is imperative that this identification is made quickly and correctly. Since most processes are monitored by relying on models built from process data as opposed to models derived from fundamental equations, using several techniques simultaneously and seeking some consensus among their results is more advantageous than relying on any single technique.

3.5.3.2 Plant and Process Simulation

Chemical processing plants have been traditionally categorized into continuous and batch plants. An uninterrupted flow of material through the plant characterizes continuous plants, which are most commonly operated in a steady-state manner. By contrast, in batch plants, discrete quantities of material undergo physicochemical transformations. The process parameters usually vary overtime, operating conditions may be different from one batch to next and available resources may be allocated differently to create a variety of production routes. Thus, batch plants are more dynamic and flexible than continuous ones. In practice, continuous and batch operations can both be present in chemical production facilities to benefit from the advantages of each and/or because of the hybrid nature of the underlying physical-chemical processes themselves. It is therefore necessary to consider this hybrid nature of the continuous-discrete interactions taking place in chemical processing plants within an appropriate framework for plant and process simulation (Engell *et al.*, 2000).

A common tool that is needed is the computer modeling and simulation of such interactions between continuous and discrete elements of chemical plants. This is addressed in the next section.

3.5.4 ORGANIZING INFORMATION: MODELING FRAMEWORKS

The discrete transitions occurring in chemical processing plants have only recently been addressed in a systematic manner. Barton and Pantelides (1994) did pioneering work in this area. A new formal mathematical description of the combined discrete/continuous simulation problem was introduced to enhance the understanding of the fundamental discrete changes required to model processing systems. The modeling task is decomposed into two distinct activities: modeling fundamental physical behavior, and modeling the external actions imposed on this physical system resulting from interaction of the process with its environment by disturbances, operation procedures, or other control actions.

The physical behavior of the system can be described in terms of a set of integral and partial differential and algebraic equations (IPDAE). These equations may be continuous or discontinuous. In the latter case, the discontinuous equations are modeled using state-transition networks (STNs) and resource-task networks (RTNs) which are based on discrete models. Otherwise, other frameworks based on a continuous representation of time have appeared more recently (Event Operation Network among others). The detailed description of the different representation frameworks is the topic of the next section.

3.5.4.1 Process Representation Frameworks

The representation of *State - Task – Network* (STN) proposed by Kondili *et al.* (1993) was originally intended to describe complex chemical processes arising in multiproduct/multipurpose batch chemical plants. The established representation is similar to the flowsheet representation of continuous plants, but is intended to describe the process itself rather than a specific plant.

The distinctive characteristic of the STN is that it has two types of nodes; mainly, the *state* nodes, representing the feeds, intermediates and final products and the *task* nodes, representing the processing operations which transform material from input states to output states. State and task nodes are denoted by circles and rectangles respectively (Figure 5).

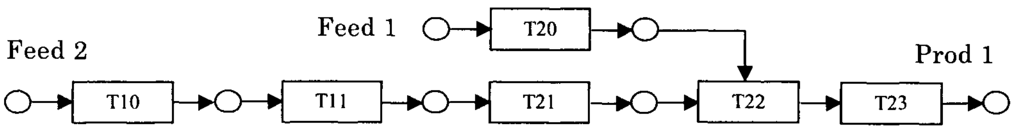


Figure 5. State-task network representation of chemical processes.

This representation is free from the ambiguities associated with recipes networks where only processing operations are represented. Process equipment and its connectivity is not explicitly shown. Other available resources are not represented.

The STN representation is equally suitable for networks of all types of processing tasks –continuous, semicontinuous or batch. The rules followed in its construction are:

- A task has as many input (output) states as different types of input (output) material.
- Two or more streams entering the same state are necessarily of the same material. If mixing of different streams is involved in the process, then this operation should form a separate task.

The STN representation assumes that an operation consumes material from input states at fixed ratio and produces material for the output state also at a known fixed proportion. The processing time of each operation is known a priori and considered to be independent of the amount of material to be processed. Otherwise, the same operation may lead to different states (products) using different processing times.

States may be associated to four main types of storage policies:

- Unlimited Intermediate Storage.
- Finite Intermediate Storage.
- No Intermediate Storage.
- Zero Wait (the product is unstable).

An alternative representation; the *Resource-Task Network* (RTN) was proposed by Pantelides (1994). In contrast to the STN approach, where a task consumes and produces materials while using equipment and utilities during its execution, in this representation, a task is assumed only to consume and produce resources. Processing items are treated as though consumed at the start of a task and produced at the end. Furthermore, processing equipment in different conditions can be treated as different resources, with different activities consuming and generating them –this enables a simple representation of changeover activities. Pantelides (1994) also proposed a discrete-time scheduling formulation based on the RTN which, due to the uniform treatment of resources, only requires the description of three types of constraint, and does not distinguish between identical equipment items. He demonstrated that the integrality gap could not be worse than the most efficient form of STN formulation, but the ability to capture additional problem features in a straightforward fashion are attractive. Subsequent research has shown that these conveniences in formulation are overshadowed by the advantages offered by the STN formulation in allowing explicit exploitation of constraint structure through algorithm engineering.

The STN and RTN representations use discrete-time models. Such models suffer from a number of inherent drawbacks:

- The discretization interval must be fine enough to capture all significant events, which may result in a very large model.
- It is difficult to model operations where the processing time is dependent on the batch size
- The modeling of continuous operations must be approximated and minimum run-lengths give rise to complicated constraints.

Therefore, attempts have been made to develop frameworks based on a continuous time representation.

A realistic and flexible description of complex recipes has been recently improved using a flexible modeling environment (Graells *et al.*, 1998) for the scheduling of batch chemical processes. The process structure (individual tasks, entire subtrains or complex structures of manufacturing activities) and related materials (raw, intermediate or final products) is characterised by means of a Processing Network, which describes the material balance. In the most general case, the activity carried out in each process constitutes a general Activity Network. Manufacturing activities are considered at three different levels of abstraction: the Process level, the Stage level and the Operation level.

This hierarchical approach permits the consideration of material states (subject to material balance and precedence constraints) and temporal states (subject to time constraints) at different levels.

At the process level, the *Process and Materials Network* (PMN) provides a general description of production structures (like synthesis and separation processes) and materials involved, including intermediates and recycled materials. An explicit material balance is specified for each of the processes in terms of a stoichiometric-like equation relating raw materials, intermediates and final products (Figure 6). Each process may represent any kind of activity necessary to transform the input materials into the derived outputs.

Between the process level and the detailed description of the activities involved at the operation level, there is the S*tage level.* At this level is described the block of operations to be executed in the same equipment. Hence, at the stage level each process is split into a set of the blocks (Figure 7). Each stage implies the following constraints:

- The sequence of operations involved requires a set of implicit constraints (links).
- Unit assignment is defined at this level. Thus, for all the operations of the same stage, the same unit assignment must be made.

- A common size factor is attributed to each stage. This size factor summarises the contribution of all the operations involved.

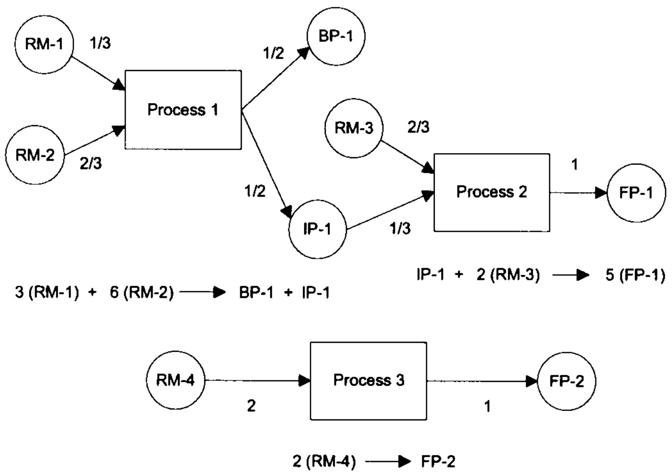


Figure 6. PMN describing the processing of two products

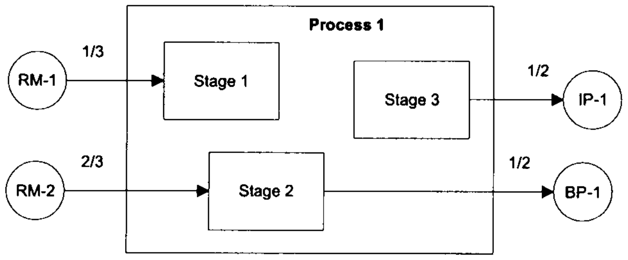


Figure 7. Stage level. Each stage involves different unit assignment opportunities.

The *Operation level* contains the detailed description of the activities contemplated in the network (tasks and subtasks). While implicit time constraints (links) must be met at this level, as indicated in Figure 8 by the thick arrows. The detailed representation of the structure of activities defining the different processes is called the *Event Operation Network* (EON). It is also at this level that the general utility requirements (renewable, non-renewable, storage) are represented.

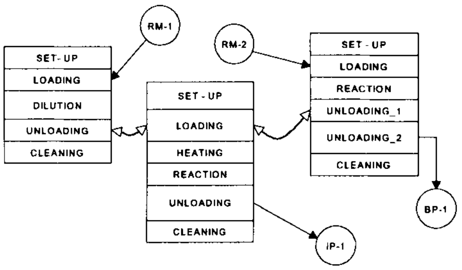


Figure 8. Operation level: operation links are shown.

The Event Operation Network representation model describes the appropriate timing of process operations. A continuous time representation of process activities is made using three basic elements: events, operations and links.

Events designate those time instants where some change occurs. They are represented by nodes in the EON graph, and may be linked to operations or other events (Figure 9). Each event *n* is associated to a time value T_n and a lower bound T_n^{min} (1).

Operations comprise those time intervals between events (Figure 10). Each operation *m* is represented by a box linked with solid arrows to its associated nodes: initial *NI m* and final *NF m* nodes. Operations establish the equality links between nodes (2) in terms of the characteristic properties of each operation: the operation time, *TOP* and the waiting time *TW*. The operation time will depend on the amount of materials to be processed; the unit model and product changeover. The waiting time is the lag time between operations, which is bounded by equation (3).

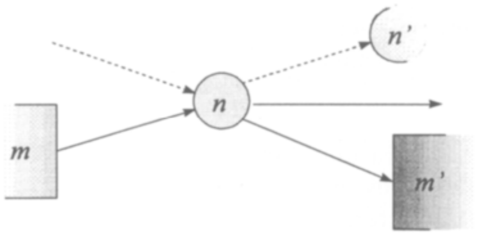


Figure 9. Nodes can be connected to other nodes or operations.

$$T_n \geq T_n^{\min} \tag{1}$$

$$T_{NFm} - T_{NIm} - TOP_m = TW_m \qquad 2)$$

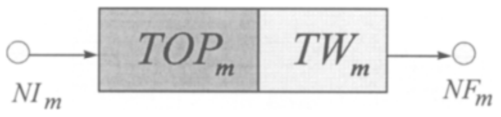


Figure 10. The time description for Operations

$$0 \leq TW_m \leq TW_m^{\max} \tag{3}$$

Finally, links are established between events by precedence constraints. A dashed arrow represents each link k from its node of origin NO_k to its destiny node ND_k and an associated offset time ΔT_K. The event –event links can be expressed by the inequality constraint (4)

$$T_{NDk} \geq T_{NOk} + \Delta T_k \tag{4}$$

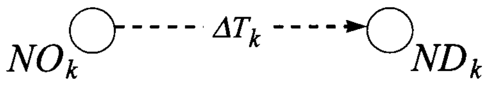


Figure 11. Event to event link and associated offset time representation

Despite its simplicity, the EON representation is very general and flexible and it allows the handling of complex recipes (Figure 12). The corresponding TOP, according to the batch size and material flowrate also represents transfer operations between production stages. The necessary time overlapping of semicontinuous operations with batch units is also contemplated in this representation through appropriate links.

Other resources required for each operation (utilities, storage, capacity, manpower, *etc.*) can be also considered associated to the respective operation and timing.

Simulation of plant operation can be performed in terms of the EON representation from the following information contained in the process recipe and production structure characteristics:

- A sequence of production runs or jobs (*Orp*) associated to a process or recipe *p*.
- A set of assignments (*Xujpr*) associated to each job and consistent with the process *p* (*Xujpr* ≤ *xujp*).
- A batch size (*Brp*) associated to each job and consistent with the process *p* ($B_{rp}^{min} \leq Brp \leq Brp^{max}$).
- A set of shifting times (T_n^{min}) for all the operations involved.

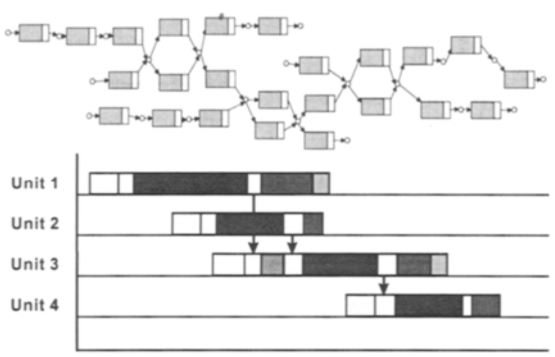


Figure 12. The recipe described as a structured set of operations. The EON representation allows the handling of complex synthesis problems.

These decisions may be generated automatically by using diverse procedures for the determination of an initial feasible solution. Hence, simulation may be executed by solving the corresponding EON to determine the timing of the operations and other resources requirements.

The effective simulation of batch processes requires representation of the dynamics of the individual batch operations, the decision logic associated with the start and stop of operations, as well as the decisions associated with the assignment of equipment and other resources to specific operations as defined through the product recipe. The BATCHES (Batch Process Technologies) simulation framework does accommodate the above mentioned batch process features and uses advances in combined discrete-continuous dynamic simulation methodology to follow the progress of the batch plant over time.

A BATCHES simulation model consists of three main building blocks: a recipe network, an equipment network and a set of processing directives (Mockus and Reklaitis, 1996). The equipment network defines the equipment specifications and the connectivity or transfer limitations between the equipment. The recipe for the manufacture of a product is modeled as a network of tasks and each task as a sequence of subtask. A task consists of all of the operations performed in a single item of equipment: a subtask consists of a model of one of these operations. Tasks have associated with them requirements of specific types of equipment and selection priorities. BATCHES provide a library of models of various types of operations (heating, cooling, decanting, batch reaction, *etc.*). Subtasks may have associated with them additional requirements for resources types and levels such as operator types and utilities as well as definition of conditions under which the subtask is to be terminated. These may be state dependent (a specific temperature or composition level is achieved) or directly specified (completion time or duration). Processing directives consist of information that drives the execution of the process over time. These include information such as the amounts and sequences in which the various products are made, the due date and amount for finished product delivery, or the amounts and frequency of raw materials deliveries or other resource releases.

BATCHES uses a dynamic solution strategy under which the dynamic models associated with all of the subtasks that are active in a given time period are solved simultaneously using a DAE solver. As the solver advances through time, the occurrence of subtask termination or start events is tested at each solver time step. As events are identified and occur, the set of active subtask models is reconfigured and the solution process continued. This computational approach is effectively a decomposition strategy as only the models of the subtasks active at a given point in time are actually included in the integration step executed. To accommodate stochastic parameters, BATCHES allows Monte Carlo sampling of simulation parameters from a library of distributions using established techniques from the discrete event simulation literature. It also provides linkages to physical properties estimation capabilities. More complex decision processes, such as solution of an assignment or scheduling model, can be accommodated by defining event conditions under which the simulation is interrupted, the information necessary to execute the decision model is assembled, the decision model solved, and the simulation of resulting actions transferred back to the simulation executive.

3.5.5 EXPLOITING INFORMATION FOR MODEL IMPROVEMENT

3.5.5.1 Parameter Estimation

Hybrid systems require an accurate modeling environment capable of continuous model updating and improvement.

One fundamental task in real time model improvement is model parameter estimation (PE). This task could either be the sole aim of the user (that is, one only requires estimates of some process parameters without a data reconciliation step), or a subtask of data reconciliation (the process model used for data reconciliation (DR) contains some unknown or uncertain parameter that must be determined). If the uncertain parameter is related to the other variables in a non-linear relationship, the two tasks (DR, PE) must be performed simultaneously, otherwise the parameters could be estimated with corrupted data. In the case where the relationship is linear the two tasks can be performed sequentially, first DR and then PE.

3.5.5.2 Data Reconciliation

Data reconciliation (DR) is a procedure that adjusts the values of the raw measurements to be consistent with the conservation's laws and other exact constraints, so that the random error is eliminated and the variable variance is reduced. However the presence of gross error on the data can invalidate the DR procedure. The reader is referred to chapter 3.4 for a discussion on "data reconciliation framework".

3.5.5.3 Gross Error Detection

The adjustment of process data leading to better estimates of the process variable true value is normally performed in two steps. Non-random measurement errors, such as persistent gross errors, must first be detected, then removed or corrected. Next, the measurement(s) and/or constraint(s) that contain the gross error must be identified Indeed, meaningful data reconciliation can be achieved if and only if there is no gross error present in the data.

Thus the functionality of gross error detection encompasses the detection of the gross error, the identification of the variable subject to error and if possible the correction of the error encountered. On the other hand GED could return information about the type of gross error that has been identified, namely, a process-related error or measurement-related error. GED must be performed prior to DR step since a key assumption during DR is that errors are normally distributed. The residual vector (**e**) in Eq. (5) affect the violation of the constraints by the measurement and is the fundamental vector used in gross error detection with its covariance matrix. In order to maintain the degree of redundancy and observability, it is preferable, if possible, to compensate the measurement rather than eliminating the variable in error.

$$f(x_m) = e \tag{5}$$

The first step in the development of data reconciliation, parameter estimation, Gross error detection or variable classification is the preparation of a process model. This model is generally based on balance equation (energy, mass, components). This use case allows the introduction of the entire model constraints that represents the whole plant and the model parameters.

3.5.6 EXPLOITING MODELS FOR DECISION-MAKING & OPTIMIZATION

Modeling hybrid processes can become very complex depending on the level of detail required to adequately represent real-life operations, and very difficult to standardize if it involves a very wide variety of operations, as is the case with large-scale processes. Also, a major limitation of present solutions is that they do not adequately reflect the distributed nature of the problem (Puigjaner, 1999) in terms of organization and production units (plants, production departments, and lines, batch units). As a consequence, internal disturbances occurring at any level of this organizational context or external perturbations caused by the market environment may create frequent and irrecoverable readjustments in real-life industrial operations. A realistic answer to this situation inevitably entails appropriate consideration of the interaction between various planning levels linked to the batch control system:

- Plant management and scheduling control, including planning, scheduling and plant wide optimization;
- Subplant co-ordination between major production areas, including local schedule adjustments and recipe modifications;
- Switching and supervisory control of process units, including appropriate handling of emergencies;
- Individual equipment regulatory and fault diagnosis actions.

All these levels should operate on real time process information base which must be supported with data reconciliation and trend tracking capabilities (Pekny *et al.*, 1991).

3.5.6.1 Scheduling and Planning

The complex problem of what to produce and where and how to produce it is best considered through an integrated, hierarchical approach which also acknowledges typical corporate structures and business processes (Rickard *et al* 1999). The scheduling problem is in essence a complex decision problem, which involves many factors. As a result the decision to adopt a particular schedule is best made within a framework that provides some assessment of the merits and properties of the proposed schedule. Thus, a practical scheduling tool should not only generate a Gantt Chart and a Resource use Diagram corresponding to "the" solution, rather it should allow an assessment of the impact of that schedule in context. A detailed simulation most readily provides that context. For this reason it is desirable to integrate a scheduling and planning tool with a simulation capability. A scheduling tool with integrated simulation capability allows ready evaluation of the flexibility of the proposed schedule, and for instance to see the impact of variations in the products, operating conditions, environmental and safety factors that might not be explicitly taken in to account in the scheduling formulation. Of course, from the simulation point of view, it is also necessary to have a tool, which allows the correct simulation of the system under consideration.

The information flows, which occur between the components of a scheduling system and various users, are shown in Figure 13. The Scheduling and Planning system receives from the management information system (ERP, MRP) in level 4 (Figure 4), detailed information on due dates, forecasts and orders, and returns to the ERP system the predicted inventory status and the work plan. The scheduling system also sends the predicted schedule to the supervisory control system, which launches the realized schedule. Finally, operators, engineers and managers have access to information about the schedule through a Graphical – User interface. On the other hand the end user-operational staff can manage the information, create new schedules and analyze the results.

The problem of short-term scheduling can be formulated as follows:

Given:

- The production recipe.
- The available units for each task, the units connectivity and their availability.
- The list of amount, time and kind of utilities consumption required by each unit available for each task (could be variable with the amount of material being processed).
- The market requirements expressed as specific amounts of products at a given instant of time (product orders) and other limitations on shared resources.

Determine:

- The optimal sequence of tasks performed in each unit (under specific performance criteria)
- The amount of material being processed at each time in each unit.
- The processing time of each task in each unit.
- The use of utilities as function of time.

Therefore, this problem involves:

- The assignment of units and resources to tasks.
- The sequencing of the tasks assigned to specific units.
- The determination of the start and end times for the execution of all tasks.
- The resources consumption at any time.

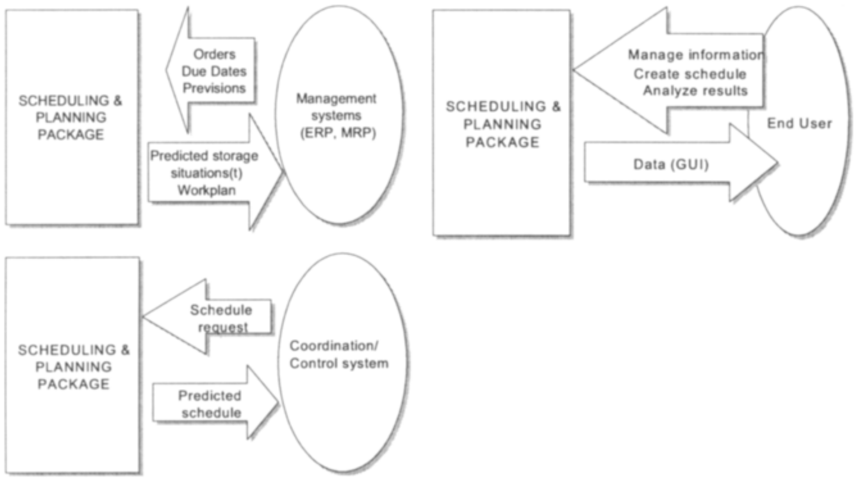

SCHEDULING & PLANNING PACKAGE
Orders
Due Dates
Previsions
Predicted storage situations(t)
Workplan
Management systems (ERP, MRP)
SCHEDULING & PLANNING PACKAGE
Manage information
Create schedule
Analyze results
Data (GUI)
End User
SCHEDULING & PLANNING PACKAGE
Schedule request
Predicted schedule
Coordination/ Control system

Figure 13. Data flows between the scheduling system and other systems

Additional issues which might arise, include:

- Consideration of exclusion times, *i.e.*, time periods during which assignment of some equipment is not allowed due to maintenance requirements.
- Discrepancies between the proposed schedule and what is actually achievable. Discrepancies may be caused by processing time variations, rush orders, failed batches, equipment breakdowns, etc. So, the schedule should be adjusted to incorporate this new information. This is called reactive scheduling.

Once this information is processed, the resulting modified solution of the scheduling problem becomes the schedule implemented. The schedule is reported as a sequence of recipe task to unit assignments and resource consumption time profiles. All this information is usually represented using a Gantt Chart and Resource Diagram

The Gantt chart shows the use of the different equipment units along time (see Figure 14). With this chart it is possible see the start and end of a task, and the unit used for making that task. The colors correspond to different product batches and the numbers are task identifiers.

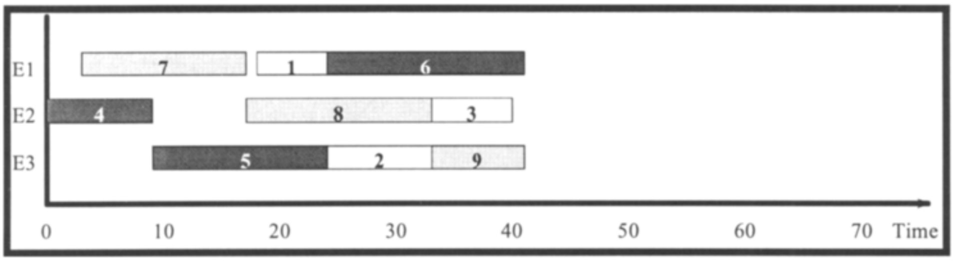


Figure 14. Gantt Chart

The Resource Diagram shows the consumption of resources like steam, electricity, manpower and others, also as a time function.

These graphical constructions are a major source of information for the production staff, but are also needed by the sales and supply departments, for inventory control purposes, and plant maintenance.

For the evaluation of any plan or schedule, there are a number of different performance measures that can be used according to the specific needs of an enterprise. Some measures have to do with the time a batch spends in the plant; others pertain to performance relative to specified due dates; and still others concern utilization of production resources (Silver *et al.*, 1998).

Note that these three general goals may be in conflict with each other. For example, high resource utilization may increase the time spent, and may degrade due date satisfaction. Therefore in spite of the very short-term nature of the scheduling problem, larger strategic concerns often are at issue.

Managers should be very concerned with the trade-off implied in the choice of which performance measure to use. For example, this choice may implicitly trade off Work in Process Inventory (WPI) cost with customer satisfaction regarding meeting due dates, and amortization of capital investment. These three general goals can be specified in more precise performance measures.

One performance measure that is a primary focus of plant managers is the average WPI level. High WIP levels mean that more money must be invested in inventory, thus adding to the operational cost and incurring opportunity costs. Flowtime is the time that a batch spends from the moment it starts processing until its completion, and includes any waiting time prior to processing. The makespan is the total time for all jobs to finish processing.

Some performance measures have to do with performance relative to each batch's due date. These include lateness, earliness and tardiness. Lateness is the amount of time a batch is past its due date. Lateness is a negative number if a batch is early. Tardiness equals lateness if the batch is late, or zero if it is on time or early. Earliness is the amount of time prior to its due date at which a batch's processing is complete.

Managers concerned with amortizing investment in equipment would like to see the equipment continuously processing batches that bring in revenue. Equipment utilization and labor utilization are the primary performance measures of plant utilization.

The selection of the performance measure is a managerial decision and has to do with the strategic objectives of the particular enterprise.

The information required for the Coordination/Supervisory Control level is basically the Schedule to be executed. This information is included in the Results Report of the previous part. Eventually, the SPP will require information from the Coordination/Supervisory Control level on the actual Schedule being realized and the production capacity of the plant (see Table 1).

Table 1. Information flow

Coordination / Supervisory Control	Scheduling & Planning Package
Schedule information retrieval	
Ask the SPP for the Schedule to be performed in the plant	Responds with the Schedule

3.5.6.2 Supervisory Control

This level considers activities related to the Model Predictive Control module, Data Reconciliation and the Fault Detection and Diagnosis modules.

Model Predictive Control receives the sensor signals and the previous control inputs to the plant. The outputs are the control signals to the plant and to the set points of the multiloop controllers. This module determines the optimal control actions. Control methods and strategies can be classified into a number of different categories representative of the evolution of the field. These include the PID family, classical methods, such as selective controllers, model based and predictive techniques, new generation advanced techniques based on optimal control theory, expert systems, neural networks, fuzzy and hybrid systems.

A general scheme of a model predictive control module is shown in Figure 15. The optimizer receives the differences between the plant's output and the set points. The future control signals are optimized according to the predicted behavior of the plant in a given time horizon. The signal is sent to the plant via the DCS. In order to predict the behavior of the plant a plant model is needed (Morari and

Lee, 1999). This model need to be periodically updated as the system changes with time.

3.5.6.3 Data Reconciliation

Process measurements are subject to errors due to random errors, as well as instrumentation faults, such as incorrect installation or calibration, process leaks, etc. The main purpose of data reconciliation is to provide consistency between the measured variables and the constraints, which arise from mass and energy balances and are assumed to be the "true" process model. Mathematically this can be expressed as reconciling the process measurements by minimizing a weighted least squares or maximum likelihood function of the measurement errors (Chouaib *et al.*, 2000).

3.5.6.4 Process Fault Detection and Diagnosis (PFD&D)

The PFD&D subsystem receives sensor data from the plant as well as the control signals. The information can be in the form of continuous signals (temperatures, flowrates, pressures) or discrete signals (valves open or close, pumps on or off). The outputs are a set of suspected faults. The signal corresponding to each suspected fault is considered to be binary (0 or 1). At the next level of detail the size of the diagnosed fault or its estimated probability could also be given. This output can then be used by the advanced control module in order to take control actions, or by the operators who take appropriate action or by other levels in the management and control structure, such as the scheduling system (Ruiz *et al.*, 2000).

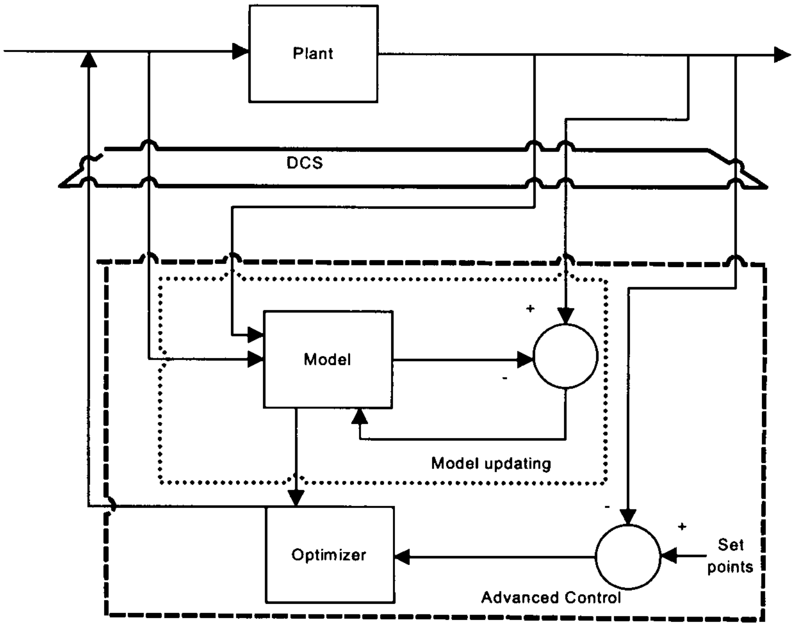


Figure 15. Scheme of the advanced control module

Table 2. Example of a part of HAZOP analysis

Variable	Guide -word	Cause	Consequence	Corrective actions	Preventive actions
Flow	Low	Pipe leakage	Loss of product Time needed for tank charge increased	Close the valve Check the pipe	Container/pipe Maintenance

Table 2 shows as an example a small part of a Hazard and Operability Analysis (HAZOP) of a process stream in a chemical plant.

Based on the previous analysis there is a suspected fault called "Pipe leakage". The output of the PFD&D system has different forms according to the decision level and associated system modules (Table 3).

Table 3. Information to be sent from the PFD&D system to the other levels

Module	Level	Translation from the PFD&D output
Control system	2	Close the valve
Scheduling system	4	Time needed for tank charge increased
Supervisor console	2	Check the pipe

The use of a DataBase system simplifies the communication of information because the same information is used by the different modules at different levels of detail and from different points of view. It also assures that the actions to be performed are coherent and avoids redundancies.

Figure 16 shows the data flows in the Fault Detection and Diagnosis system and Figure 17 shows the data flows between the Fault Detection and Diagnosis system and other systems (Leonhardt and Ayoubi, 1997; Isermann and Baillé, 1997)

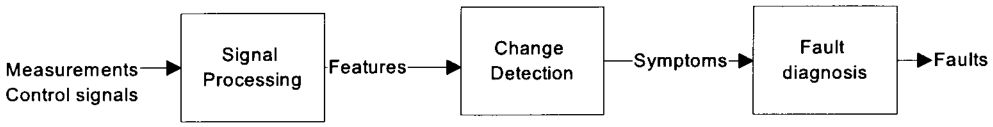


Figure 16. Fault Detection and Diagnosis System

The analytical knowledge about the process is used to produce quantifiable, analytical information. To do this, *Signal Processing* based on measured process variables have to be performed, to first generate the characteristic values, consisting of parameters, state variables or residuals. Special *features* can be extracted from these characteristic values. These features are then compared with the normal features of the non-faulty process. For this purpose, methods of *Change Detection* and classification are applied. The resulting changes (discrepancies) in the described directly measured signals, signal models or process models are considered as analytical *symptoms*. In addition to symptom generation using quantifiable information, using qualitative information from human operators can produce heuristic symptoms. In this way heuristic symptoms are generated, which can be represented as fuzzy variables (*e.g.*, small, medium, large) or as vague numbers (*e.g.* around a certain value).

The task of *Fault Diagnosis* consists of determining the type, size and location of the fault as well as its time of occurrence based on the observed analytical and heuristic symptoms. If no further knowledge of fault symptom causalities is available, classification methods can be applied which allow a mapping of symptom vectors into fault vectors. If, however, a-priori knowledge of fault-symptom causalities is available, diagnostic reasoning strategies can be applied. Three types of strategies are in use:

- Classification methods, including geometric, statistic, neural and polynomial classifiers, all use reference patterns for learning. Their structure is not transparent but they can be adapted during use.
- Inference methods are based on linguistic rules. Most of the time they are given in a fuzzy way. Expert systems fed with fuzzy rules allow a fuzzy decision-making, the so-called "approximate reasoning". The problem with

this approach is the long time needed to develop the rules and the difficulties involved in adjusting the rule base.

- Adaptive neuro-fuzzy systems combine the strengths of both methods. The idea is to obtain an adaptive learning diagnosis system with transparent knowledge representation.

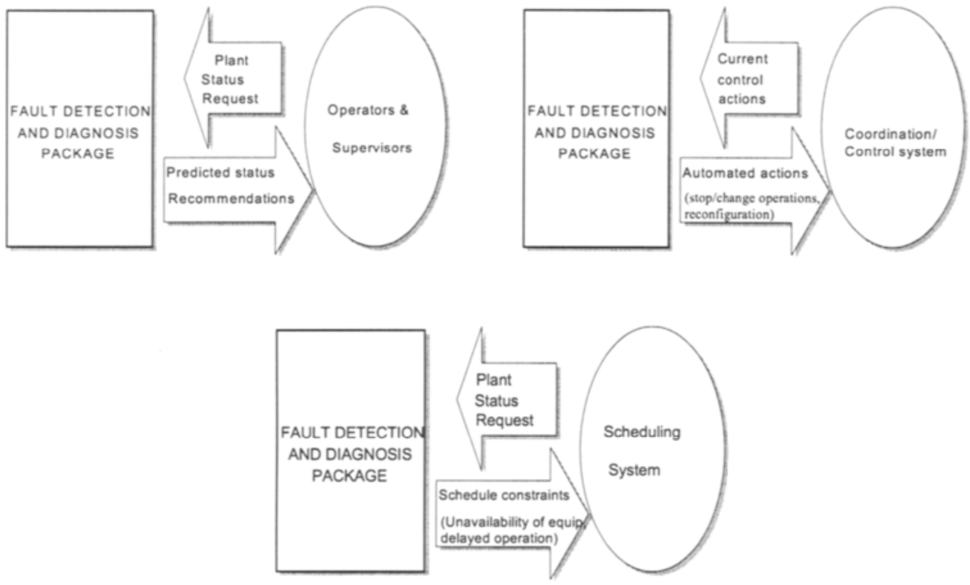

FAULT DETECTION AND DIAGNOSIS PACKAGE
Plant Status Request
Predicted status Recommendations
Operators & Supervisors
FAULT DETECTION AND DIAGNOSIS PACKAGE
Current control actions
Automated actions (stop/change operations, reconfiguration)
Coordination/ Control system
FAULT DETECTION AND DIAGNOSIS PACKAGE
Plant Status Request
Schedule constraints (Unavailability of equip, delayed operation)
Scheduling System

Figure 17. Data flows between the Fault Detection and Diagnosis package and other systems

3.5.7 BATCH CONTROL

Batch control has the characteristics of a hybrid system in itself. Since batch processes typically are neither purely continuous nor discrete, but instead have characteristics of both, the automation of batch processes is considerably more complicated than that of continuous or discrete processes. In a continuous process, process control is required only to monitor that the system is working within the optimum limits, and if not, to apply local control in order to return the system to within the optimal limits. In a batch process, the control system must also detect when a phase has been completed, and then change from one dynamic configuration to another dynamic configuration, with changes in the local controllers for each phase.

In batch processes there are three different types of process control: basic control, procedural control and co-ordination control. A combination of control activities and control functions of these control types provides batch control, defined as 'a means to process finite quantities of input materials by subjecting them to an ordered set of processing activities over a finite period of time using one or more pieces of equipment.

The S88 standard takes into account two types of information: equipment dependent and product dependent, to structure the description of the batch processes in the form of three models. These models are combined to represent these two types of information. The first model is the *physical hierarchy model.* The other is the *procedural model,* used to describe the procedural control elements. The combination of the equipment control functions and physical equipment the equipment entities are obtained, that has the same name as the physical model level that represent. The third model, is the *recipe structure.* The characteristics of these models are described next.

A multilayer *physical hierarchical model* to describe the enterprise, lower level groupings is combined to form higher levels in the hierarchy. The model has seven levels (Figure 18). At the top there are the three business levels, an enterprise, a site and an area level, all three of which are outside the scope of the s88.01 standard. The lower four levels are the process cells, units, equipment modules and control modules.

Process cells define a logical grouping of equipment necessary to produce one or more batches, though not necessarily a final product; sometimes process cells are called trains. Defining process cells makes production scheduling easier. Considerations when structuring process cells include:

- Establish clear boundaries.
- Functions performed must be consistent regardless of what product is being produced.
- Interaction with other process cells minimal and when necessary, conducted at the same or higher entity level i.e., process cell to process cell; and
- Maintaining consistency so operators interacting with similar entities do so naturally and without confusion.

Units are a collection of equipment and control modules in which major processing activities, such as react, distil, crystallize, make solution, *etc*., can be conducted. Unit characteristics include:

- Operate on only one batch at a time.
- Cannot acquire another unit; and
- Operate independent of other units.

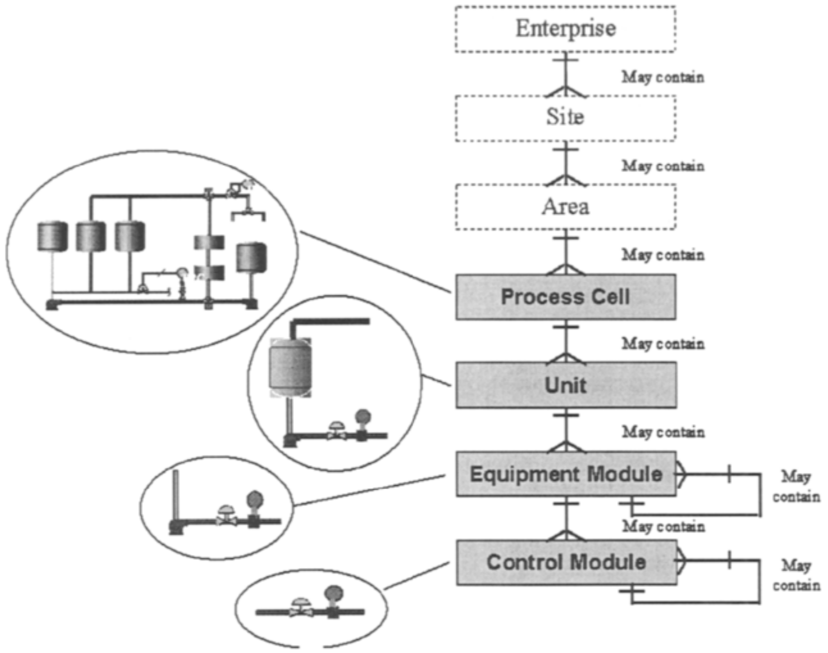


Figure 18. S88 Physical mode

Equipment modules are functional groups centered around a piece of processing equipment that carry out defined activities, such as header control, dosing, weighing, jacket service management, scrubber control, etc. A collection of control modules can become an equipment module if the collection executes one or more equipment phases.

Control modules are the lowest grouping of equipment capable of carrying out basic control. For example, solenoids and limit switches combined can form Off/On valve control modules, and transmitters and valves can be combined into PID control modules.

The physical model described is used to describe the plant in terms of physical objects, the equipment-dependant information.

The *procedural model* describes the interaction with upper levels (coordination, scheduling and planning), which involve sequential control activities.

Procedures are the strategy for carrying out batch activities within a process cell. Procedures, such as clean in place, do not always produce a product or a product intermediate.

Unit procedures provide a strategy for carrying out operations and methods in a contiguous manner within a single unit. A unit procedure can execute concurrently on different units.

Operations are independent processing activities that usually result in a chemical or physical change in the material being handled. Operations include the instructions necessary for the initiation, organization, and completion of activities such as prepare reactor, charge, heat, cool, react, *etc.*

Phases accomplish a specific process-oriented task, can be executed sequentially or in parallel, can be self-terminating, and need to account for exception condition handling. When defining phases, a few of the considerations include:

- Consistent use of predefined states, such as holding, held, restarting, failing, etc.
- Consistent use of predefined commands, such as hold, stop, abort, etc.
- Definition of the modes for each phase, and how the phase will respond to each mode. For example, is a single-step mode needed for troubleshooting?
- Definition of exception handling and recovery mechanism's; and
- Data collection of phase-related activities.

The *recipe model* is hierarchical, the degree of details depending of the level of the physical model at which the recipe is specified. The standards for batch process automation define four major categories of recipes:

- General Recipes, which are maintained at the corporate level (typically within the ERP system) and which permit companies to make the same product in plants around the globe on a variety of equipment, but based on the same source recipe.
- Site Recipes, which typically reside on manufacturing execution systems (a layer that lies between plant floor control and ERP systems). Site recipes define local site control of recipes across different hardware platforms, such as those supplied by control systems manufacturers, Honeywell, Fisher-Rosemount, Rockwell Automation, ABB, etc.
- Master Recipes, which are the specific procedures that actually execute the recipe in a particular manufacturing area, known as a process cell.
- Control Recipes, which are the running recipes in the process cell control systems.
- The master recipe is more general than the control recipe. In the master recipe equipment clauses, for example, are stated and quantities are usually specified normalized. A master recipe can also be used for manufacturing a large number of batches.

A control recipe is used for manufacturing a single batch only. The control recipe is made from a master recipe adding batch-specific information. For example the equipment and the exact quantities of ingredients to be used.

The hierarchical structure makes the systems easy to understand and the layer structure allow encapsulating the not relevant details of one level to the others; each level contains only the relevant information for it.

3.5.7.1 Integration

The new communication technologies and the advances in computer hardware have permitted the solution of many problems in control and production management in the process industries, in part through a more effective and intensive data exchange between different sectors of the process. At the same time, this has led to a growing complexity in these systems. The development of applications for these systems using only the operating system and some programming language is today an unaffordable task both from economic and technical points of view. To develop these complex applications requires the integration of different specific subsystems developed with various tools in a single application that fulfils the performances required by the system.
In the design of the integration architecture, the following aspects should be considered:

- Open interfaces to other network levels. Client/server scheme, event operation supports.
- Time synchronization.
- Data translation.
- Data abstraction.

The general communication architecture uses the client/server paradigm. At the higher level (near real time data) the communications are implemented over a higher-level protocol via a TCP/IP network. In some non-critical communication task the common higher-level Internet protocols are used. The planning and scheduling system and the control system share a common database that contains the recipe descriptions and plant description data. The most detailed level of recipe information is used mainly for the control system; while the planning and scheduling system uses less detailed information to perform its task. The advantage of this common database is that a common interface can be used to input data, so that data consistency between the two modules is ensured because both systems are using the same data. Additionally the communication between the Planning system and the Control system is carried out using a data server, which manages the queues of the different messages and delivers them on request. The use of a data server allows easy upgrades and changes of the different modules. It also allows easy monitoring of the communication between the different modules of the whole system. The third main advantage of this system is its open architecture that allows communicating with third parties software and hardware.

An effective way to represent the integration model is through the use of a hierarchical layered model (Figure 2), with each of the items of Item 1(a) being subordinate, or below that above it in the list, in the model. At the same time, the aggregation of Item 1(b) occurs naturally, going upward in the same layered fashion.

- The deployment of this architecture over the network is shown in Figure 19. There are three basic elements in the system platform that guaranty connectivity, openness and reliability of each modules of the architecture:
- The *operating system* guaranties an independent, quasi-parallel execution of modules and ensures independence from the specific hardware.
- The *communication system* ensures the co-operation of modules in a standardized way.
- The *configuration system* serves to build up a software topology of the set of available modules by instantiating and connecting the different modules.

Any system integration has to be built in terms of these elements. Each specific functional software module has a common application program interface (API) which must be made accessible to the others.

The *communication system* is the only means for information interchange between system modules. It must support both, the exchange of information between modules located on the same processor-board as well as exchange between modules located on different boards connected through a bus-system (network). A standardized protocol has to be defined to ensure uniform data formats and a fixed set of messages.

To connect the modules over the network a core bus (see Figure 20) is defined based on the OSI base reference model. It implements any middleware technology from simple TCP/IP protocol or other higher level Internet protocol like HTTP to CORBA. Over this middleware technology two additional layer have to be built one for object mapping specific for the selected middleware technology and other for message parsing. The inter-process communication is made through DDE/OLE technology, and RPC, the same parsing mechanism is used to processing the cross application message. In the figure the system modules are shown over the deployment network, also, in these figure the information transmitted is indicated.

Today the availability of a communication infrastructure based on TCP/IP protocol and the use of standard Internet application protocol (HTTP, SMTP, and FTP) allows much easier sharing of information than was possible just a few years ago. A web-based communication mechanism has been recently proposed (Nougués, 2000). The communication mechanism follows a three-layer communication structure (Figure 21) that uses the core bus to exchange data. The core bus implementation depends on the specific middleware technology used to distribute the application. Here a simple TCP client server protocol are used to communicate the data server with the servlet, and the HTTP protocol to communicate the client application (Browser and applets) with the web server and servlets.

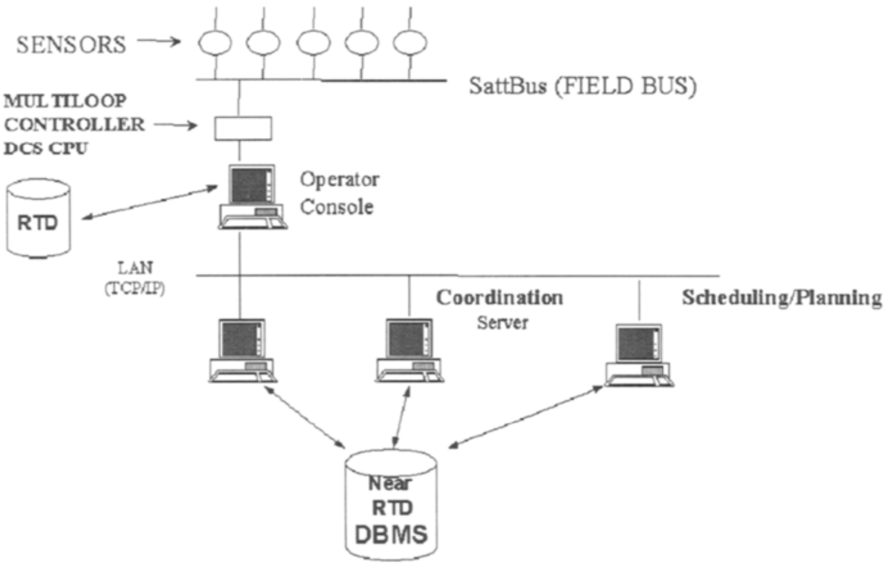


Figure 19. Deployment of system architecture.

Today the availability of a communication infrastructure based on TCP/IP protocol and the use of standard Internet application protocol (HTTP, SMTP, and FTP) allows sharing information much easier than a few years ago. A web-based communication mechanism has been recently proposed (Nougués and Puigjaner, 2001).

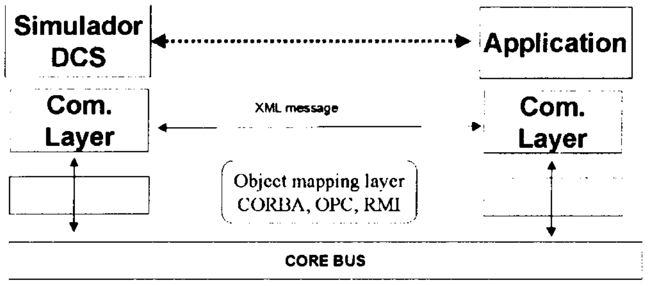


Figure 20: Network core bus.

The communication mechanism follows a three-layer communication structure (Figure 21) that uses the core bus to exchange data. The core bus implementation depends on the specific middleware technology used to distribute the application. Here a simple TCP client server protocol are used to communicate the data server with the servlet, and the HTTP protocol to communicate the client application (Browser and applets) with the web server and servlets.

In Figure 21, a detailed modules relationship is shown and the type of communication between the modules is also indicated. It is important to remark that the simulator engine, Data Server and Simulator can be run on different computers i.e. can be distributed object over the network.

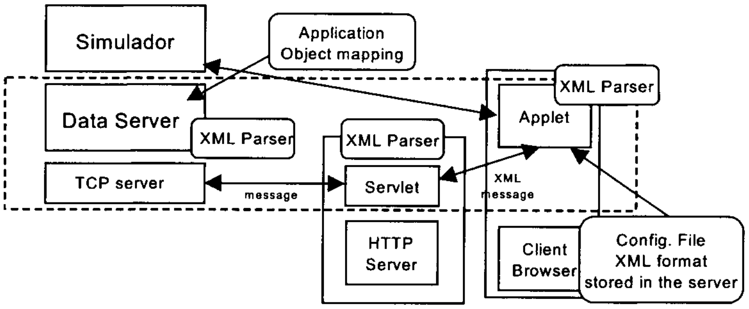


Figure 21. Subsystems and information flow.

Message exchanged between the different modules is formatted in XML syntax. That way, XML parser is implemented in each sub-module as can be seen in the figure.

3.5.7.2 Optimisation Techniques

Mathematical Programming Techniques

Many Mixed Integer Linear Programming (MILP) models have been proposed for scheduling problems arising in the process and related industries (Pekny and Reklaitis, 1998; Shah, 1998, Pinto and Grossman, 1998). While a major advantage of the mathematical programming approach is that it provides a general framework for modeling a large variety of problems, its main limitation lies in the potentially very large computational effort required to solve problems of practical size, which can be a severe limitation for industrial applications. Additionally, the use of linear models to describe the manufacturing environment, objectives, constraints and policies can lead to unsatisfactory or unfeasible solutions. To overcome the combinatorial explosion characteristic of this type of problem, several authors have developed logic-based optimization methods, such as Generalized Disjunctive Programming (Raman and Grossmann, 1994), Constraint Programming (Hentenrych, 1989), and/or combinations of both with generic MILP methods to reduce de search space (Harjunkoski and Grossmann, 2001). Through these means a substantial reduction in the computation effort by several orders of magnitude can at times be achieved with large size problems.

Stochastic Optimization

Non deterministic methods have seen increasing application in the optimization of large complex plants. Improved random searh algorithms have been reported (Salcedo, 1992). Metaheuristic algorithms (Simulated Annealing, Genetic Algorithms, Artificial Neural Networks, Tabu Search) are also finding a widespread application in process engineering. These methods may be efficiently combined with rigorous approaches to find quasi optimal solutions within reasonable computing time for complex hybrid production scenarios

Specific Solutions for Specific Problems

The development of a general scheduling tool is an extremely ambitious objective and in fact given present understanding of the nature of scheduling problems constitutes an impossible task. Production facilities tend to be very particular because company policies, objectives and constraints are very specific and of different nature. Specific constraints in batch chemical processing made this situation even harder. For these reasons, scheduling methods need to be adapted and user interfaces rebuilt each time a new application is encountered. However, the modelling of batch processes may follow a common framework and share the same basic simulation procedures, user interfaces and data organization and management. Consequently, most of the computer code developed for the general case need not be rewritten each time a new application is encountered, but instead can be reused. This is one of the main advantages claimed by Component Technology. Distributed components and distributed computing have been recognized as the way to reduce software complexity and cost as well as to increase software flexibility and extensibility.

3.5.8 REFERENCES

Antsaklis,, P.J., 2000, "Scanning the Issue/Technology", *Proceedings of the IEEE* **88**, 879-887.
Bagajewicz M., 2001, "Review of Recent Results in Instrumentation Design and Upgrade for Process Plants". *Proceedings of the 4th IFAC Workshop on On-Line Fault Detection & Supervision in the Chemical Process Industries*, June 8-9, Seoul, Korea.
Barton, P.I. and C. C. Pantelides, 1994, "Modeling of Combined Discrete/Continuous Processes". *AIChE Journal.* **40**(6), 966-979

Chouaib, Ch.,. Tona, R.V., Espuña, A. and L. Puigjaner, 2001,"On-line application of Integral Dynamic Data Reconciliation", PRESS'01, Florence

Engell, S.,Kowalewski, S., Schulz, C. And Stursberg, O, 2000, "Continuous-Discrete Interactions in Chemical Processing Plants", *Proceedings of the IEEE* **88**, 1050-1068.

Graells, M., Cantón, J.,Peschaud, B.and L. Puigjaner, 1998, *Comp. Chem. Eng.*, **22S**, 395-402

Harjunkoski, I. and J. E. Grossmann, 2001, "Combined MILP-Constraint Programming Approach for the Optimal Scheduling of Multistage Batch Process". Proc. ESCAPE 11, Scanticon, Denmark.

Hentenryck, P. V., 1989, "Constraint Satisfaction in Logic Programming", MIT Press, Cambridge, MA.

ISA-S88.01-1995. "Batch Control. Part 1: Models and Terminology". Approved February 28 1995

ISA-S88.02. "Batch Control. Part 2: Data Structures and Guidelines for Languages". Draft 13. February 1999

ISA-dS95.01-1999, "Draft Standard, Enterprise - Control System Integration, Part 1: Models and Terminology". Draft 14, November 1999

Isermann, R. and Ballé P., 1997, Trends in the application of model-based fault detection and diagnosis of technical processes, *Control Eng. Practice*, **5**, 709-719

http://www.opcfoundation.org

Leonhardt, S. and Ayoubi, M., 1997, "Methods of Fault Diagnosis", *Control Eng. Practice*, **5**, 683-692.

Morari M., Lee, J.H., 1999, "Model predictive control: past, present and future", *Computers & Chemical Engineering*, **23** (4-5) 667-682.

Mockus, L. and Reklaitis, G. V., 1996, "Continuous Time Representation in Batch/Semicontinuous Process Scgheduling-Randomized Heuristics Approach", *Computers & Chemical Engineering*, **S20** ,S1173-S1178.

NAMUR, "GO Koncept", NE33, February 1987.

Nougués, J.M. and L. Puigjaner, 2001, "Web-Based Process Simulator", Proc. ESCAPE-11, Scanticon, Denmark.

Pekny, J. and Reklaitis, G.V., 1998, "Towards the convergence of Theory and Practice; a Technology Guide for Scheduling/Planning Methodology". *Foundations of Computer-Aided Process Operations* (Eds. J.F. Pekny and G.E. Blau), AIChE Symposium Series no. **320**, 91-111.

Pinto, J. M. and Grossmann, I.E., 1997,"A Logic-Based Approach to Scheduling Problems with Resource Constraints", *Computers & Chemical Engineering*, **21**, 801-818.

Puigjaner, L., Espuña, A., 1998, "Prospects for Integrated Management and Control of Total Sites in the Batch Manufacturing Industry". *Computers & Chemical Engineering*, **22**, 87-107.

Puigjaner, L., 1999, "Handling the increasing complexity of detailed batch process simulation and optimisation" *Computers & Chemical Engineering*, **23 S** (ISSN: 0098-1354), S929-S943.

Raman, R. and I. E. Grossmann, 1994, "Modeling and Computational Techniques for Logic based Integer Programming". *Computer and Chemical Engineering*, **18**, 563-578.
Rickard, J.G. Machieto, S. and Shah, N. 1999, "Integrated Decision Support in Flexible Multipurpose Plants", *Comput. Chem. Engng.* Suppl.,**23**,. S547-S550
Ruiz, D., Nougués, J. M., Cantón, J., Espuña, A. and L. Puigjaner, 2000, "Fault Diagnosis System Support for Reactive Scheduling in Multipurpose Batch Chemical Plants". *Computer Aided Chemical Engineering – ESCAPE 10* (Ed. S. Pierucci), Vol. **8**, 745-750.
Salcedo, R. L., "Solving Nonconvex Nonlinear Programming and Mixed-Integer Nonlinear Programming Problems with Adaptive Random Search". *Ind. Eng. Chem. Res.*, **31**, 262.
Saraga, M. I., Kahoraho, E., Burgos, A. and J. I. Llorente, 1999, "An evaluation of different manufacturing generic models from the reuse point of view". Proceedings 7th IEEE International Conference on Emerging Technologies and Factory Automation (Ed. J. M. Fuertes), Vol. 1, 767-773, IEEE Society, Piscataway, N.J.
Shah, N., 1998, "Single- and Multisite Planning and Scheduling: Current Status and Future Challenges". *Foundations of Computer-Aided Process Operations* (Eds. J.F. Pekny and G.E. Blau), AIChE Symposium Series no. **320**, 75-90.
Silver, E., Pyke, D. and R. Peterson, 1998, "Inventory Management and Production Planning and Scheduling". John Wiley and Sons.

ACKNOWLEDGEMENTS

The financial support from the European Community (project Global-Cape-Open-IMS 26691), the Generalitat de Catalunya (through CeRTAP - Centre de Referéncia en Tecnologies de Producció") and CYCIT (project QUI99-1091) are thankfully appreciated