COMPUTER AIDS FOR PROCESS MODEL-BUILDING

by

Richart Vázquez-Román

A thesis submitted for the degree of
Doctor of Philosophy of the University of London
and for the Diploma of Membership of the Imperial College

Department of Chemical Engineering and Chemical Technology
Imperial College of Science, Technology, and Medicine
London SW7 2BY

March 1992

To the memory of my father

To my mother, wife, and daughter

ABSTRACT

Model-building is a difficult task which demands experienced engineers in order to develop suitable models. However, this activity can be considerably simplified by the utilization of computers. Some flowsheeting packages already exist which provide facilities for users to write their models in terms of equations and/or in terms of predefined modules which contain either equations or procedures.

The possibility of allowing the user to define his process models in terms of more elementary operations or physical mechanisms was explored in this study. A suitable computer program initially would verify the physical description for consistency and completeness and subsequently build up the appropriate describing equations. This package would also inform the user of the degrees of freedom, suggest consistent sets of free variables, and the variables which may be specified to yield consistent initial values.

This research has resulted in a new approach to modelling. The approach consists of four steps: the process representation, analysis of the representation, the build-up of the model, and the analysis of the model. The process representation translates the original system into an appropriate format to facilitate the internal manipulation of a computer program.

A prototype has been developed incorporating the proposed approach. The package was written in Smalltalk so that the object-oriented capabilities could be utilised. It has resulted in a user-friendly system which confirms that the approach presented is feasible. It also demonstrates that the introduced concepts provide great capabilities to model lumped parameter process systems.

ACKNOWLEDGEMENTS

I would like to express my most sincere gratitude to Professor R.W.H. Sargent and Professor J.D. Perkins for their invaluable advice through many hours of discussion during the development of this work, and their help in the preparation and writing up of this thesis.

Many thanks are also due to the staff and colleagues in the Centre for Process Systems Engineering. In particular, thanks to Jose Luis Morales for his help and orientation. Also thanks to Diane Biss for their unconditional assistance in the writing up of earlier version of the thesis.

Finally, I am indebted to the CONACyT from Mexico and the ORS Awards Scheme from U.K. for their continuous financial support. The partial support "at the right moment" by the Centre for Process Systems Engineering at Imperial College is also recognised.

CONTENTS

NOTATION INDEX

Listed here are most of the symbols used in the context of the thesis together with a brief description name. Where applicable the dimensions are also given in terms of mass (M), length (L), time (t), temperature (T), force (F), and energy units (E). In general the dimension (M) is construed as mass except where moles is specifically stated. In general, entities like phases, reactions, connections, etc are indicated in superscripts; however, they may appear as subscripts when the superscript indicates a power. To clarify the notation of the equations we adopt the following conventions: all vectors and matrices are denoted in bold symbols, all vectors are column vectors, and the transpose of a vector **v** is denoted as **v**'.

Latin Symbols

a	Absorptivity
A	Area (L^2)
A	Frequency factor in (4.38)
c	Concentration (M/L^3)
C	Valve constant
d	Density (M/L^3)
D	Mass diffusivity (L^2/t)
e	Energy rate (E/t)
E	Energy flux (E/tL^2)
E	Activation energy in (4.38)
f	Mass flowrate (M/t)
h	Heat transfer coefficient (E/tTL^2)
H	Specific enthalpy (E/M)
H	Molar partial enthalpy in (2.8) (E/M)
J	Diffusive molar flux (M/tL^2)
k	Mass transfer coefficient (M/tL^2)
k	Thermal conductivity (E/tLT)
K	Equilibrium ratios
Kr	Constant of reaction
l	Thickness (L)
m	Mass (M)
M	Molecular weight
nc	Number of chemical species
N	Molar flux (M/tL^2)

P	Pressure (F/L^2)
r	Molar rate of reaction (M/t)
R	Gas constant (E/MT)
R	Molar production rate (M/t)
s	Stefan-Boltzmann constant (M/t^3T^4)
t	Time (t)
T	Temperature (T)
U	Specific internal energy (E/M)
v	Velocity (L/s)
w	Total mass flow (M/t)
x	Mass fraction
y	Mol fraction
z	Rectangular coordinate (L)

Greek Symbols

α	Absorptivity
δ	Parameter in (4.12)
Δ	Finite difference in state property
ε	Emissivity
Φ	Transfer matrix in (4.23)
μ	Chemical potential (E/M)
μ	Viscosity (M/Lt)
σ	Stoichiometric coefficient
ζ	Dimensionless parameter in (4.23)

Subscripts

b	Blackbody
e	Equilibrium
i	Chemical species i
t	Turbulent flow

CHAPTER ONE

INTRODUCTION

"I don't have any solution but I certainly admire the problem"

Ashleigh Brilliant

1.1.- Importance of Process Model-Building

In several fields of pure and applied science, simulation is an increasingly popular subject that is taught in universities and extensively used in industry. It permits investigation of complex interactions which are expensive and risky to test in practice. A model is proposed and then simulation is performed in order to provide insight into real situations by interpreting the model's solution. In chemical engineering, simulation has been used not only for studying the behaviour of existing process units but also for designing new ones.

The proposed model is always a simpler representation of reality; it is a formal symbolic representation of the system. Therefore, any model is merely an attempt to duplicate the original behaviour by following laws or rules of similarity or analogy. Indeed, an important aspect of the modelling problem is the identification of the significant features of the behaviour for the purpose in hand, and the construction of a model which adequately reproduces these features without irrelevant complexities [Sargent, 1983].

A model can be a computer program where the program text can symbolically represent the system. The syntax and semantics of the programming language provide the formalism required to make the model explicit. In that case the program is the model and the model is the program. In this work, however, a model refers to the mathematical description of a process, i.e. the mathematical model. So, a model is a mapping of the relationship between the physical variables onto mathematical structures like algebraic equations, differential equations or systems of differential-algebraic equations.

The activity of modelling is focussed on solving specific problems. In earlier times, models were often purely empirical. As a result, only models applicable to special cases were available and they very often failed because they did not consider a variety of effects.

Process model-building represents the procedure undertaken to generate a process model. Its practice has been revolutionised by the advent and rapid growth of digital computing. The feasibility of automation has been extensively proved. New computing facilities, in software and hardware, have allowed the development of more sophisticated models. Some of the advantages of using computers for process model-building are to:

- Avoid repetitive work.
- Save time.
- Reduce sources of error when performing tedious tasks.
- Secure consistency and accuracy of results.
- Provide advice to less experienced engineers.
- Assist experts in their logical reasoning about a process.
- Improve model quality.

Evolution in computing has increased the interest in developing packages for process model-building. The extensive use of these packages has promoted the need to have more flexible and user-friendly systems. New software development in process engineering should include more flexible external interfaces for on-line modelling. So, modelling tools may increase the productivity of the modeller.

1.2.- Current Techniques for Process Model-Building

There are two different levels in modelling process systems: units and processes. Processes are modelled as sets of linked units and unit models are derived from physical laws.

In conventional flowsheeting packages, unit models are developed externally and then added to the package's model library. In this form these packages provide the users with a predefined set of unit models where the equations and variables are

normally hidden. A process is then modelled by describing it as a set of interconnected units. In the initial stages of flowsheeting programs, the interest of modelling was focussed on the solution of steady-state designs rather than dynamic behaviour.

Nowadays, the state of the art in the field of process flowsheeting includes three types of approach: the sequential-modular, the equation-oriented and the simultaneous-modular approach. An executive part of the flowsheeting program controls the data input to the program, checks the input for feasibility, controls the progress of the calculations, stores the model outputs, sometimes controls the convergence and presents the output in a suitable form. In all these approaches, users may describe their flowsheet using a "modular" input language [Perkins, 1986].

In the sequential-modular method, unit calculations are performed by procedures or modules which produce outputs from given inputs. A process consisting of interconnected units is evaluated by executing the procedures or modules in sequence. So, the results of one unit are used as inputs to the next. The order of evaluation normally agrees with the natural order in which the equipment information is input. The sequential-modular approach is normally good at solving steady-state simulations. However some problems may appear. One of these problems is, for instance, the case of recycles in a process. These are handled by guessing their values and then iterating on their values until convergence, i.e. tearing of streams. Another problem arises when design specifications are defined. Since modules in this approach calculate outputs from inputs, one can not easily specify an output. This problem is handled by placing a kind of root finding procedure around the modules and adjusting a parameter in the module. This procedure has been named "a control block". It may generate loops within loops.

In the equation-oriented method units are described by equations. The executive program extracts from a model library all the equations representing the units. Once the equations are assembled, standard numerical methods can be used to solve the non-linear set of equations. For the sequential modular approach these equations must also be produced for writing a module but in the equation-oriented approach it is not necessary to convert these equations into procedures, neither do the units have the rigid structure of outputs and inputs. Thus specification of outputs or recycle loops causes no special problems.

The main idea in the simultaneous-modular approach is to combine the advantages of the other approaches. However, there is some disagreement in what features should be included. Most of these suggestions seek to retain the units modelled by using the same modules as in the sequential-modular method but using a different flowsheet calculation. E.M.Rosen[1980] identifies two cases of the simultaneous-modular approach:

- The simultaneous solution of torn recycle streams and design specifications.
- The two-tier approach.

The two-tier approach involves solving at each iteration two models. One of these models is used to generate guesses for use in the other. The first one can be a linear or ideal model while the second is the rigorous case [Perkins, 1983].

Dynamic flowsheeting packages can be divided in a similar way to that for steady-state analysis. In fact, the development of currently available dynamic flowsheeting packages has followed their earlier counterparts [Cameron, 1983]. The equation-oriented category represents models as sets of differential-algebraic equations and the integration is by simultaneous solution of all model equations. The sequential-modular technique includes subroutines or modules which are solved sequentially.

Two different strategies have been detected in the sequential-modular strategy [Perkins, 1986]. In the first strategy, all the calculations associated with a particular unit are performed within the unit module, including integration of all the differential equations associated with that unit. In the second strategy, a global algorithm in the executive program is used to integrate all the differential equations.

In spite of the research done in the area of dynamic flowsheeting and compared to the large number of steady-state flowsheeting packages available, very few dynamic programs have been developed to a commercial standard. Some of these packages with a brief discussion of their characteristics have been summarized in Marquardt [1991].

A new trend in modelling tools has been to increase the productivity of the modeller by improving the form of interaction between the user and the computer.

This results in adding capabilities such as documentation, graphics, knowledge-based tools and organization of multi-facetted representations. Some of the packages which have appeared in the literature with improvements in this direction are Design-Kit [Stephanopoulos et al, 1987], SpeedUp [Pantelides, 1988], DIVA [Kroner et al., 1990] and ASCEND [Piela et al, 1991].

Most current flowsheeting packages include facilities for modelling complete processes (flowsheets). SpeedUp, for instance, provides users with the possibility to write their own process model in terms of equations with added subroutines. It also provides a "macro" facility which allows definition of composite units in terms of collections of other units. However, the build up of a model has been rather confined to a description of predefined units containing equations or procedures which reproduce the characteristics of the process. Modelling of a new unit is normally developed externally to the package and then the unit is incorporated into the model library.

There is still a need for the automation of modelling process units. Some work has been done in this direction. For instance, a prototype knowledge-based modelling system based on modelling from first principles has been described by Meyssami and Asbjornsen [1989]. Preisig et al [1989, 1990] use a system decomposition to model physico-chemical-biological systems. A more complete study was done by Piela [1989] who describes the design of a system for building and solving mathematical models. Recently, Stephanopoulos et al [1990, 1990a] have proposed a modelling language for process engineering modelling based on physico-chemical phenomena.

1.3.- Objectives of the Present Work

Although modelling process systems is a widespread activity, examination of the state-of-the-art in computer-aided modelling indicates that most of research is focussed on developing techniques to solve the model. Rather than continuing in this direction, the aim of this study is to research in the earlier step of building mathematical models. It is believed that computers can be wisely used to build models efficiently and reduce the time it takes to develop models, in particular for new units.

The purpose is to explore the possibility of allowing the user to define his process models in terms of more elementary operations or mechanisms, with the program itself building up the appropriate describing equations and checking that the physical description is consistent. The program then informs the user of the physical data and other design parameters required, the degrees of freedom and appropriate free variables.

Indeed, the practising engineer may well be more concerned to know what mechanisms and properties exist in his formulated problem rather than to represent the mathematical description of the model according to some predefined units since they provide little information about the process phenomena.

One of the basic assumptions taken in this work is that the real world behaviour can be conveniently abstracted into mathematical relationships.

Models can be classified into steady-state and dynamic models. A process is in steady-state when none of the involved terms changes with time. Otherwise, the model is dynamic.

A large number of physical and chemical processes of engineering interest has been modelled by considering operation in steady state. Historical standard chemical engineering design techniques have dealt almost entirely with steady-state operations. However, during the course of normal operation it is inevitable that small disturbances occur. These disturbances may cause the system to move away from its design state.

Cameron [1983] and Perkins et al [1987] support the use of dynamic models. They give a list of applications of dynamic models which is provided below:

- Start up and shut down of plants.
- Effects of changing process parameters.
- Hazard analysis.
- Response to equipment failure.
- Operator training.
- Operability studies.
- Control strategies.

Since dynamic models of these applications represent more realistically the behaviour of a real process, the build-up of dynamic models is preferred in this study. In fact, the steady-state operation is included in the scope of the dynamic model. The steady-state in a dynamic model is visualised as a condition of equilibrium where time derivatives are zero.

A further classification of dynamic models can be done according to the kind of differential equations describing the system. Hence models represent either lumped parameter or distributed parameter systems.

Briefly, a lumped parameter model is one in which spatial variations have been ignored. From a mathematical point of view, lumped parameter systems are dynamic systems which are completely described by sets of ordinary differential equations. In contrast, if detailed variations from point to point are taken into account the system is represented by a distributed parameter model. As a consequence, distributed parameter systems are described by partial differential equations. The spatial variation may be in only one dimension or may be in two or three dimensions.

However, because mathematical procedures for the solution of lumped parameter systems are simpler than those for distributed parameter, very often the latter is approximated by an equivalent lumped parameter system.

Hence, this initial study is restricted to consideration of lumped parameter systems.

1.4.- A New Approach for Modelling

With the purpose and goals defined above, a solution has been conceived and implemented. The general strategy followed is outlined in this section.

The approach proposed in this study consists of four common-sense steps: the process representation, verification of the process representation, build-up of the model, and analysis of the model.

The process representation is a very important step in this approach. It concerns the clear definition of the problem. A good problem definition comes from answering questions such as the following: What part or parts of the system are to

be modelled? What are the properties and mechanisms occurring in the system?. Subsequently, the modelling problem can be identified based on a proper description of the process.

The development of a special purpose language was used to obtain a solution for the process representation. This language consists of symbols which, used in a proper form, express all the points of interest in the definition of the modelling problem. The aim was to describe the system in a very natural language. Utilisation of complex terminology or extended vocabulary was avoided for the sake of better communication and the hope of generation of more user-friendly systems. An important issue in process representation is that its description is completely separated from its solution.

However, errors may appear even when using simple languages. Packages should be developed assuming that users may make mistakes. The power of a package is in its ability to sort out users'mistakes. The process representation provided by the users is analysed to ensure the problem has been properly described in terms of the terminology implemented.

Once the modelling problem has been completely and properly represented, the build-up of the model is performed. If the analysis of the representation is successfully achieved, the build-up process does not require an user interaction to produce the mathematical model. The problem is internally manipulated and dealt with as a synthesis problem.

Once the mathematical relations have been assembled, they have to be arranged into a solution strategy, that is, decisions have to be made on which variables are to be determined. Degrees of freedom have to be eliminated and initial conditions specified in order to get the numerical solution. All these features are part of the model analysis.

The block diagram shown in Figure 1.1 provides a graphical description of the interrelationship of these steps. Note that information travels in both directions only between the process representation and the analysis of the representation.

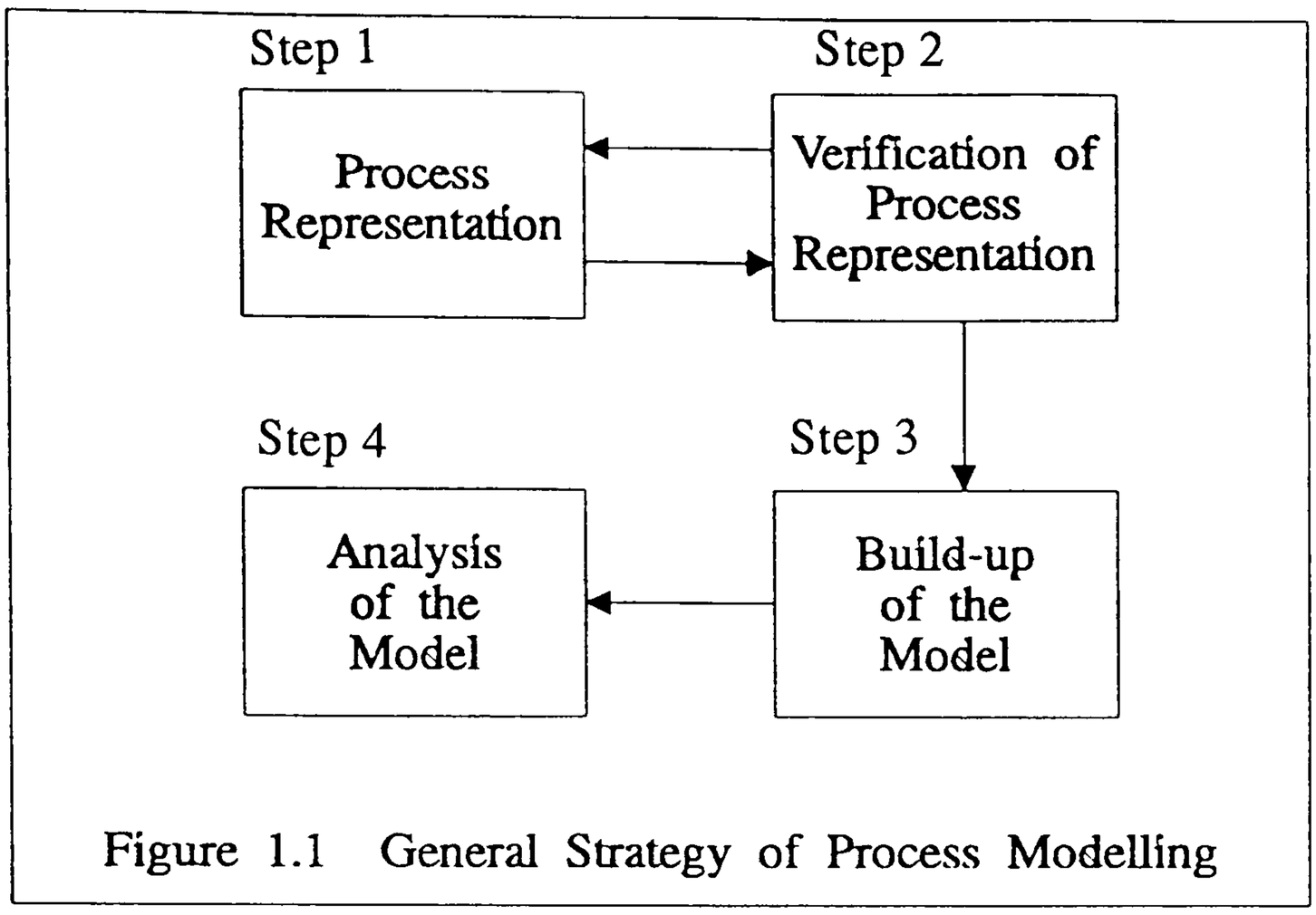


Figure 1.1 General Strategy of Process Modelling

1.5.- Outline of the Thesis

This thesis is divided in six chapters. A brief outline of the contents of the thesis is as follows:

In chapter one, a brief overview of the evolution and state-of-art of process modelling was presented. Then, the goals of this research were discussed.

Chapter two gives a very broad description of the modelling approach proposed in this study. It identifies the principles regarded as relevant to the lumped parameter modelling area. It also introduces the elements required to describe the system being modelled to facilitate both the description of the transfer mechanisms and the generation of the model.

Then, chapter three is devoted to systematic formal definitions of all the elements introduced in chapter two. It includes a proposal to graphically describe the topology of the system. A language is proposed to declare the properties and transfer mechanisms desired to be modelled.

Chapter four builds on the transfer mechanisms and properties considered for modelling in chapter three. This shows the equations which should be generated given the topology and the complete description of the system.

Chapter five shows that the approach proposed is relatively easy to implement into a computer code. An object-oriented programming language has been used for the implementation. Thus, this chapter starts with a description of the basic object-oriented ideas. Then, it describes the prototype.

Chapter six draws the conclusions about the work and outlines the future directions.

Finally, some examples are presented in the appendix. These examples show how the various themes introduced in the approach to the modelling problem are applied into some cases of study. This demonstrates the versatility of the system.

CHAPTER TWO

THE GENERAL MODELLING APPROACH

"Everything should be made as
simple as possible, but not simpler"

A. Einstein

2.1.- Introduction

In chapter one we have established that the purpose of this research is to generate the mathematical model of a process system given the description of the system in terms of the transfer mechanisms and more elementary operations occurring within the system. Thus, a computing procedure would incorporate the proper information and would generate the intended mathematical model.

In trying to achieve a suitable solution to the modelling problem, a proper breakdown of the system has been imperative. In this chapter, an overview of the modelling approach is presented where double effort was done to generate a proper process representation which then would simplify both the process description in the intended terms and the generation of the model. Several issues concerning the modelling of process systems are also discussed here.

In agreement with the goals of the thesis, all the work presented here was conceived with the idea of supporting the user in modelling the dynamic lumped parameter type, meaning that any dependent variable can be assumed to be a function only of time and not of spatial position.

Generally speaking, the approach is based on the fact that any process system can be decomposed into thermodynamic subsystems. In this way, the mathematical model can be built based on smaller pieces facilitating the model-building process. The simplest cases of thermodynamic subsystems are those parts of the system throughout which all of the properties are uniform, i.e. phases.

Any process system is represented by its characterising phases. Then, each phase is characterised by the values of the properties associated with the phase itself. Thus, the basic primitive objects of our modelling approach are represented by the phases of the system and the properties associated with these phases. The values of these properties cannot arbitrarily be assigned since they are constrained by constitutive relations.

The following analysis in this section considers the system as consisting of a single phase though the generalisation of the main result to the multiple phase case is relatively straight-forward.

Any attempt to characterise the system will unavoidably lead us to the concept of the state of the system. Indeed, the concept of the state is one of the most fundamental concepts in the process of model-building. This concept seems to have two roots: thermodynamics and dynamics. At this stage we analyse the implications from the point of view of thermodynamics and later we will consider the dynamic behaviour.

In thermodynamics, the concept of state is assigned to the condition in which a system exists at any particular time. The thermodynamic state of the system "is the totality of the properties of the system" [Spalding and Cole, 1973]. Hence, the thermodynamic state is defined by specifying the numerical value of the system's properties such as pressure, temperature, volume, internal energy, and enthalpy [Modell and Reid, 1983].

A change in the thermodynamic state of the system occurs if there is a change in at least one of its properties. However, if by a series of interactions with its environment the phase is restored to its original state, then all the properties by which that state was originally characterised must return to their original values. Thus, the state property is one whose value corresponds to a particular condition and is completely independent of the sequence of steps followed to achieve that condition.

The above definition of state is not very practical because the number of properties that are required to specify the state of a thermodynamic system is not always known. However, a large body of experimental data indicates that there are particular types of states that can be specified by delineating only a certain number of properties. These states are called stable states. In principle, non-stable

states can also be specified from a finite number of properties; however, the number of properties is not given by the principles of classical thermodynamics.

Any thermodynamic state is considered stable if the system has a total entropy not less than its total entropy when in another state for which the system has the same volume, internal energy, and mass of the various chemical species. In other words, a state is considered stable if the total entropy among all the compatible states is maximised. An equivalent statement can be referred to the total internal energy: a thermodynamic state is stable if the total internal energy has been minimised among the set of all states compatible with fixed entropy, volume, and mass of each chemical species.

It follows, according to Gibbs' statement, summarised in the Gibbs-Duhem theorem, and more recently reviewed by Feinberg [1979], that the stable state of any phase at a given time is completely determined by the knowledge of two independently variable properties in addition to the masses of each chemical species in the system.

The analysis above implies the existence of equations relating the thermodynamic state properties. In fact, the so called equation of state is one type of these constraint equations. This sort of equation typically relates the pressure, the temperature, and the volume occupied by given masses of the chemical species contained in the phase. Probably the most successful type of equation of state is the polynomial equation that is cubic in the volume. It offers a compromise between generality and simplicity that is suitable to represent both liquid and vapour behaviour. The first general cubic equation of state was proposed by JD van der Waals in 1873.

A second group of equations is represented by mathematical relations where properties are normally expressed as functions of measurable properties like pressure, temperature, or volume. The so called Maxwell equations are normally used as the basis for finding these relations [Modell and Reid, 1983].

In fact, the two independent variables suggested by the definition of the stable state are the total internal energy and the total volume, or the total entropy and the total volume. Then, every added property can be related to these two independent state variables and the masses of each chemical species. However, it is the existence of these relations that introduces the possibility of selecting other

combinations of variables as the two independent variables. For instance, it may be possible to select: enthalpy and pressure, pressure and entropy, pressure and total volume, etc.

2.2.- The Basic Elements of the Process Representation

A conclusion conjectured in section 2.1 is that any process system can ultimately be decomposed into thermodynamic phases whose instantaneous states can be considered stable. Let us now consider some simple examples to show how this decomposition, for the sake of representation, can be applied in any process system being modelled.

The tank shown in Figure 2.1.a is formed of just one phase. The flash tank shown in Figure 2.1.b may be divided into two phases, liquid and vapour, which may be in equilibrium. Even more complex systems like a distillation column can ultimately be decomposed into phases. A distillation column, see Figure 2.1.c, may be divided into plates which contain two phases per plate, one liquid and one vapour phase, plus the phases in the additional units like the condenser and the reboiler.

In general, there may exist a surface enclosing and separating one phase from the rest of the phases in the system which is normally referred to as the boundary. It can be observed, however, that there exist surfaces which enclose two or more phases with the same structure. It seems to be convenient to incorporate this characteristic into the modelling process, in particular when more than one phase is involved in the container. In the context of this research, the sort of surface which encloses at least two phases is referred to as a vessel. Then, the vessel provides instantaneous relations of some of the thermodynamic state variables. To be more precise, the pressures of the phases in the vessel are essentially the same and the sum of the volumes of the phases in the vessel equals the vessel volume. The flash tank already given in Figure 2.1.b is perhaps the most common case of vessel in process systems.

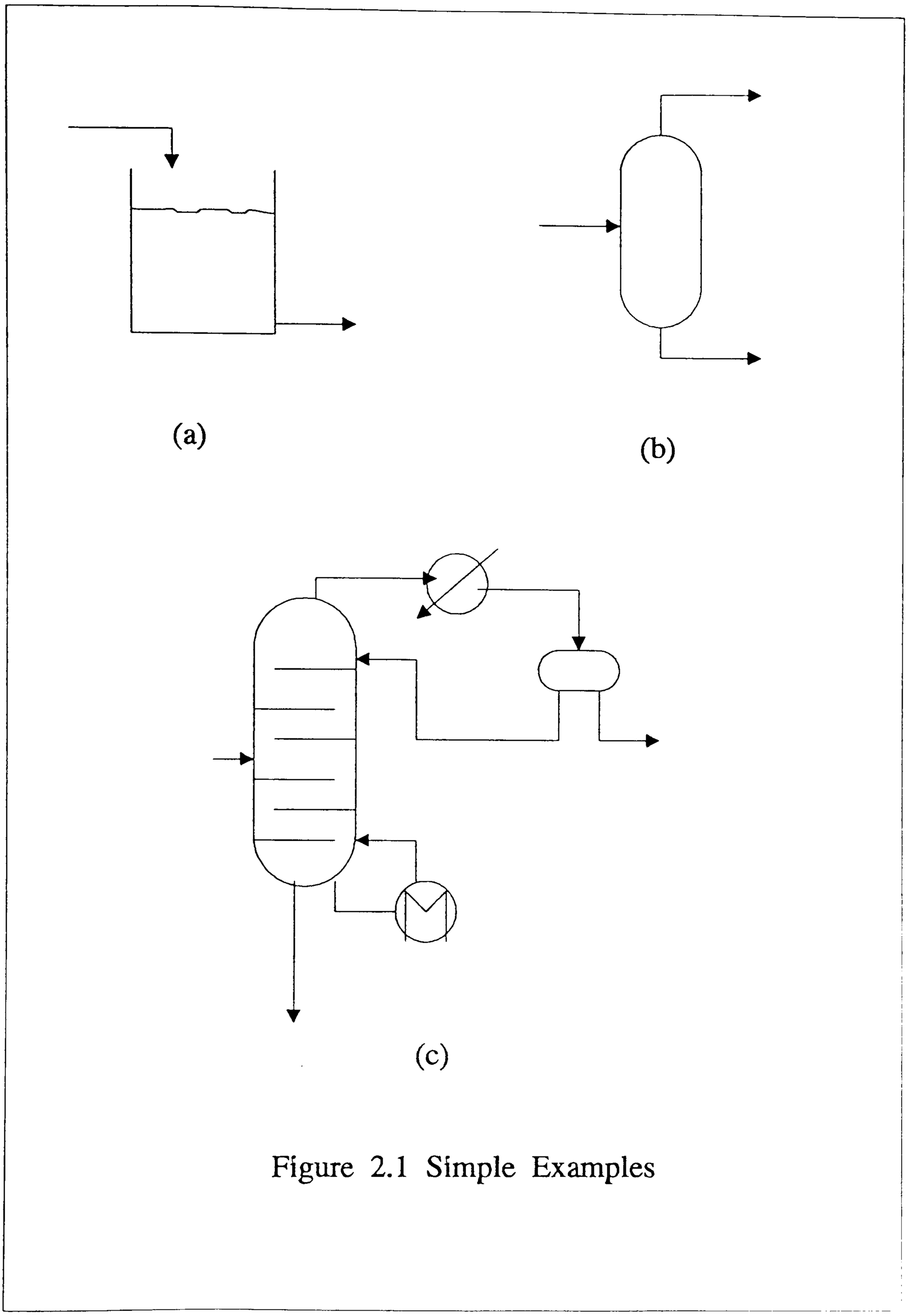


Figure 2.1 Simple Examples

The phases into which the system is divided should not exist as isolated entities. Indeed, all interactions between phases are largely governed by the nature of their common boundary. In fact, the various types of boundaries can provide a classification of phases. For instance, a closed phase is obtained if the boundary is impermeable to the mass flow, and an open phase is obtained if a boundary, which may be rigid or movable, permits a mass flux of at least one chemical species through at least one point.

The interactions between phases are incorporated in the modelling process via "connections". Thus, a connection is a topological element of the model-building process which is used in this approach to interconnect two, and only two, entities of the system. For the moment, let us consider only phases as the interconnected entities.

Connections are used to describe the transfer of quantities through the boundary between any two phases. From classical thermodynamics it is known that there are two elements crossing the border of a physical system: mass and energy. Hence, either mass, energy or both will be transferred through connections.

For the sake of simplicity, connections are elements of the modelling approach with no hold-up so that any mass and/or energy transfer is instantaneously exchanged. Thus, mass and/or energy hold-up is only allowed in the phases.

The transferred energy may be manifested in various forms, e.g. heat, electricity, etc. In general, energy can be exchanged between a phase and its surroundings in essentially three ways:

- By transfer of mass.
- By performing work.
- By transfer of heat.

It is important to keep in mind that any transfer of mass always carries with it a simultaneous transfer of energy. The reason is because every chemical component possesses its own internal energy. Therefore, the transferred mass conveys the energy transfer of at least its own internal energy. Thus, it is impossible to transfer mass without transferring energy though the opposite, transfer of energy without any mass transfer, is quite normal. Hence, any mass transfer declaration will implicitly convey an energy transfer.

Now we are at the point where the description of the system in terms of the transfer mechanisms can be considered. It is believed that the decomposition emerging from the principles of thermodynamics provides an adequate framework to describe the system in terms of the underlying transfer laws. For instance, suppose that the surface separating two phases only allows the transfer of heat, then the transfer law may be declared as heat conduction or heat radiation; if mass is transferred between two phases, the transfer mechanism can be a pressure-drop driven flow, etc.

Note that a reaction may be carried out in any of the phases without altering the topological description of the system.

In this work, any connection is declared to identify only one transfer mechanism. Therefore, if the heat transfer between two phases is the result of both conduction and radiation, two connections will be required.

Once the system for modelling has been selected and properly decomposed into phases, the user should state the transfer mechanisms. However, there may be some remaining parts of the universe interacting with the system being modelled. These parts are conventionally referred to as the environment or surroundings. In this work, however, each external part to the system sharing a common boundary with the system being modelled is referred to as a reservoir. For the purpose of modelling, each reservoir is considered as an infinite source or sink so that its state is unaffected by addition or removal of material or energy.

In fact, simulation of large process systems is sometimes impossible to perform because of memory limitations in computers. The difficulty of including all the process units obliges engineers to divide their systems into subsystems with more convenient dimensions. Then, each subsystem may be reproduced independently from the others. In order to model and analyse any of these subsystems, those parts in the rest of the subsystems, connecting the subsystem being modelled, are also considered "invariable" in their properties. Thus, the concept of reservoir can be extended to represent those units of the subsystems which share a common boundary with the subsystem being modelled.

Thus, all the basic elements which may be considered sufficient to represent and model any process system have been outlined: vessels, reservoirs, phases and connections. Generally speaking, the behaviour of a system can be studied by

monitoring the changes of state that it experiences in its constituted phases as a result of the flows of mass and energy crossing the boundary or because of internal changes within phases, for example due to chemical reactions. The transfer mechanisms can be declared based on the decomposition of the system in order to reproduce the system through a mathematical model.

Let us now consider the following two illustrative examples:

Example 2.1 *The Oxygen Bottle*: Take for instance the simple case of a bottle of compressed oxygen and let us model the act of opening the valve of the cylinder. Starting with the structural information, we may describe the system as having one phase inside the cylinder, one reservoir which is in fact the environment, and one connection to transfer the oxygen from the cylinder to the environment, i.e. from the phase to the reservoir; see Figure 2.2.a. Once the topology of the system has been properly delineated, the transfer mechanisms can be described. Then, for instance, the above connection can be modelled as a pressure-drop driven flow.

Example 2.2 *The Fuel Cylinder:* Suppose now that a large fuel storage tank is used to fill a small cylinder as shown in Figure 2.2.b. The cylinder initially contains a single vapour phase composed of a mixture of the residuals according to the composition of the last charge. The operation starts by pressurising the fuel tank with air which is available from large external storage tanks. Then, an appropriate valve is opened in order to fill the cylinder. The initial vapour phase contained in the cylinder is compressed and homogeneously mixed with the new liquid charge. Thus, the system can be described by four phases: the gas and liquid in the fuel storage tank, and the gas and liquid in the cylinder to be filled. There is one reservoir representing the large external tank, and four connections to interconnect the external tank with the gas in the fuel tank, the gas in the fuel tank with the liquid in the fuel tank, the liquid in the fuel tank with the vapour in the cylinder to be filled, and the vapour in the cylinder with the liquid in the cylinder.

In addition, two vessels may be declared to link the volume relations of the phases in the fuel tank and the phases in the cylinder.

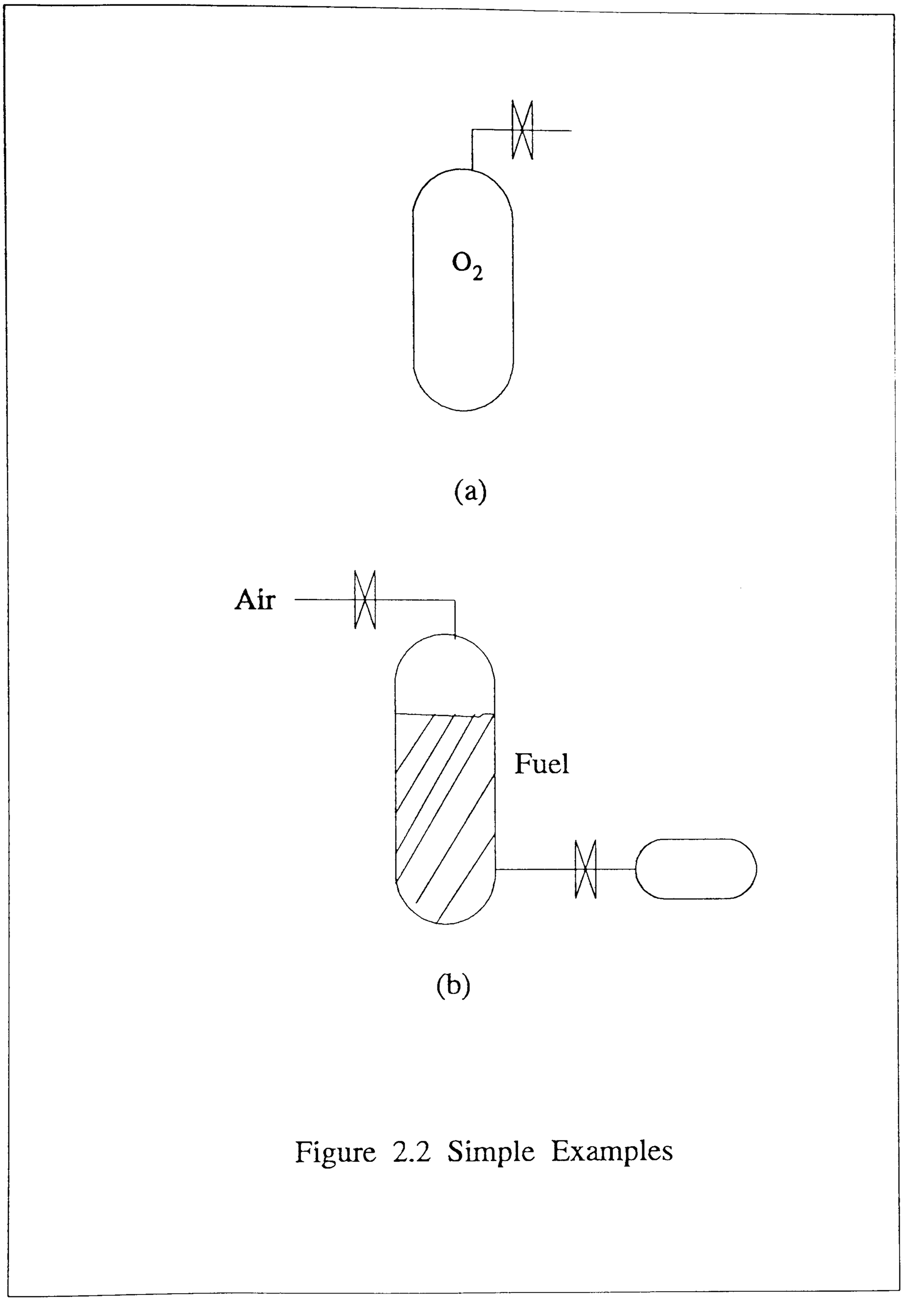


Figure 2.2 Simple Examples

Then, the transfer mechanisms can be defined so that we may consider pressure drop driven flows to model the connections between the reservoir and the fuel tank and between the fuel tank and the cylinder, and, perhaps simultaneous mass and energy transfer between the two phases, gas and liquid, contained in the fuel tank and probably the film model will be used to reproduce the phenomenon.

Phases contained in a vessel are typically interconnected for simultaneous mass and energy transfer. Very often, these phases coexist in equilibrium. However, the condition of equilibrium among the phases contained in a vessel is not a generalised characteristic of vessels. In any case, however, the vessel always introduces volume relations.

2.3.- Utilisation of the Conservation Laws

Having decomposed the system into interconnected phases, vessels, and reservoirs, the problem of modelling is reduced to the utilisation of the conservation laws, coupled with the laws dictating the rates of transfer, together with the relations between the volumes and pressures of phases in vessels.

The principle of conservation of a quantity S states that the variation of S within a phase during a time period is given by the difference between the flow in, the flow out, and the internal change (consumption or production) in that time period. The relation, in words, has the form,

$$\left\{\begin{matrix}Rate\ of\\ Accumulation\end{matrix}\right\} = \left\{\begin{matrix}Rate\ of\\ Flow\ In\end{matrix}\right\} - \left\{\begin{matrix}Rate\ of\\ Flow\ Out\end{matrix}\right\} + \left\{\begin{matrix}Rate\ of\\ Generation\end{matrix}\right\} - \left\{\begin{matrix}Rate\ of\\ Consumption\end{matrix}\right\} \qquad (2.1)$$

Let us now replace the words in (2.1) with mathematical expressions. For convenience, each term in (2.1) is represented by a single variable which may then be characterised by an additional equation. For instance, suppose that the phase A is connected only with the phase B and no reaction is carried out within the phase A. Then, the mass balance in phase A can be written as:

$$\frac{d}{dt}\mathbf{m}^{\wedge} = -\mathbf{f} \qquad (2.2)$$

where nc is the number of chemical species, $\mathbf{m}^{\wedge}$ is the nc-vector of the masses of each chemical species in phase A, and **f** is the nc-vector of mass rates of flow out of each component.

Then, the rate of flow **f** is characterised by an algebraic equation which is related to the transfer mechanism.

In setting up the balances, the generation or consumption of mass of any component inside a phase will be assumed to be due to chemical reaction. The quantity S in (2.1) refers to the mass of each chemical species, the total mass, or the total energy. However, it is a basic principle that the total mass and the total energy can neither be created nor destroyed. Therefore, if we consider that there are no nuclear reactions in the phase being modelled, the terms of generation and consumption in (2.1) will disappear when applied to total mass or total energy. In other words, all changes in the internal inventory of total mass and total energy must be accounted for by interchanges between the phase and its surroundings. The mass balance in reactive phases referred to any of the chemical species in the phase will involve the incorporation of the stoichiometric coefficient and the rate of reaction for each chemical reaction carried out in the phase. These cases will be analysed in chapter four.

The total energy of a phase, in the absence of electric and magnetic fields, is composed of three parts: internal energy, kinetic energy, and potential energy. The internal energy is the stored energy a phase possesses by virtue of the atomic and molecular energy of the matter of which it is constituted. The kinetic energy is the form of energy a phase or any object possesses relative to its state at rest by virtue of its bulk movement; it may be computed by integrating the momentum of the particles of the phase over their velocity from rest to the velocity of the phase. The potential energy is the energy a phase possesses because of its relative position in a uniform gravitational field; it may be evaluated by defining the elevation of the phase relative to some datum plane.

However, it is worth noting at this point that, in most process problems, the combined effect of changes in the potential energy and the kinetic energy terms is often negligible.

Proper transfer terms are also incorporated in the energy balance equation. Thus, the variation of the total energy is reduced to an expression of the type:

$$\frac{d}{dt}\left\{U\sum_{i=1}^{nc} m_i\right\} = \sum_{in} e - \sum_{out} e \qquad (2.3)$$

where U is the specific total internal energy, and e refers to the rate of energy transferred in any form. As for the terms in the mass balance, additional equations may characterise each of these transfer terms. For instance, if the connection considers heat conduction between two interconnected phases the proper equation will characterise this term.

Indeed, the above formulation also covers the case of reacting systems.

Thus, the total energy balance for the phase A of the example given above may be written as:

$$\frac{d}{dt}\left\{U^{\wedge}\sum_{i=1}^{nc} m_i^{\wedge}\right\} = -e \qquad (2.4)$$

It is clear from above that the whole system is thus characterised by a set of differential-algebraic equations.

It is also evident that the application of the conservation principle to the mass of each chemical species i of a multicomponent mixture in a phase provides a set of nc differential equations while the principle of conservation of the total energy provides one differential equation.

One important issue which appears in modelling dynamic systems is related to the concept of the dynamic state of the system. The dynamic state of the system is defined as a non-unique set of variables whose values at some time, together with

the future inputs is sufficient to allow determination of the system's future behaviour.

It is now possible to tie the two concepts of state: the thermodynamic and the dynamic. Since we have assumed that the instantaneous thermodynamic state is stable, the determination of the evolution of only (nc + 2) independent variables is sufficient to characterise the state of any phase at any time.

Since (nc + 1) dynamic relations are obtained from applying the conservation laws, the evolution of (nc + 1) properties can be predicted. Therefore, there is one missing relation to characterise the complete dynamic evolution of the system. It seems that there exists no other general law which could provide this missing relation. However, in process systems it is sometimes possible to find a property which either remains constant or it is assumed as a constant through the complete operation of the system, or some relations between properties of the phases may be known. For instance, the previously introduced concept of vessel suggests the use of volume or the pressure as a known property. Thus, if the volume of the phase happens to be a constant, the volume should be the selected dynamic state variable, e.g. the volume of the compressed oxygen bottle in Figure 2.2.a is a constant. There are some other cases in which the volume changes but then the pressure is essentially a constant so that pressure becomes a better choice as a dynamic state variable, e.g. see the tank shown in Figure 2.1.a.

2.4.- The Concept of Region

Let us now consider a closed tank containing two phases, A and B, where heat transfer occurs between the phase B and the environment (reservoir). The application of the conservation laws result in the following equations:

$$\frac{d}{dt}\mathbf{m}^{\wedge} = \mathbf{f} \qquad (2.5.a)$$

$$\frac{d}{dt}\mathbf{m}^{B} = -\mathbf{f} \qquad (2.5.b)$$

$$\frac{d}{dt}\left\{U^{\wedge}\sum_{i=1}^{nc} m_i^{\wedge}\right\} = e \qquad (2.5.c)$$

$$\frac{d}{dt}\left\{U^{B}\sum_{i=1}^{nc}m_{i}^{B}\right\}=-e+Q \qquad (2.5.d)$$

where **f** refers to the nett flow which is transferred from one phase into the other, Q is the heat exchange with the reservoir, and e is the nett energy transferred between the two phases. Then, the variables **f** and e should be characterised by some algebraic relation. For the sake of simplicity, let us consider the phases to contain only two chemical species so that the mass transfer between the two phases can be related to the binary mass transfer coefficients as below:

$$N_1 = k^{A}(y_1^{A,I} - y_1^{A}) + y_1^{A}(N_1 + N_2) \qquad (2.6.a)$$

$$N_1 = k^{B}(y_1^{B} - y_1^{B,I}) + y_1^{B}(N_1 + N_2) \qquad (2.6.b)$$

where N_i is the molar flux of chemical species i, k^j is the mass transfer coefficient in phase j for a binary system, y_i^j is the mol fraction of chemical species i in phase j, I refers to the interface.

The total transport rate **f** is thus equal to the molar fluxes multiplied by the total interfacial area A and by the diagonal matrix of mol weights **M**:

$$\mathbf{f} = A\mathbf{M}\mathbf{N} \qquad (2.7)$$

The heat fluxes can be related to the heat transfer coefficients as follows:

$$E = h^{A}(T^{I} - T^{A}) + H_1^{A}N_1 + H_2^{A}N_2 \qquad (2.8.a)$$

$$E = h^{B}(T^{B} - T^{I}) + H_1^{B}N_1 + H_2^{B}N_2 \qquad (2.8.b)$$

where E is the energy flux, h is the heat transfer coefficient, T is the temperature, H is the molar partial enthalpy, and the superscript refers to conditions at the phase or the interface.

The total transport rate e is equal to the energy flux multiplied by the total interfacial area, i.e.

$$e = AE \tag{2.9}$$

To complete the model, we assume that the equilibrium prevails at the interface; thus, the following usual equations of phase equilibrium relate the mole fractions on each side of the interface $y_i^{A,I}$ and $y_i^{B,I}$:

$$K_1 = y_1^{A,I} / y_1^{B,I}; \quad K_2 = y_2^{A,I} / y_2^{B,I} \tag{2.10.a}$$

$$y_1^{A,I} + y_2^{A,I} = 1; \quad y_1^{B,I} + y_2^{B,I} = 1 \tag{2.10.b}$$

where K_i are the equilibrium ratios or "K-values". These equilibrium ratios are complicated functions of temperature, pressure, and composition: $K_i = K_i(T^I, P, y_i^{A,I}, y_i^{B,I}, i = 1,2)$.

However, in engineering practice the two phases considered may be assumed to coexist in physical equilibrium in which case the above formulation may have some difficulties. The numerical values of the transfer coefficients would become infinite although the nett flows **f** and e remain finite, and can no longer be obtained from (2.6) and (2.8). Thus, the molar fraction and the temperature at the interface are equivalent to the conditions at the referred phases, and the nett flows must be such as to maintain the equilibrium.

In fact, the above formulation is one case of the problem known as high index. The definition of high index can be found in Gear [1988], Brenan et al [1989], or Hairer et al [1989]).

However, we are not interested in the nett flows to maintain equilibrium, and if these are eliminated from the equations by writing conservation equations for the multiphase equilibrium system, the index problem is avoided (a similar conclusion is given in Ponton and Gawthrop, [1991]). The concept of "region" is introduced for this purpose.

A region is a particular subdivision of the system being modelled composed of single or multiple phases in thermodynamic equilibrium.

The state of a region containing multiple phases is also considered to be stable from the Gibbs point of view given above. Therefore, they are also characterised by two independently variable properties in addition to the masses of each chemical species of the system.

A difficulty arises, however, in the choice of the two independently variable properties required to specify the system. For instance, consider a closed tank containing a simple thermodynamic system composed of a pure component in liquid-vapour equilibrium. Then the pressure must be the vapour pressure of the component at the temperature in question. If the temperature and the total volume of the vessel are given, the system is completely specified because the volume and temperature are independent variables. However, if the pressure were specified instead of the total volume, the system could not be reproduced because given either pressure or temperature the other is automatically specified. A variation of the same problem is obtained if the internal energy and the temperature are fixed and the properties are evaluated assuming ideal behaviour.

Once the state of the region is evaluated, the equilibrium relations can be used to evaluate the rest of the characterising properties and the distribution of mass and energy within the phases which make up the region. Thus, the phases are still the elements containing the mass of the system. For instance, the mass and energy balance of the closed tank may be written as:

$$\frac{d}{dt}\mathbf{m}^r = 0 \qquad (2.11.a)$$

$$\frac{d}{dt}\left\{U^r \sum_{i=1}^{nc} \mathbf{m}_i^r\right\} = Q \qquad (2.11.b)$$

where $\mathbf{m}$ and U are referred to the region r.

Then, the conditions of equilibrium are extended to all the phases of the region within a flash procedure. Regions are considered as a means of taking advantage of the well understood problem of phase equilibria and their detection is not the responsibility of the user but must be automatically performed.

2.5.- Additional Elements for Modelling Process Systems

In some way, we can say that the basic elements for modelling process systems to achieve our intended goals have already been provided in the previous sections. However, there are some additional peculiarities in process systems which cannot be modelled unless we introduce additional elements. The purpose of this section is to introduce these required elements.

During the course of the modelling activity, the modeller very often starts by dividing the process being modelled into very general sections. Subsequently, he/she may proceed to decompose each of these sections into a number of smaller problems that are much simpler to handle. It results in a hierarchy in the description of the process similar to the hierarchy of design decisions proposed in Douglas [1988]. This procedure is recognised as a top-down description of the process whose importance has also been discussed in Marquardt [1991].

According to our modelling purposes, the top-down procedure helps the user to order his modelling problem. For instance, a chemical plant may be divided in three general sections: the pretreatment, the reaction, and the separation sections, see Figure 2.3.a. If we proceed to a further decomposition of the separation section we observe that it may contain a set of distillation columns which can be represented as in Figure 2.3.b. A disaggregation with more details can be obtained if we take a distillation column so that we have plates, condenser and reboiler, see Figure 2.3.c. Finally, the description of plates would be in terms of the phases contained within each plate.

Of course, the description order can be the opposite: two phases are contained in a plate which belongs to a distillation column within the separation process. This strategy is called the bottom-up description.

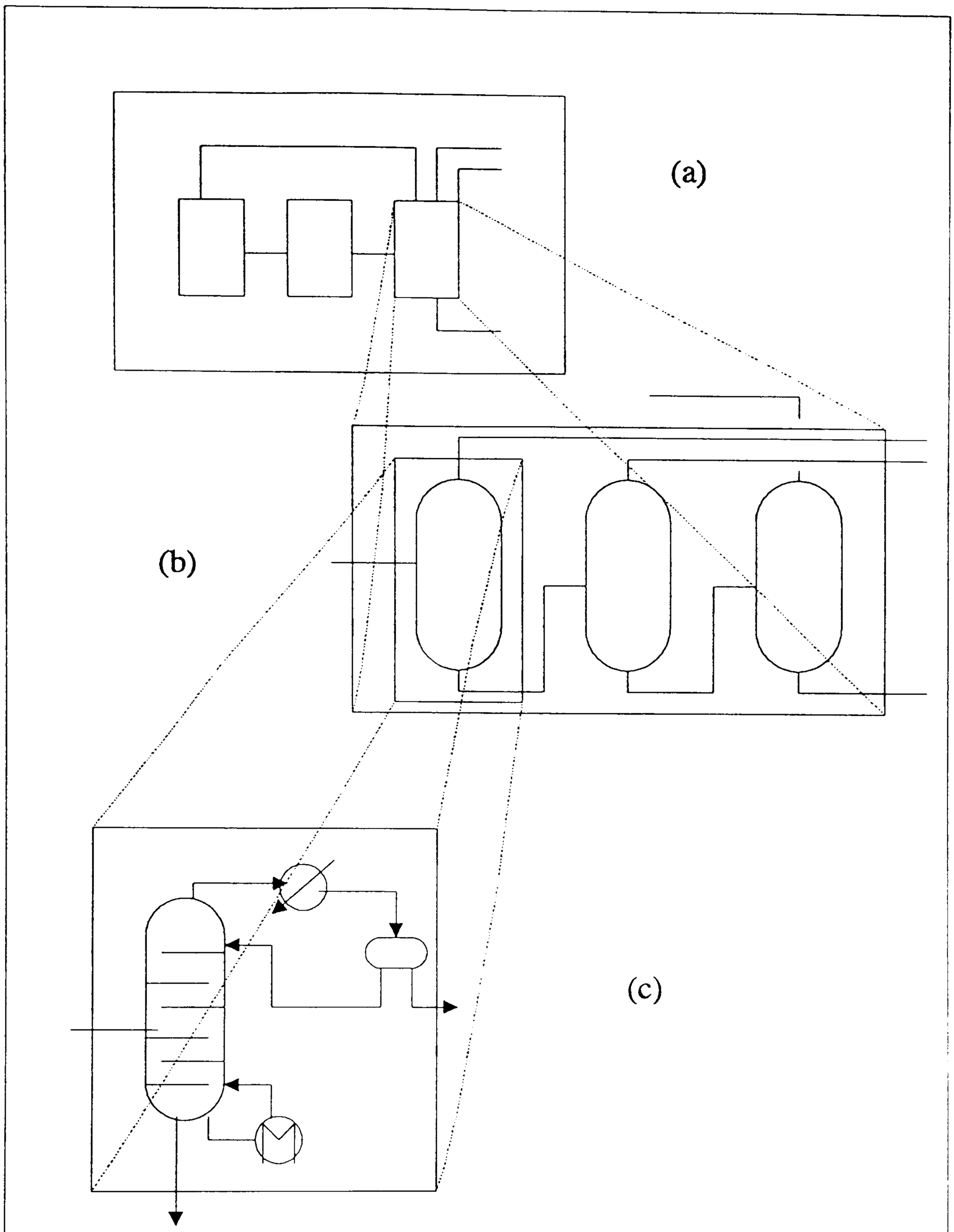


Figure 2.3 The Hierarchical View of Process Systems

Until now, the only way the user has to group phases is through a vessel. Unfortunately, the vessel introduces volume relations which are not always relevant. Therefore, it is evident that there is an element missing to allow the integration of the phases as pointed out above.

The concept of process is introduced as an element of the system description whose function is in essence to group phases which are related. It is clear that the concept of process is so general that a vessel is in fact a type of process. The plates, the distillation columns, the separation section, etc in the example given in Figure 2.3 are all different instances of process.

Let us now consider some characteristics that the user may want to associate with a group of phases. It seems normal that within a complex chemical plant two or more units have similar designs. Thus, two storage tanks possess the same total volume, two or more tanks are open to the atmosphere so that the pressure is the same and equal to the pressure of the environment, etc. Thus, it seemed sensible to include the possibility of describing this sort of feature. Since the phases are grouped via a process, the solution to this problem has been to associate this description with the declaration of a process.

Thus, two types of processes have been identified: purely topological and functional.

The purely topological type introduces no further equations in the mathematical model. In general, the purpose of this type of process is rather to identify a section of the system so that it provides the multilevel hierarchical representation of the system. A logic rule to allow the multilevel representation is established as follows: if the process "A" contains an element of the process "B", then either "A" is a subset of "B" or "B" is a subset of "A". In other words, a process must have only one parent. However, a process may have more than one child. For instance, the form given in Figure 2.4.a. is not allowed for purely topological processes but the forms given in Figures 2.4.b and 2.4.c are perfectly possible.

The functional type of process affects the mathematical model by introducing equations. They appear only at the level of the phases and any of the cases shown in Figure 2.4 are feasible.

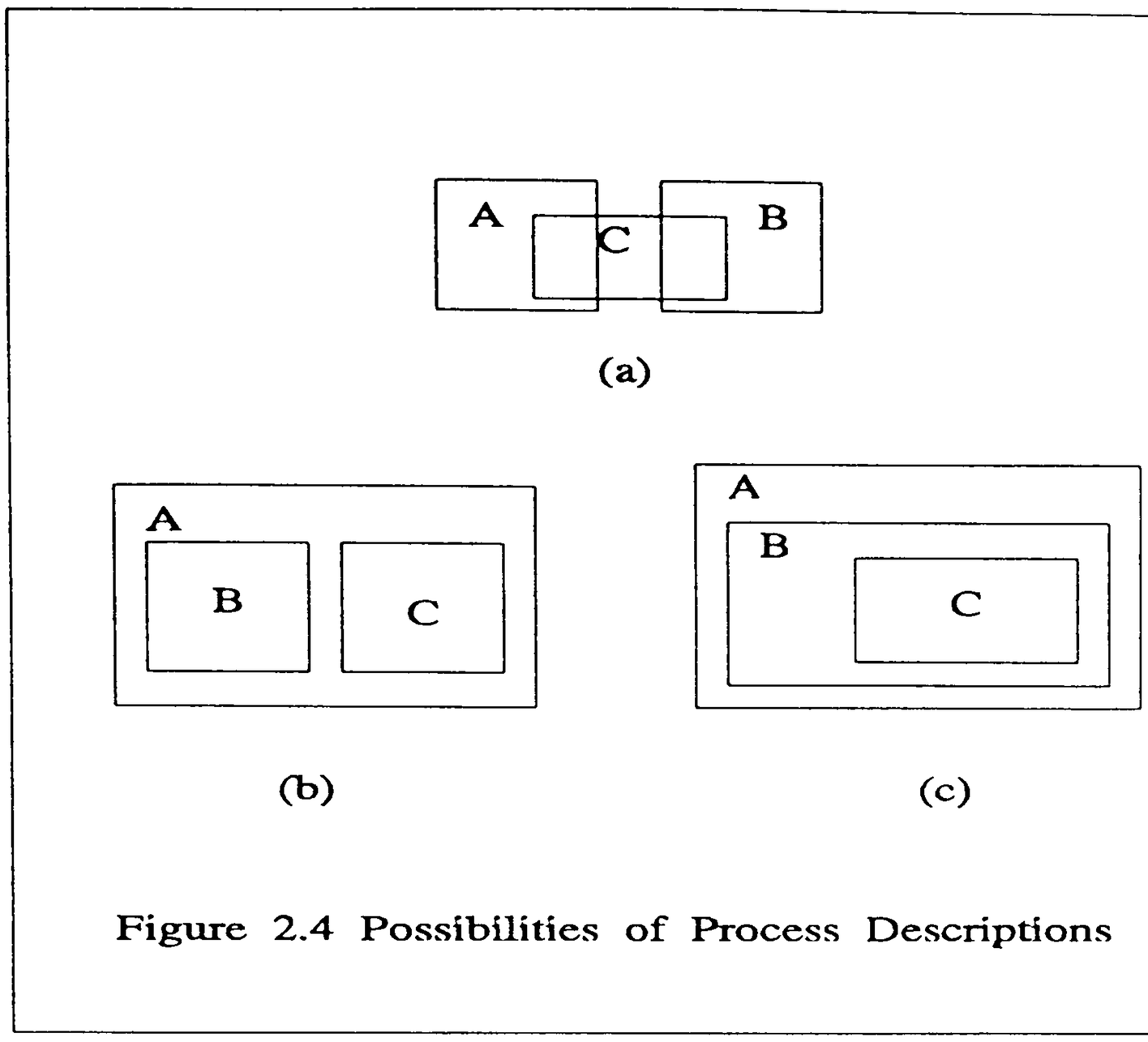


Figure 2.4 Possibilities of Process Descriptions

2.5.1.- Connecting Processes

It has been indicated before that phases are the ultimately interconnected entities of the system. One characteristic of process systems is that as a result of even a normal operation some of the phases may appear or disappear. For instance, suppose that eventually the liquid contained in the flash tank shown in Figure 2.1.b is completely removed. Since the connection at the bottom of the tank was defined to connect the liquid phase then, in the best case, the evaluation of the flow streaming out becomes nil. This result may not correspond to the practical value since it is more likely that the flow exists but the vapour is in fact the phase streaming out. Therefore, it seems sensible to expand the concept of connection in order to reproduce this sort of phenomenon.

The solution suggested was based on the analogy with the reality: the connection is directed to the vessel rather than the phase. Thus, connections are allowed to interconnect vessels though it is clear that still the ultimately interconnected entity is always a phase.

It was also observed that the connection to a vessel is required not only when one phase disappears but also in other cases. For instance, consider the case shown in Figure 2.5. The cylinder is clearly a vessel which contains two phases separated by an impermeable boundary. Let us now assume that the following operation proceeds: initially, mass is flowing into the cylinder in the point "A" and flowing out in the point "C", at the same time, mass is injected into the cylinder in "B" which may produce a movement of the piston until eventually the trajectory "A"-"C" is closed and mass starts flowing from "B" to "C". Then, it is observed that the connection "C" may connect either the phase over the piston or the phase below the piston. Thus, the solution given was to consider that the connection is on the vessel rather than the phases.

Extending the connection to a vessel gives the idea of having a sampling mechanism to select the phases streaming in or out. The reproduction of these systems with a mathematical model requires the inclusion of logic variables which may be defined by the user. For instance, the model of the cylinder may include the following sort of statements:

```
if Logic then
   flow= {phase over the piston}
else
   flow= {phase under the piston}
endif
```

where the variable Logic is perhaps declared by the user as Logic= {volume under the piston < a numerical value}.

Let us suppose now that a similar flash tank contains three phases: two liquid phases and one vapour. Suppose that none of the phases disappears. Then, the vapour streams out at the top of the vessel and the liquid flows out at the bottom. The question arises here about what liquid phase is streaming out. A logic analysis indicates that the heaviest phase may flow out. However, it may happen that the two liquid phases are in fact homogeneously distributed so that the actual stream is a mixture of both phases.

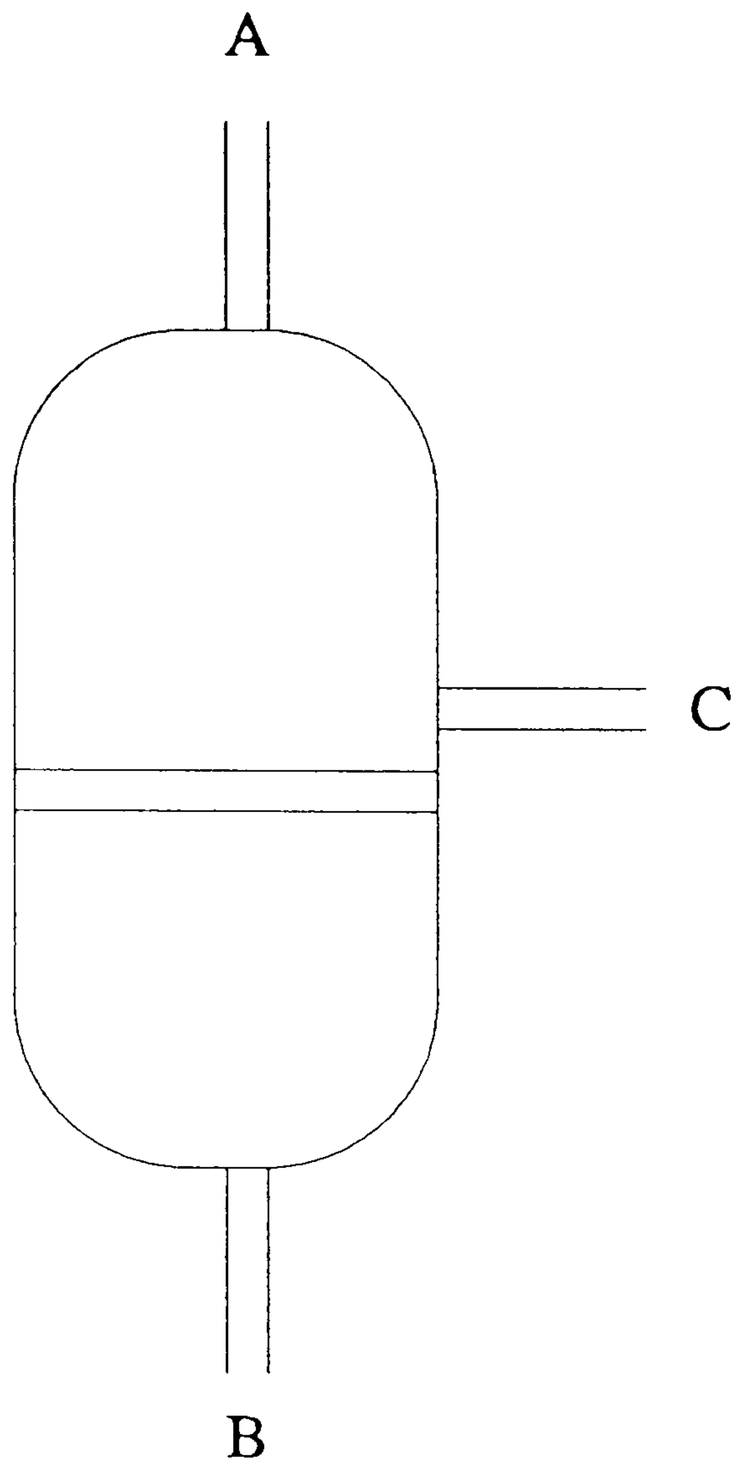


Figure 2.5 A Cylinder Fitted with a Piston

Thus, to overcome the modelling problem we have considered the possibility of grouping these phases via a process and then the connection is defined to interconnect the process rather than any of the involved phases.

If we consider that a vessel is a sort of process, then it is possible to say that a system is composed of a set of interconnected phases and processes.

In fact, the global description of a process system is often simplified by using a flowsheet which, in a global view, indicates a set of interconnected sections having hidden or incomplete information in order to make the process map more readable and easy to understand. The subsequent descriptions provide the actual interconnected processes and/or the actual interconnected phases.

2.5.2.- The Case of Reactive Systems

There are many ways of classifying chemical reactions. The scheme used here is the breakdown according to the number of phases involved so that they can be homogeneous or heterogeneous.

In homogeneous reactions, all reacting materials are found within a single phase, be it gas, liquid, or solid. If the reaction is catalytic, the catalyst must also be present within the same phase if the reaction is homogeneous. The catalyst is considered as a compound which enables a reaction to proceed at a different rate by interacting with some reactants to form intermediate compounds which, in turn, interact with more reactants to form a desired product and regenerate the catalyst.

A heterogeneous reaction requires the presence of at least two phases to proceed. Then, the reaction is carried out in the interface of the two involved phases. As a result, the phases involved in the reaction must be interconnected. Very often, the catalyst is present in a phase different from those of the reactants.

The rates of physical processes like mass and energy transfer can be adequately estimated in many cases from the properties of the substances participating in the reactions, flow patterns, the geometry of the vessel, etc. However, chemical rate data for most industrially important reactions rely upon experimental investigation of the specific chemical reactions involved. In many cases a reaction mechanism

is determined by trial and error by postulating that the over-all reaction takes place via two or more elementary reaction steps. The reaction steps may proceed reversibly, concurrently, and/or consecutively.

Most reactions important in industrial processes are quite complex to model. In any case, however, the description of a reaction is based on the stoichiometric representation and a name associated with the model. Various models can be found for instance in [Perry et al, 1974]. The number of models considered in this thesis are far from the total existing. Because of the importance of identifying what reactions occur in the system, it was considered sensible to incorporate each reaction as another class of element of the process description.

2.5.3.- The Case of Controllers

Process systems may be controlled in order to give more uniform and higher quality products by the application of automatic control, and this often leads to higher profits. Automatic control is also beneficial in certain remote, hazardous, or routine operations.

The control of process systems is carried out by a controller. In general, a controller is multivariable and involves three types of elements: sensor, control law, and final control element. The sensor detects the measurement of one or more properties in the phases of the system, the control law relates the measurement to control signals, and the signal modifies the condition of the final control element, commonly a process-control valve, so that the flow associated with a connection is modified.

Industrial automatic process controllers vary from simple on-off devices to special purpose computing instruments. They are used to couple the controlled process variables to the manipulated variables. Recently, control by computers has been used to improve the performance of the unit operation. With computer control systems, it is possible to calculate, specify, and control performance variables which previously could be neither measured nor controlled.

Simulation of control systems has become a valuable tool for control systems analysis. In simulation, a mathematical model responds to disturbances and adjustments or modifications in the same way as the real process. Then, the

responses may be observed as automatically plotted process or control variables or computer printouts rather than in a plant environment.

Thus, the generation of control models requires the inclusion of the control equations, and a controller is modelled here as a black box which, given the inputs, produces some outputs. The inputs are in fact properties of the phases and the outputs correspond to some variables modelling the connections.

2.6.- Summary

The various sort of elements required for modelling process systems have been introduced in this chapter. However, only six of them are the responsibility of the user to provide a clear definition. These six elements are:

- Phases
- Processes
- Connections
- Reservoirs
- Reactions
- Controllers

Regions are internally and automatically manipulated.

Three specific areas were identified for further work:

- A definition language for the user to describe the elements of the process system being modelled.

- The procedure to automate the model generation.

- The computer implementation.

In the following three chapters, each of these three areas will be discussed in detail. Procedures which meet the goals set out in chapter one will be developed and described.

CHAPTER THREE

A DEFINITION LANGUAGE FOR PROCESS SYSTEMS

"Proper words in proper places make the true definition of style"

Swift

Having described the general modelling approach, the immediate problem which arises here is the necessity of defining a language general enough to be able to describe the elements of the representation which are the user's responsibility. A proper language is an indispensable precondition for modelling systems with a computer program.

This chapter is devoted to both the formal definition of the elements of the representation and the definition of a language to describe the characteristics of the system. Because of the new developments in computing, the possibility of a graphical representation has also been suggested in parallel to the language definition.

3.1.- Introduction

The language may be considered as the instrument for conveying ideas from the user to a computer code. Then, the job of the computing program is to apprehend the meaning readily and precisely. Various languages for modelling process systems have been reported [Piela, 1989][Stephanopoulos et al, 1990]. In fact, every flowsheeting package possesses its own modelling language in order to translate the information about the system being modelled. Unfortunately, the existing languages do not support the intended description.

The general objective of a language definition is to provide a convenient way of describing the original system to interface with the computer program. There are two conceptually different approaches to symbolically describe a process. These approaches are named here as declarative and graphic techniques.

In the declarative approach, the knowledge of the process is described by using sets of words which are normally included as a part of a common language. Normal languages like English, Spanish, French, etc. may be used for this purpose though sometimes the terminology used is rather program-specific. For instance, an engineer may report a problem in the unit TD-400 when the difficulty exists in a distillation column. One of the goals in the declarative approach is to use normal words in such a way that no further explanation is necessary and the information is documented by itself. The declarative approach is the most used technique to represent a process in current flowsheeting packages.

In the graphic approach, the knowledge is described by geometric symbols rather than words. Then, a meaning is associated with each symbol. Simplicity is essentially desirable in this approach. A block diagram, being the simplest form of representation, should be used to represent parts of the process. For example, block diagrams have been useful for representing processes in a simplified form in reports and textbooks. The symbols in a block diagram can be of any shape, but it is usually convenient to use a mixture of squares and circles drawn with a template. A current practice is that every design office uses its own standard symbols, though some symbols have been formally presented in the literature in order to make their use uniform. Some sets of symbols, for use in flowsheets, have been published in the British Standard, BS 1553 [1977], and by the American National Standards Institute (ANSI).

Unfortunately, the utilisation of only a graphic representation may prove to be too difficult, expensive, or impractical. In addition, the implementation depends on the available software. As pointed out in Piela [1989], the progress in process simulation is partially dependent on developments in computer technology. However, it seems that nowadays it is possible and relatively easy to incorporate visual features into computer interfaces. In this thesis, the graphic technique is used for representing the topology of the system being modelled; then, this description may be combined with the declarative approach. Thus, both techniques, declarative and graphic, are used to describe a process without losing the primary motivation of keeping the separation between the problem description and the technical solution.

3.2.- Overview of the Proposed Modelling Language

After examining some of the capabilities of new programming languages and having acquired an understanding of the modelling problem itself, a new language to describe and represent any process system has been conceived in this work. The proposed language allows a modeller to describe a process in terms of more elementary operations, physical mechanisms, and properties of the entities describing the system. It also gives modellers a formalism by which they can organise their model-building process. In addition, it can be used to provide advice to other modellers by facilitating the extraction of knowledge.

Our guiding principles are to use the simplest language and to take advantage of the currently used technical terms. A good language can be defined simply as a language which is readily understood by the user. To be clear is to be efficient; to be obscure is to be inefficient.

The modelling language should not try to give all the details of the law relevant to the subject, but to be content with stating the essentials to explain. The first requisite is to ensure that the users will know just what meaning the words convey. The compromise is to choose the right words and the right form in order to make the meaning clear. We will try to use compact words with precise meaning.

The above requirements can be summarised as follows: the language definition must be clear, comprehensive, accurate and complete to achieve its intended application.

Thus, the characteristics that the user wants to incorporate into a mathematical model are declared based on the following general form:

$$name = \{declaration\} \tag{3.1}$$

We begin always by providing a name which is used as a reference to facilitate the understanding of the computer's reply, perhaps the mathematical model itself. Then, a set of specific words naming the properties or attributes will provide the actual declaration of the entity.

Naming properties is considered an important step in describing the system. Attribute names should consist of words that describe the attribute and make it unique. Prepositions, connections, and meaningless words should not be used since they make the attribute name longer than necessary and add nothing to the clarity of the name. For instance, the use of the popular "In Language" or "Of Language", e.g. PressureInLiquid, is avoided.

3.3.- Description of Phases

Phases are the simplest cases of thermodynamic subsystems in which a lumped process system has been subdivided. Indeed the word phase has been adopted as a part of the common language and used for a long time in chemical engineering. Perhaps the oldest reference where a phase is well defined was given by Gibbs in 1878 in his paper: "On the equilibrium of heterogeneous substances." Gibbs' assertion is expressed in words: "In considering the different homogeneous bodies which can be formed out of any set of component substances, it is convenient to have a term which shall refer solely to the composition and thermodynamic state of any such body without regard to its size or form. The word phase has been chosen for this purpose."

Phases, as elements of the model-building process, are described by a statement of the form:

$$p = \{\alpha\} \qquad (3.2)$$

where p is the name of a given phase and α is a set of properties of p that the user may want to incorporate into the model. A comma is used to separate each element of α.

Some symbols and abbreviations are unique, descriptive, meaningful, and represent the smallest unit of information; they result in substantial length reductions. Based on this consideration, it was decided to represent the properties with symbols which are currently used. Table 3.1 gives a list of the properties and their respective symbols used in this work. Observe that the possibility of assigning a numerical value to the properties has been also considered.

Property	Symbols
Pressure	P P= #
Temperature	T T= #
Volume	V V= #
Entropy *	S S= #
Enthalpy *	H H= #
Thermal conductivity	k k= #
Density *	d d= #
Viscosity	v v= #

is a given numeric value
* specific properties

Table 3.1 Valid Properties of a Phase

For the sake of simplicity and clarity the phase name is also omitted from the properties name, but it is implied by the name of the set containing the attributes.

Some examples of phase representations by using the declarative approach are:

Example 3.1 *Phase Declarations:* Suppose that in a phase named "Phase1" the properties of interest are pressure, temperature, and volume; then, the phase may be declared as below:

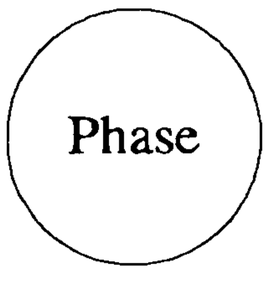
$$Phase1 = \{P, T, V\}$$

A variation of the above example is obtained if the volume occupied by the phase is known. Then, it should be declared as:

$$Phase1 = \{P, T, V=100.\}$$

Care must be taken when assigning a numerical value since the number of variables the user can fix depends on the degrees of freedom which is related to the state of the system.

Using a graphic representation, a phase will be topologically described by a circle as below:



3.4.- Description of Vessels and Processes

It was considered in chapter two that a vessel is in fact a sort of process. Therefore, the language to declare a process is also used to declare a vessel.

Processes are elements of the process model-building described by statements of the form:

$$s= \{P; \beta\} \qquad (3.3)$$

where s represents a name for the process, and β is a set of attributes of a non-void set P of phases and/or processes. A semicolon is used to separate the set of entities from the set of attributes and a comma is used to separate entities and attributes within the respective set.

There are two sorts of attributes which can be declared through processes. The first kind specifies that the phases in the set P possess the same numerical value of the property indicated. The second kind specifies that while the values of the indicated property may change in each phase, the sum of these values may be equal to a known value. This second group is, in practice, very much reduced to the volume occupied by the phases of a vessel. It is believed that the numerical value of the total volume of a vessel may be known so that the user should be allowed to incorporate this value.

A detailed list of the symbols used in this work for describing structures is given in the Table 3.2. Observe that the string 'sum' has been added to the property's symbol to represent the second kind of properties indicated above.

A vessel may be identified by the inclusion of the attributes sumV and P.

If a set of phases are defined to be in equilibrium, the user must declare these as a process, and connections may only be made to this process (not to the individual phases). The user must also take care to declare all possible phases likely to coexist. See Section 4.8 for further explanation.

Some examples for process representation following the previous convention using the declarative approach are given below:

Example 3.2 *Process Declarations:* Consider two phases named "Ph1" and "Ph2" where the user wants to impose the condition that the enthalpy is the same in both phases. Then the process may be declared as:

$$Pr1= \{Ph1, Ph2; H\}$$

Property	β Attribute
Pressure	P
Temperature	T
Volume	V sumV sumV=#
Entropy	S
Enthalpy	H

is a given numeric value

Table 3.2 Valid operations for processes

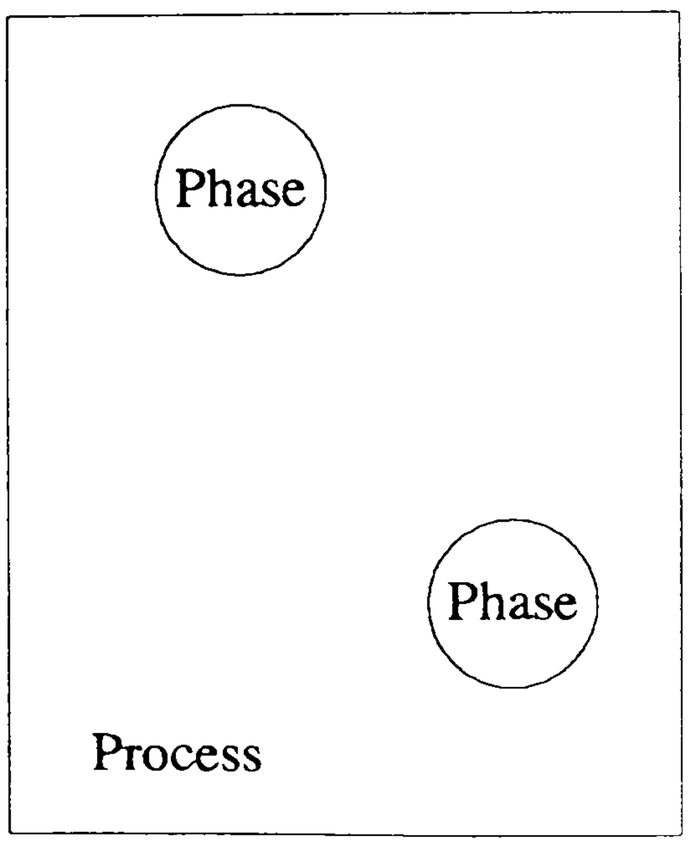
Two phases "V1" and "L1" contained in the same tank (a vessel) are declared as below:

Pr2= {V1, L1; P, sumV}

A variation of the above declaration is given if the total volume is known. Then it is declared as follows,

Pr2= {V1, L1; P, sumV=100}

Using the graphic approach, any process is represented in this work by a rectangle which contains the affected phases, i.e.



3.5.- Description of Reservoirs

A reservoir is defined as below:

> "A Reservoir is an infinite source or sink such that its state and properties are independent of transfers between it and any other portion of the system"

There will usually exist more than one reservoir in the process system being modelled. In addition, any reservoir can be interconnected with more than one phase or process but never with another reservoir. For instance, more than one tank or other process unit could be open to the atmosphere, heat exchangers could use the same cooling fluid, etc.

Reservoirs are elements in the process representation of the form:

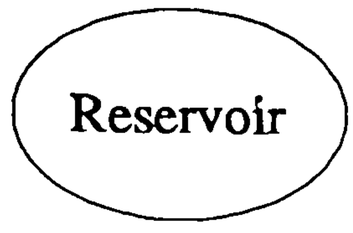
$$R = \{\} \qquad (3.4)$$

where R stands for the name of a reservoir. Note that it has the same form as a phase declaration but without any property declaration. Explicit declaration of the properties in a reservoir is not necessary because every required property will be automatically detected. In fact, the form (3.4) can be reduced to the simple declaration of the name of the reservoir. In principle, the values of properties of reservoirs should be provided by the users as known parameters when the model is solved.

Example 3.3 *Reservoirs:* Some examples of declarative representation of reservoirs are:

Reserv1= { },
Reserv2.

Reservoirs are graphically represented by an ellipse, i.e.



3.6.- Description of Connections

It has been established before that a process system is conceived as a set of interconnected phases, processes, and reservoirs. In this section, the concept of connection and its parts are defined and analysed. In general, connections are used to describe the transfer of mass and/or energy through the boundary of any phase or process.

Connections are formally defined as below:

> "Connections are elements of the process representation with no hold-up, through which mass and/or energy is transferred between two phases or processes"

It was observed that any connection should be characterised by the following three pieces of information:

- The two elements, phases or processes, that are connected.
- The sampling mechanism in each connected element.
- The transfer rate model required to determine the flow.

Thus, a connection can be described by a statement of the form:

$$C = \{X; \chi; \delta\} \tag{3.5}$$

where X is a set of two elements and each element can be either a phase or a process, χ contains information about sampling mechanisms in the connection and δ contains information about the transfer rate model. The semicolon is used to separate the two connected entities, the sampling mechanisms, and the name of the model.

For modelling purposes, we adopt the convention that the flowrate is an output for the first entity which appears in X and it is an input for the second entity, e.g. the connection C1={*A*, *B*; ....} is an output for *A* and an input for *B*. Thus, the variables representing the transfer rate will have positive values if the sense of flow is in the expected sense; otherwise, the flow has the opposite sense. Thus, reverse flow will be manifested by negative values of the transfer rate variables.

In the following subsections the concepts of sampling mechanisms and the transfer rate model are discussed. Then, the declarative and graphic representations of a connection are given in detail.

3.6.1.- Sampling Mechanisms

The sampling mechanism represents a sort of policy to select the components that are removed from or added to any phase or process. However, it was observed that the sampling mechanism can be evident from the transfer law when the entity connected is a phase; thus, the sampling mechanism is not required when connecting phases. Hence, the importance of the sampling mechanism is only remarked when connecting processes.

After researching on the multiple possibilities of sampling mechanisms, we have distinguished three types, two of them are exclusive for mass transfer and the other is exclusive for energy transfer. These sampling mechanisms are: simple sampling, phase selective, and energy sampling.

We have only considered a subset of all the connections possible in a general process system. Specifically, only input connections to phases or regions have been developed, and output connections to phases or processes.

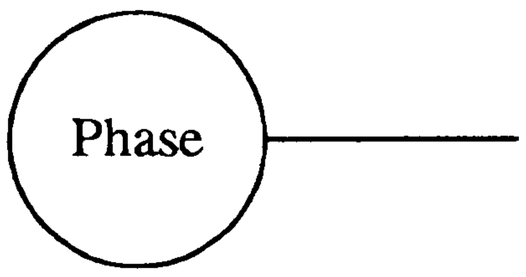
To give an example of the possible utility of more general connections, consider a distillation column. We might wish to determine the optimal feed plate location as

part of the model. In that case, the connection to the column is not made to a prespecified phase or region but only to the process representing the column. The implications of supporting this type of connection have not been worked out, and our prototype implementation does not allow this type of connection.

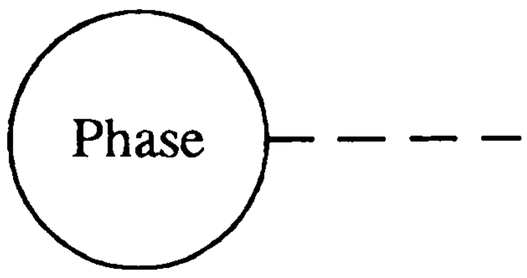
It is convenient to keep the graphical distinction between mass and energy transfer through a connection. In general, connections will be graphically represented by a line. In order to differentiate between simultaneous mass and energy transfer and only energy transfer the former is represented by a continuous solid line and the latter by a dashed line; i.e.

Mass and Energy Transfer Energy Transfer

Thus, a mass transfer connection to a phase is graphically represented by joining the line, which represents a connection, to the boundary of the circle, which represents the phase; i.e.



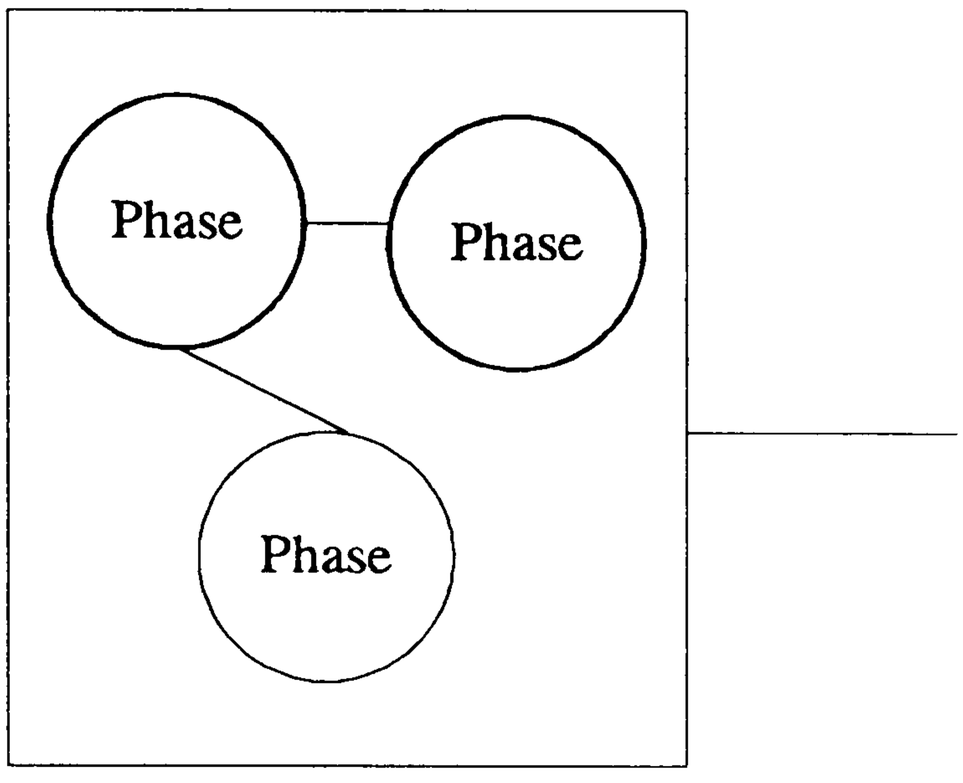
An energy transfer connection to a phase is graphically represented by using a dashed line connecting the phase; i.e.



A process can have two kinds of sampling mechanisms: simple and phase selective. The simple sampling is considered if the phases are physically distributed in such a way that they cannot be selectively extracted. The

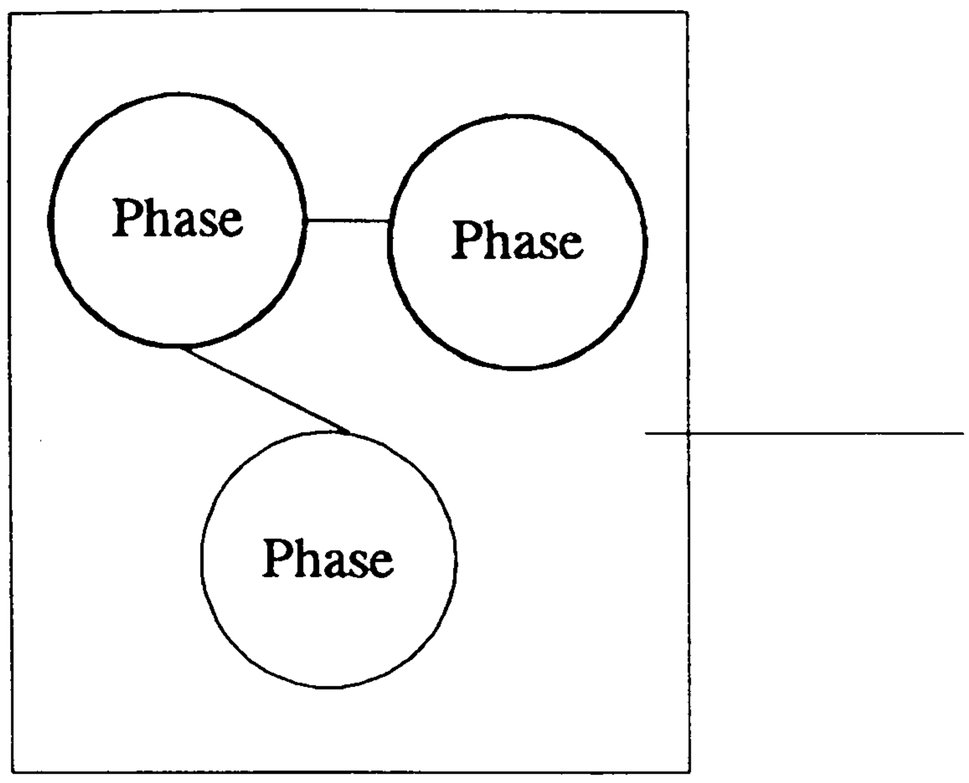
compositions of the flow in a simple sampling mechanism correspond to a "mean" value. This mean value is evaluated considering that the grouped phases are homogeneously distributed.

A simple sampling connection of a process is graphically represented by joining the solid line representing the connection to the edge of the rectangle representing the structure. The phases grouped by the process must be interconnected to allow a connection with this type of sampling.



When a process has phase selective sampling, the properties of the mass flowing out are equivalent to the properties in the selected phase. If this phase eventually disappears, then the mass flowing out will correspond to another of the phases contained in that process according to a priority order provided by the user.

The graphic representation of a phase selective sampling mechanism in a process connection consists of drawing a line which goes inside the rectangle where the phases to select must be contained, i.e.



For the sake of simplicity, one of the sampling mechanisms can be defaulted so that the user only has to declare the existence of the non-defaulted mechanism. In this work, the simple sampling mechanism will be defaulted so that only phase selective sampling will be explicitly declared. Thus, the explicit declaration of sampling is reduced to indicate any of the following three possibilities: "-phaseSelective", "phaseSelective-", or "phaseSelective-phaseSelective". The first case indicates that the second entity possesses phase selective sampling, the second case indicates that the first entity possesses phase selective, and the third case indicates that both entities possess phase selective sampling.

It can be observed that the language above does not provide any possibility to specify the priority order of the phases connected in order to automatically define the logic variable indicated in section 2.5.1. Apparently, it is not possible to provide a generalised form for this variable. For instance, a process which groups three phases *A*, *B*, and *C* may be physically connected at the bottom of the unit represented by the process; then, an order of the phase streaming out can be provided so that the model could be written as follows:

if mass of $A > 0$ **then**
    flow= {phase *A*}
else if mass of $B > 0$ **then**
    flow= {phase *B*}
else
    flow= {phase *C*}
endif

However, if the connection is at an intermediate level the order cannot be provided since the phase streaming out depends on the volume occupied by the other phase. More generally, the phase streaming out depends on the geometry of the physical unit being modelled. It was decided not to incorporate in this study any particular geometry so that the model leaves the logic variables as a part of the set of free variables.

3.6.2.- Transfer Rate Models

The last piece of information that a connection conveys is the transfer rate model. It refers to the particular equations which will model the flow through the connection. These equations will use the properties of the interconnected phases to evaluate the involved flow of mass or energy. They can be created either from a library of equations or using a mechanism to build up the equations. Some of the possible transfer mechanisms for both mass and energy transfer are defined below.

Mass can be transported from one phase into another by either of the following forms: bulk flow or molecular diffusion. The most commonly employed method of transporting fluids in a bulk form is forcing the fluid to flow through a piping system. Fluids can be moved through a pipe either due to the difference in the pressure between the two connected phases or by use of a pump or fan. In this study we will include four types of models for flow through a pipe: turbulent flow, laminar flow, pressure drop driven, and bulk flow. In the turbulent flow and laminar flow options we assume that the flow is unidirectional. The pressure drop driven model should be used if reverse flow is expected to occur; then, both models turbulent and laminar are wisely combined to produce a more general formulation (these models are discussed in chapter four). Finally, bulk flow should be used if there exists no model in the library to evaluate the transfer rate but it is known that bulk flow occurs in the transfer.

Principles of physics must be applied in order to derive the mathematical expressions which involve the pressure drop between two phases and the flowrate. However, most fluid-flow problems encountered in engineering involve streams that are influenced by so many factors that applicability of equations depends very much on each particular case. Only a few problems in fluid mechanics can be

entirely solved by mathematical means; all other problems require methods of solution which rest, at least in part, on experimentally determined coefficients.

The equations used in this work are not a complete set of the many possibilities; but certainly are representative and good enough to demonstrate the ideas given in this approach. The theoretical analysis is presented in the following chapter and here we simply identify the models.

The names for mass transfer connections correspond to the following types:

- Bulk Flow
- Pressure Drop Driven
- Turbulent Flow
- Laminar Flow
- Equilibration
- Two Film Model

The sorts of energy transfer connections are as follows:

- Specified Energy
- Heat Convection
- Heat Conduction
- Heat Radiation

Let us now give some examples of connections.

Example 3.4 *Connections:* In the following declaration of connections those entities which correspond to Phases start with "P" and those which correspond to processes start with Z.

C1= {P1,P2; ; TurbulentFlow}

C2= {P1,Z1; -phaseSelective; LaminarFlow}

C3= {P1, P3; ; SpecifiedEnergy}

C4= {Z1,Z2; phaseSelective-phaseSelective; LaminarFlow}

3.7.- The Language for Reactions

In general, the language to describe chemical reactions has been designed to follow the conventional stoichiometric formulation used in textbooks where, in order to keep track of chemical reactions, they make use of an accounting system which is expressed symbolically in the form of an equation. A generalised representation of a reaction is, for instance, as follows:

$$aA_{(g)} + bB_{(l)} = cC_{(g)} + dD_{(g)} \qquad (3.5)$$

By analogy, a general form to represent a reaction can be written as,

$$\rho 1[r1](\pi 1)+\rho 2[r2](\pi 2)+....= \sigma 1[p1](s1)+\sigma 2[p2](s2) \qquad (3.6)$$

where ρi is the stoichiometric coefficient of reactant ri in phase πi, and σi is the stoichiometric coefficient of product pi in phase si. The symbol of each chemical species, active in the reaction, appears multiplied by its stoichiometric coefficient.

Following the above form, the reaction in which methane burns with oxygen to carbon dioxide and steam is described by the equation:

$$[CH4]\ (G)+\ 2[O2](G)\ =\ [CO2](G)\ +\ 2[H2O](G) \qquad (3.7)$$

Though homogeneous reactions could be declared in the phase where the reactions are carried out and heterogeneous reactions could be declared in a process, it was felt better to declare either reaction as a separate entity. A reaction is declared according to the following form:

$$name=\{reaction;\ type\} \qquad (3.8)$$

where *name* refers to a name that the user may associate to identify the reaction, *reaction* is the actual stoichiometric equation as indicated in (3.6), and *type*

identifies a model to reproduce the reaction. The *type* also identifies if the reaction is reversible or irreversible. The semicolon separates the reaction from the type of reaction.

The reaction models will be discussed in chapter four so that we limit this section to listing their names. Among these names we have the following:

- Equilibrium
- IrreversibleMassAction
- ReversibleMassAction
- Empirical Order

For the particular case of empirical order of reaction, it is necessary to indicate the order and the chemical species referred to in the model. Perhaps it is better to give some examples to illustrate the language for reactions.

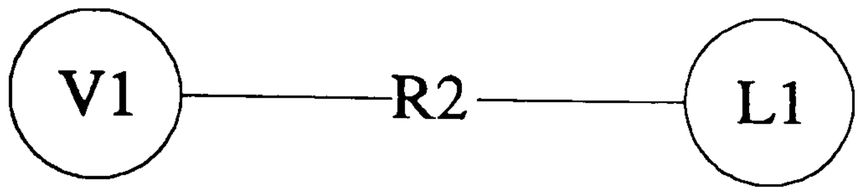
Example 3.5 *Homogeneous Reaction:* Suppose that the reaction A + B = C + 3D is carried out in phase "L1" following the reversible mass action law, then we would declare:

R1= { [A](L1)+[B](L1)=[C](L1)+3[D](L1); ReversibleMassAction}

Example 3.6 *Heterogeneous Reaction:*Suppose that the reaction A(g) + B(l) = C(g) + 3D(g) is carried out following a second order on A(g), then we may declare:

R2= {[A](V1)+[B](L1)=[C](V1)+3[D](V1); order_2_on_[A](V1)}

The graphical representation will automatically indicate interconnection of the phases involved in a heterogeneous reaction. For instance, the reaction declared in the example 3.6 is represented as follows:



3.8.- The Language for Control Systems

In this work, controllers are modelled as a black box where, given inputs, some outputs are obtained. Therefore, we limit the language for control systems to declare the inputs and outputs of the controller. Then, this information is internally manipulated to indicate a procedure which has to be defined by the user.

Thus, a controller is declared by an statement of the form:

$$C= \{input: l; output: c\} \qquad (3.9)$$

where l is a list of property variables of phases and c is a set of connections where the flowrates are controlled. The semicolon separates the inputs from the outputs. The list l has the form:

property1 in p1, property2 in p2, ...

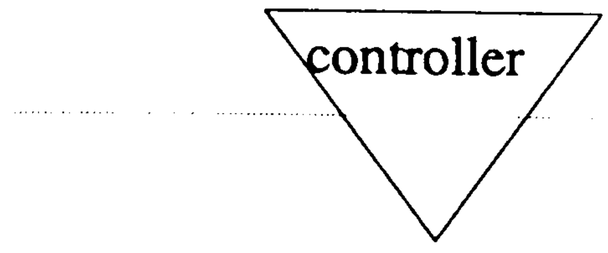
where the property1,property2, .. are symbols as indicated in the table 3.1 and p1, p2, .. are the referred phases.

The following example illustrates the language used to declare controllers.

Example 3.7 *Control Systems:*

C1= {input: P in P1; output: C2}
C3= {input: T in P1; output: C4}
C5= {input: P in P2, T in P3; output: C8}.

A graphical representation of a controller is indicated by a triangle. The phases involved in the input variables and the connections involved in the outputs are graphically indicated with a dot line, i.e.



3.9.- Summary

In this chapter, the design of a general language for modelling process systems has been developed to declare the elements of the process representation. The language is felt to possess the ability to characterise process systems.

The table 3.3 recapitulates the salient parts of the declarative and the graphic representation used in this work. In addition, a graphical representation of the interconnectivity of the modelling elements is given in Figure 3.1.

The language proposed above results in a clear evolution in modelling from "unit operations" to "elementary physical and chemical phenomena". The major benefit of the language is that the user will employ terms which lie closer to the normal terminology customary in the discussion of these specific topics.

Parts	Form of the Declarative Representation	Form of the Graphic Representation
Phases	p= { α }	Phase
Processes and Vessels	s= { X; β}	Process
Reservoirs	r= { }	Reservoir
Connections	c= {X; χ; δ}	——— - - - - - - - - -
Reaction	R= {[..]+..=[..] ; n }	——— *
Controller	C= {input: ξ; output: ψ}	controller

* only for the heterogeneous case

Table 3.3 Elements of the Process Representation

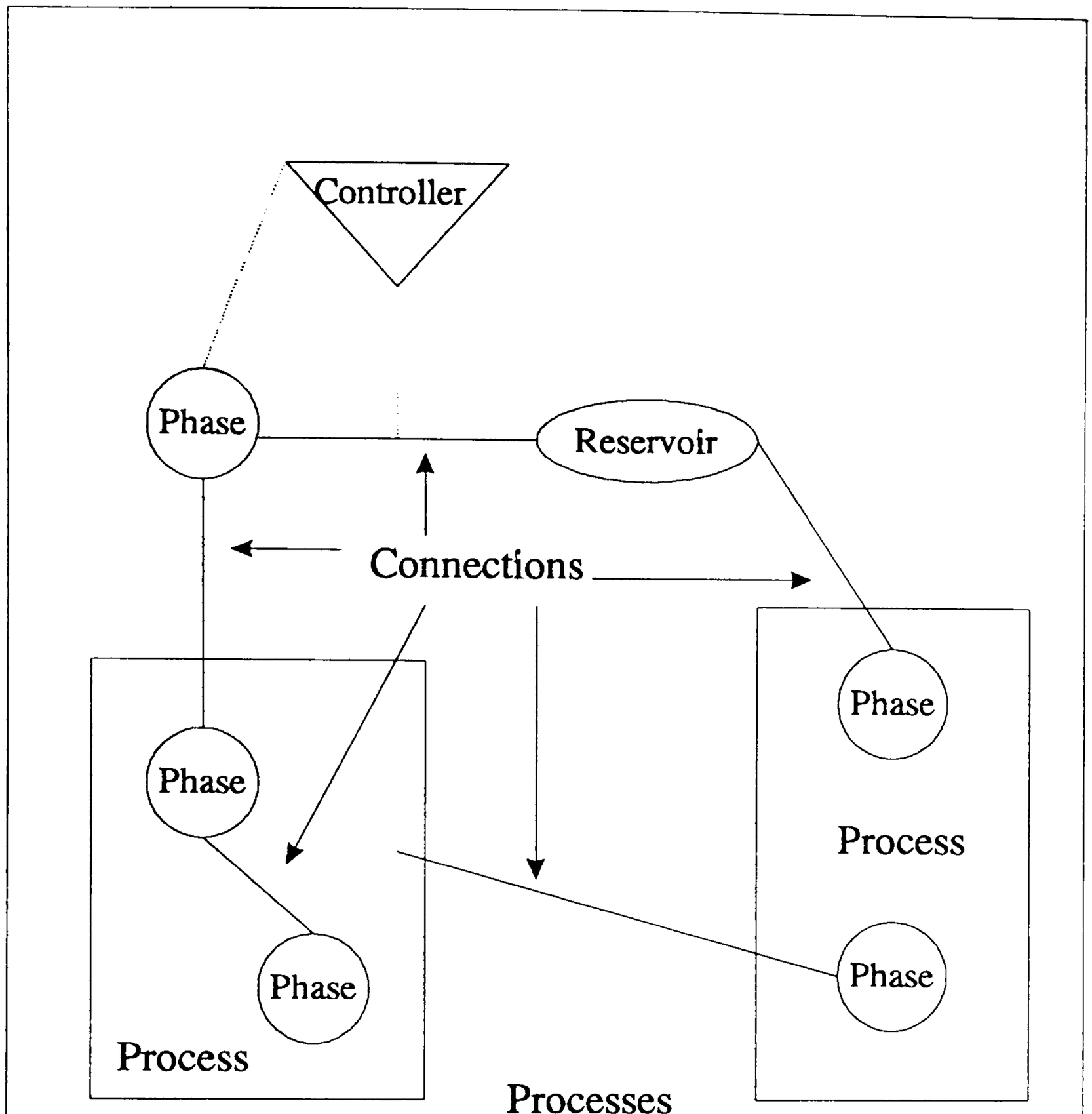


Figure 3.1 Interconnectivity of The System Representation

CHAPTER FOUR

THE MATHEMATICAL MODEL BUILD-UP

"Mathematics is a matter of personal satisfaction"

In the last chapter we have provided the language required to define the elements of the process system whose description is in fact the user's responsibility. Considering that the user already knows this language, this chapter is devoted to provide the procedures to build up the equations and to identify the transfer rate equations.

4.1.- Preliminaries

There are some formal aspects of modelling which it is also necessary to analyse. We begin by pointing out that every equation must possess proper dimensionality. The relationship between the variables expressed by a physical law cannot depend on the particular units used for the variables. In the automatic generation of the mathematical model we deal with the verification for dimensionality before incorporating the equation in the library or in the procedure to generate the model. Thus, the generated model will always be dimensionally consistent.

In general, an automatic procedure for model-building should generate a non-redundant set of equations and prevent, as far as possible, poorly formulated models. A model is considered "poorly formulated" if it may produce difficulties in the solution and require reformulation prior to a solution being obtainable.

Some of the problems are not possible to detect at the early stages of building general models. For instance, inconsistencies in the model are manifested by structural singularities of the system which may be as a result of poor selection of the free variables. The problem of high index may also appear as a result of a bad selection of these variables. In our case, the models are normally written to cover

both simulation and design cases. Hence the impossibility to even identify the index of the system. These problems are beyond our goals for modelling in this research.

However, there are some cases of poorly formulated models which attract our attention since they can be partially or totally detected and prevented in the step of modelling. Among these problems we have the cases of undefined quantities and inconsistencies of the model. In our case, divisions, logarithms, or fractional powers like square-roots are avoided as far as possible or they are expressed in a more convenient form unless it is known that they will always be well defined.

The knowledge of the chemical species is required in order to generate the model. The number of chemical species is used to define the size of various vectors like mass in each phase or flowrate in a connection. In particular, the names of the chemical species are important to define the vector of stoichiometric coefficients in reactive systems, and to enable the computation of physical properties.

4.2.- The General Procedure for the Model Generation

The procedures proposed and used for building the mathematical model are presented here and the following sections will describe the type of rate transfer equations generated. As pointed out in chapter two, the problem of modelling concerns the utilization of the conservation laws applied to each region coupled with the characterisation of the transfer rates. We assume that the user possesses the capability to incorporate the description of the system in the terms discussed in the two previous chapters. Thus, the information provided by the user is properly classified to have sets of phases, reservoirs, processes, reactions, controllers, connections, and chemical species.

The general procedure is described as follows:

Procedure 4.1. Model Generation

Model Generation (input, output)
input: phases, reservoirs, processes, reactions, controllers,
connections, and chemical species
output: the mathematical model

Steps:

1 Detect the regions.
2 Generate the mass balance equations.
3 Generate the energy balance equations.
4 For each reaction, generate the characterising equation.
5 For each connection produce the transfer rate equation.
6 Incorporate the equations imposed via processes.
7 Incorporate the numerical values of the properties assigned by the user as well as the ancillary equations.
8 Incorporate the procedures for:
a) flash
b) controllers
c) properties

4.3.- Detecting Regions

The convenience of generating the model based on regions has already been discussed in chapter two. It was also pointed out that the user should not worry about declaring regions. Hence, a procedure to identify the regions has to be established. The identification of the regions is largely related to the connections since phases in equilibrium are declared via connections where the transfer mechanism is defined as equilibration.

The name of a region may come from the name of the phases. However, if there exists a process grouping only the phases in equilibrium then the name of the region is taken based on the process. For example, if the two phases "L" and "V" are declared in equilibrium and the process "P" has been defined to group only these two phases, then the name of the region is referred as "P"; otherwise, the name of the region is referred as the set 'L&V'. The following procedure has been proposed to detect the regions:

Procedure 4.2. Regions Detection Procedure

Regions Detection (input, output)
input: phases, processes and connections.
output: *regions*

Steps:

1 Define *regions* as a set whose initial elements are the names of the phases of the system.
2 **Do** for each connection:
2.1 **If** the transfer mechanism is equilibration **then**
2.1.1 Remove the two connected phases from the set *regions*.
2.1.2 Define a set containing the two interconnected phases.
2.1.3 Include the set in 2.1.2 into the set of *regions*.
3 **Do** for each element of *regions*.
3.1 **If** the element is a set of phases then for each process **do**
3.1.1 **If** the process groups only the phases in equilibrium **then**
3.1.1.1 Reject the set of phases
3.1.1.2 Incorporate the process name into the set *regions*.
4 **Return**

4.4.- Mass Balance Equations

Once the regions have been detected, the first set of equations to be generated, according to the procedure 4.1, is the set of mass balance equations. The following procedure is proposed as a logic sequence for building these equations.

Procedure 4.3. Mass Balance Equations

Mass Balance (input, output)
input: regions, reactions, connections, and species.
output: *model* as a set of equations

Steps:
1 **Do** for each region
1.1 Start an equation by writing the differential term
1.2 **Do** for each connection
1.2.1 **If** the connection relates mass transfer to a phase of the region **then**
1.2.1.1 Add the rate transfer term if it is an output
1.2.1.2 Subtract the rate transfer term if it is an input
1.3 **Do** for each reaction
1.3.1. **If** the reaction involves a phase of the region **then**
1.3.1.1 Add the reaction term
1.4 Equalise the expression to zero and add it to the *model*
2 **Return**

In order to clarify the aims of this procedure, we will consider the following example: suppose that we want to produce the mass balance for a region named "A". According to the procedure 4.3, we start by writing,

$$\frac{d}{dt}\mathbf{m}^{A} \tag{4.1}$$

where $\mathbf{m}^{A}$ is the nc-vector of mass in the region A. Then, the following step is to find the connections related to any phase in the region. The search includes those cases where the phase selective sampling mechanism has been declared. If the result of this search is positive then variables representing the transfer rates are added or subtracted. Let us suppose that the following connection has been declared: C1= {A,B;; TurbulentFlow}. This connection indicates that the flowrate is an output for A. Thus, the expression in (4.1) would be modified as:

$$\frac{d}{dt}\mathbf{m}^{A} + \mathbf{f}^{C1} \tag{4.2}$$

where $\mathbf{f}^{C1}$ is the nc-vector of mass transfer rates through connection C1. In this way, each connection where the phase A is involved is incorporated into the expression.

In the case of reactions, the term includes three parts: the vector of stoichiometric coefficients, the rate of reaction, and the diagonal matrix of molecular weights. The vector of stoichiometric coefficients is obtained based on the identification of the chemical species of the system. The matrix of molecular weights provides consistency in the dimensionality of the equation. Thus, adding the term of reaction to (4.2) and equalising the expression to zero may look like:

$$\frac{d}{dt}\mathbf{m}^{A} + \mathbf{f}^{C1} + r^{R1}\mathbf{M}(1,0,-1,0)' = 0 \tag{4.3}$$

where r^{R1} is the rate of reaction.

4.5.- Energy Balance Equations

The procedure to obtain the energy balance equations is, in a way, a simplified form of the procedure to generate the mass balance equations. We assume that the package of properties provides the mixture enthalpy referred to the same reference state. A reference state may be the ideal gas at a given temperature. The following procedure has been used for building the energy balance equations.

Procedure 4.4. Energy Balance Equations

Energy Balance (input, output)
input: regions, and connections.
output: *model* as a set of equations

Steps:
1 **Do** for each region
1.1 Start an equation by writing the differential terms
1.2 **Do** for each connection
1.2.1 **If** the connection relates to a phase of the region **then**
1.2.1.1 Add the rate transfer term if it is an output
1.2.1.2 Subtract the rate transfer term if it is an input
1.3 Equalise the expression to zero and add it to the *model*
2 **Return**

4.6.- Characterising Reactions and Connections

The steps 4 and 5 in the procedure 4.1 concern the characterisation of the transfer-rate variables introduced in the mass and energy balances. The declarative statement of a reaction or a connection always identifies a type which associates a model. Thus, the work to perform in these steps is to incorporate the equation or equations associated with the model.

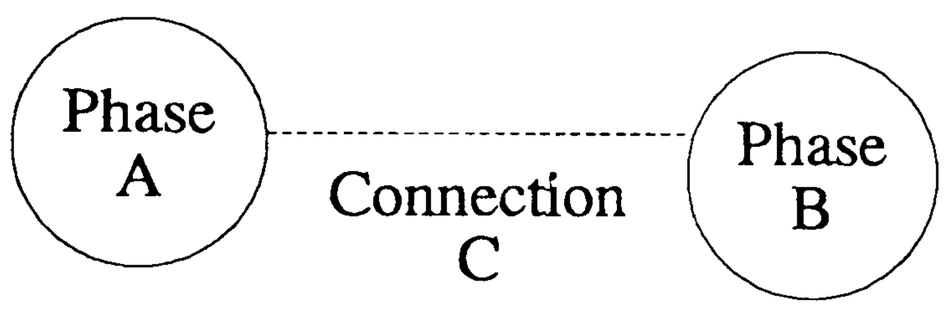
The user should verify that the transfer models he requires already exist in the library of the modelling system. If these models are not there then he should externally develop the model and add it into the system. It may be convenient to

implement facilities to add user-defined transfer laws in the simplest way; however, this work has yet to be done.

Each characterising equation is generated by an appropriate procedure according to the kind of transfer mechanism. However, a general description of these procedures is summarised as follows: detect the interconnected phases and associate the proper model. The description of the models is the topic of the following subsections.

4.6.1.- Aspects of Heat Transfer

The sort of equations concerned here are those equations which describe energy transfer without mass transfer. The energy transfer phenomenon is represented as follows:



where transfer of heat between two phases occurs as a result of a difference in temperature and takes place in one or more of the three different ways:

1. Radiation: This is a phenomenon identical to the emission of light and is a function of the absolute body temperature and the nature of the emitting surface.

2. Conduction: This is transfer by molecular motion between one part of a body to another part of the same body, or between two different bodies in contact with each other.

3. Convection: This is the process by which movement of fluid in bulk carries heat from one place to another. This model is of relevance only to liquids and gases.

The appendix provides a set of examples where models using heat transfer equations are built up, see cases A.2, A.3, A.4, A6 and A.8.

4.6.2.- Heat Transfer by Radiation

All bodies continuously emit energy because of their temperature, and the energy thus emitted is called thermal radiation. The radiation energy emitted by a body is transmitted into the space in the form of electromagnetic waves. The emission or absorption of radiation energy by a body is considered a bulk process; that is, radiation originating from the interior of the body is emitted through the surface of the body [Ozisik, 1985]. Radiation incident on the surface of a body penetrates to the medium where it is attenuated.

The maximum energy flux emitted by a body at temperature T is given by the Stefan-Boltzmann law,

$$E_b = sT^4 \qquad (4.4)$$

where s is a constant of proportionality called the Stefan-Boltzmann constant, and E_b is called the blackbody emissive power.

However, only an ideal radiator or the so called blackbody can emit radiation flux according to (4.4). The radiation flux emitted by a real body is always less than that of the blackbody; it is given by,

$$E = \varepsilon sT^4 \qquad (4.5)$$

where ε is the emissivity of the body and lies between zero and unity.

Thus, when two phases at different temperatures "see" each other, heat is exchanged between them by radiation. This radiation can be treated by differential analysis of emitted rays and their successive components. However, the analysis is complicated and the information required for this evaluation is not usually known with enough accuracy to justify a detailed calculation. A reasonably accurate

treatment is possible if we consider one case which is geometrically simple enough to permit rigorous treatment: the case of a nonblack phase *A* in a nearly black isothermal enclosure phase *B*. The simplified net radiation rate from phase *A* is plainly given by [Bird et al, 1960]:

$$e = As(\varepsilon^{A} T_A^4 - a^{A} T_B^4) \qquad (4.6)$$

where *e* is the heat transfer rate by radiation, *A* is the surface area , and *a* is the absorptivity which is a function of the temperatures in both phases [McAdams, 1958]. To get the appropriate form of equation (4.6), it is necessary to impose the condition of keeping the order in the entities declared in the connection: the first entity refers to the nonblack body and the second entity refers to the enclosure, e.g. *C= {A,B;; HeatRadiation}*.

4.6.3.- Heat Transfer by Conduction

The conduction rate equation used to compute the amount of energy being transferred is known as Fourier's law. For heat flow in the *z* direction, for example, the Fourier law is given by:

$$e = -kA\frac{dT}{dz} \qquad (4.7)$$

where *e* is the rate of heat flow through area A in the positive *z* direction. The proportionality constant *k* is called the thermal conductivity of the material and is a positive quantity.

Considering that the connection C is a plane wall separating two phases A and B, across which the temperature difference exists, the integration of (4.7) if k is constant gives

$$e = \frac{kA}{l}(T^{A} - T^{B}) \qquad (4.8)$$

where T^A and T^B are the temperatures at the hot and cold sides of the wall respectively, l is the wall thickness, and A is the interface area.

4.6.4.- Heat Transfer by Convection

The convection heat transfer mode can manifest in two possible mechanisms: natural or forced convection. When fluid is artificially induced to flow over a surface of transfer, the heat transfer is said to be by forced convection. If the fluid motion is set up by effects resulting, for instance, from density difference, the heat transfer is said to be free or by natural convection.

Regardless of the particular nature of the convection heat transfer mode, the appropriate equation to evaluate the heat transfer rate between phases A and B in a connection C is of the form,

$$E = h(T^A - T^B) \tag{4.9}$$

where E is the convective flux proportional to the difference between the temperatures in both phases. This expression is known as Newton's law of cooling, and the proportionality h is the convection heat transfer coefficient (film conductance or film coefficient). It encompasses all the effects that influence the convection mode, and it depends on various factors like the nature of the fluid motion and the transport properties.

4.6.5.- Modelling Flow Through a Pipe

The simplest case of modelling a connection where mass and energy are simultaneously transferred is the modelling of bulk flow associated with a simple pipe. Pipe lines are perhaps one of the lowest cost means of transportation. The chemical species can flow out or in as a result of a difference of the pressures in the two interconnected entities.

Generally speaking, the literature on pipe modelling is massive, see [Deutsch, 1980]. However, pipe-flow phenomena are frequently complex and often not well

understood. For instance, material specifications are determining factors difficult to incorporate in the theoretical evaluation of the flow. Therefore, it is rare to use a theoretical model because of various practical reasons.

Some practical formulas have been conveniently used by engineers [Coulson et al, 1990]. The model adopted here assumes that the pipe can be modelled as a valve where all resistance to the flow is summarised into one constant C known as the valve constant for the flow, see for instance [Ramirez, 1989]. If the flow is turbulent, this model has the form:

$$w = sgn(\Delta P) d C_t \sqrt{|\Delta P|} \qquad (4.10)$$

where w is the total mass flow, ΔP is the difference of the pressures between the two interconnected entities, d is the density of the fluid, $sgn()$ is a function which detects the sign of ΔP, and the subscript t indicates turbulent flow. Equation (4.10) has been also used to model reverse flow; however, numerical experience points to some difficulties found during its solution. The fact that the derivative with respect to the difference of pressures gives a division by zero when the equality of the pressures is reached seems to be the origin of the difficulties [Jarvis, 1992].

If the flow is laminar, the equation to evaluate the rate of flow takes the form,

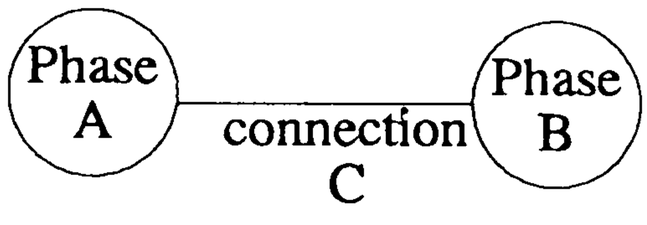
$$w = dC\Delta P \qquad (4.11)$$

The Reynolds number is the dimensionless quantity currently used to determine if the flow is either turbulent or laminar. It has the form Lvd/μ. Here L= a characteristic linear dimension of the flow pipe, v= linear velocity, d= fluid density, and μ= fluid viscosity. The critical Reynolds number corresponds to the transition from turbulent flow to laminar flow as the velocity of the fluid is reduced. Its value depends upon the channel geometry, being in the range of 2000 to 3000 for circular pipe. Unfortunately, the evaluation of the Reynolds number requires the knowledge of the pipe's geometry.

It may be useful to include a procedure to allow the selection of either of the two types of flow and then automatically select the proper model without considering the geometry. We propose here a generalised pressure drop driven model where the type of flow is considered as a function of the difference of pressures ΔP. It is expected that laminar flow appears when the absolute value of ΔP is less than a parameter δ; otherwise, the flow is turbulent, see Figure 4.1.

It follows from (4.10) and (4.11) that the constant C_t and the constant C are related by $C_t = \delta^{1/2} C$.

A subsequent problem is the identification of the fluid phase. It is obvious that the properties of the fluid are related to the phase where the fluid is coming from. Let us consider the interconnection of phases "A" and "B" as shown below:



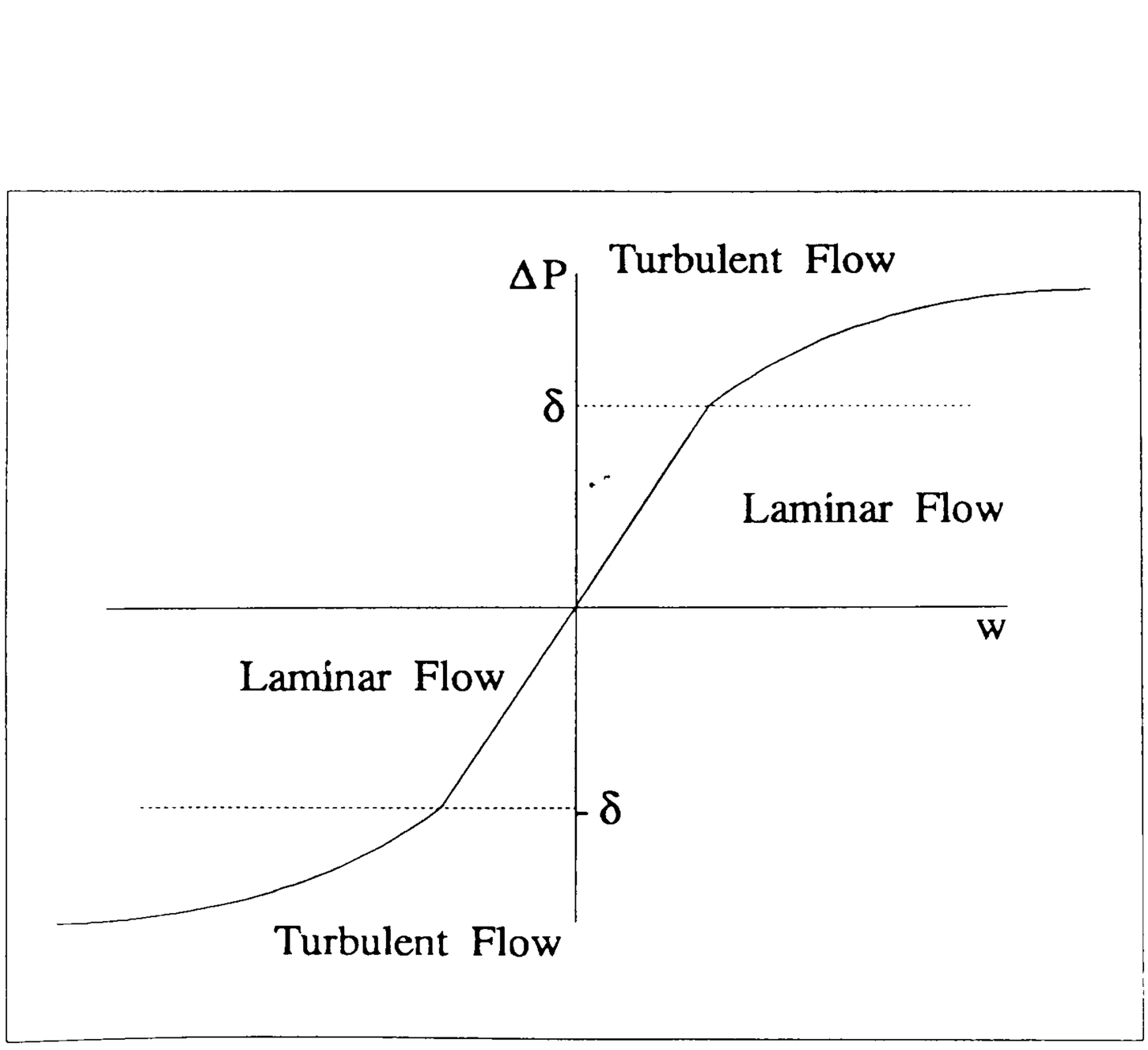


Figure 4.1 *Flow Through a Pipe*

If the fluid is an output for phase A, the intensive properties of the fluid correspond to the intensive properties of A. Otherwise, the properties correspond to phase B.

Introducing the mass flow vector **f** and the mass fraction vector **x**, the general pressure drop driven model may be written as:

if $P^A - P^B > \delta$ **then**

$$\mathbf{f} = (d^A C\sqrt{\delta(P^A - P^B)})\mathbf{x}^A$$

else if $P^A > P^B$ **then**

$$\mathbf{f} = d^A C(P^A - P^B)\mathbf{x}^A$$

else if $P^B - P^A > \delta$ **then** (4.12)

$$\mathbf{f} = -(d^B C\sqrt{\delta(P^B - P^A)})\mathbf{x}^B$$

else

$$\mathbf{f} = d^B C(P^A - P^B)\mathbf{x}^B$$

endif

If there is a valve in the line, the value of C becomes a function of the valve position. Thus, the model above incorporates two important features for modelling process systems: the possibility of reverse flow and the possibility of dealing with a discontinuous operation on any of the interconnected phases.

Another issue related to the flow through a pipe is that the reverse flow may not be feasible, and the user may want to impose this condition. In order to deal with both options unidirectional and reverse flow, we include the possibility of reverse flow just when the user declares pressure drop driven flow. Otherwise, if the user declares turbulent or laminar flow then the flow is considered unidirectional.

Finally, the transfer process is considered adiabatic so that no further heat or work interaction with the connection is considered [Balzhiser and Samuels, 1977]. As a

result, the enthalpy of the fluid remains constant in the transfer process, i.e. the energy balance can be written as follows,

$$H_{in} = H_{out} \tag{4.13}$$

where H_{in} is the total mass enthalpy at the input point, and H_{out} is the total mass enthalpy at the output point.

Introducing the possibility of reverse flow and considering the above example, the energy transferred in the bulk flow through the connection is evaluated as:

if $P^A > P^B$ **then**

$$e = H^A \sum_{i=1}^{nc} f_i$$

else (4.14)

$$e = H^B \sum_{i=1}^{nc} f_i$$

endif

where e is the rate of energy transferred in the bulk flow.

Most of the examples given in the appendix assume turbulent flow in the connections which represent pipes; however, laminar flow is used in the example A.2, and the general pressure drop driven model is used in the example A.9.

4.6.6.- A Multicomponent Model for Mass Diffusion

In this section we discuss the way to model diffusive systems. In general, modelling of mass-transfer systems leads naturally to distributed parameter models. However, lumped models can be developed under some assumptions. The discussion in this section will be limited to transfer processes between uniform phases in the absence of chemical reactions.

In general, the chemical potential gradient may be interpreted as the "driving force" for diffusion [Krishna and Taylor, 1986]. Very often, however, the

movement of a chemical species from one region to another is simplified by considering only one effect: the concentration.

4.6.6.1.- Basic Concepts

A diffusion mixture contains chemical species moving at different velocities where the local molar-average velocity v is defined as:

$$v = \frac{\sum_{i=1}^{nc} c_i v_i}{\sum_{i=1}^{nc} c_i} \qquad (4.15)$$

where c_i is the molar concentration and v_i is the velocity of species i with respect to stationary coordinates. Indeed, the velocity v_i refers to the sum of the velocities of the molecules of species i within a small volume divided by the number of such molecules rather than the velocity of an individual molecule of species i.

The molar flux of species i is a vector quantity denoting the moles of species i that pass through a unit area per unit time. The motion can be referred to stationary coordinates or to the molar-average velocity. Thus, the molar fluxes relative to stationary coordinates are defined as follows:

$$N_i = c_i v_i \qquad (4.16)$$

The molar fluxes relative to the molar-average velocity are the result of a purely diffusive mechanism. They are defined as:

$$J_i = c_i (v_i - v) \qquad (4.17)$$

It follows from (2) and (3) that the molar diffusion flux J_i is,

$$J_i = N_i - y_i \sum_{j=1}^{nc} N_j \tag{4.18}$$

where y defines the molar fraction:

$$y_i = c_i \Big/ \sum_{j=1}^{nc} c_j \tag{4.19}$$

It can be distinguished in equation (4.18) that the molar flux relative to stationary coordinates N_i is the resultant of two quantities:

- The molar flux of i from the bulk motion of the fluid:

$$y_i \sum_{j=1}^{nc} N_j$$

- The molar flux of i resulting from the diffusion on the bulk flow: J_i

The molar fluxes relative to the molar-average velocity, diffusive fluxes, possess the following property:

$$\sum_{i=1}^{nc} J_i = 0 \tag{4.20}$$

The molar fluxes can be characterised by relating the fluxes to the mole fractions and composition gradients in order to obtain the composition profiles. The equation characterising the mass transfer in a binary system is given by Fick's first law of diffusion. Considering only one spatial coordinate, say z, Fick's law is written in terms of J_i as follows:

$$J_i = -D_{ij} \frac{d}{dz} y_i \tag{4.21}$$

where D_{ij} is the mass diffusivity in a binary mixture.

In a multicomponent mixture, the expression for J_i considers, in general, two parts: one associated with the mechanical driving forces and one additional contribution associated with the thermal driving force. The first part contains the contribution of terms describing ordinary (concentration) diffusion, pressure diffusion, and forced diffusion though very often it is simplified to the first contribution. The second part contains the contribution of the thermal diffusion.

In the case of ordinary diffusion and assuming ideal-gas mixtures, the Stefan-Maxwell equations may be used to relate molar fluxes to molar fractions. For one-dimensional diffusion, these equations can be written as:

$$\frac{d}{dz} y_i = \sum_{j=1}^{nc} \frac{y_i N_j - y_j N_i}{D_{ij}} \qquad (4.22)$$

where z is the distance in the direction of the diffusion, and D_{ij} is the pair diffusivity for species i and j.

Indeed, the use of the Stefan-Maxwell equation is supported by both molecular theory [Hirshfelder et al, 1954] and irreversible thermodynamics [Krishna and Taylor, 1986].

4.6.6.2.- The Film Model

In the previous section, the basic governing differential equations prevailing in diffusive mass transfer systems have been presented. Unfortunately, in many practical situations the governing differential equations cannot be solved without gross simplifications. Thus, some additional assumptions have to be considered. These assumptions concern the choice of a simplified picture describing the actual hydrodynamic conditions prevailing in the system. Various models have been proposed and analysed in the literature [Krishna and Taylor, 1986]. Among these, the film theory is probably the simplest and the most useful model.

According to the film model, all the resistance to mass transfer is concentrated in a thin film, or layer, adjacent to the phase boundary. Then, transfer occurs within

this film and beyond the film properties are referred to as bulk properties. Mass transfer occurs through the film essentially in the direction normal to the interface; that is, any constituent molecular diffusion or convection in the laminar flow parallel to the surface due to composition gradients along the interface are negligible in comparison to the normal transfers.

Having made the simplification to the hydrodynamics, the remaining task is to solve the differential equations describing the molecular diffusion process in the film, i.e. Stefan-Maxwell equations have to be solved. Sargent [1989] expresses the system of equations in matrix form as follows:

$$\frac{d\mathbf{y}}{d\zeta} = \Phi\mathbf{y} \qquad (4.23)$$

where ζ is a dimensionless parameter defined as $\zeta = z/t$ being t the film thickness, $\mathbf{y}$ is the column vector of mole fractions and Φ is a square matrix, known as the "transfer matrix", with elements:

$$\Phi_{ij} = \frac{tN_i}{D_{ij}} \quad , \quad i \neq j \qquad (4.24)$$

$$\Phi_{ii} = -\sum_{\substack{j=1 \\ j\neq i}}^{nc} \Phi_{ji} \quad , \quad i = j \qquad (4.25)$$

In general, the fluxes N_i are constant over ζ. Thus, if we assume that D_{ij} remains constant, (4.23) can be integrated to have an expression of the form,

$$\mathbf{y} = \exp(\Phi)\mathbf{y}' \qquad (4.26)$$

where x is the bulk mixture composition and $\mathbf{y}'$ is the composition at the interface. It must be noticed that each column of Φ sums to zero. This fact provides consistency since the sum of fractions is always equal to unity. Krishna

and Standart [1976] prefer avoiding this feature and write the Stefan-Maxwell equations for only (nc-1) chemical species.

4.6.6.3.- The Model Used in This Study

If we consider two phases "A" and "B" which are interconnected to simultaneous mass and energy transfer then what we may use is the so called two-film model. Adopting the convention that transfers are from phase "A" to phase "B", the two-film model can be formulated as follows:

$$\frac{d\mathbf{y}^{A}}{d\zeta} = \Phi^{A}\mathbf{y}^{A}, \quad \mathbf{y}(0) = \mathbf{y}^{A}, \mathbf{y}(1) = \mathbf{y}^{\Lambda,I} \qquad (4.27.a)$$

$$\frac{d\mathbf{y}^{B}}{d\zeta} = \Phi^{B}\mathbf{y}^{B}, \quad \mathbf{y}(0) = \mathbf{y}^{B,I}, \mathbf{y}(1) = \mathbf{y}^{B} \qquad (4.27.b)$$

from ζ=0 to ζ=1.

However, current practice is to lump all the factors influencing the actual rates of transfer into mass transfer coefficients which is in fact a linear form. For instance, while analysing binary mass transfer systems, Bird et al [1960] introduced a mass transfer coefficient in terms of the rate of diffusion. Thus, the diffusional contribution was obtained by multiplying the concentration gradient with the mass transfer coefficient, i.e.

$$J_{ib} = k\Delta y_i \qquad (4.28)$$

where k is the binary mass transfer coefficient. In terms of the molar flux **N**, the above equation gives

$$N_i - y_i \sum_{j=1}^{2} N_j = k\Delta y_i \qquad (4.29)$$

A treatment of mass transfer in multicomponent systems has been done by Krishna and Standart [1979] where the mass coefficients are consistent with the binary coefficients. This analysis leads to the following type of equation:

$$N_i - y_i \sum_{j=1}^{nc} N_j = \sum_{k=1}^{nc-1} k_{ik}\left(y_k - y_k^I\right) \qquad (4.30)$$

The utilization of (4.30) in the analysis for multicomponent mixtures results in only (nc-1) independent equations since the molar diffusion fluxes sum over the nc species to zero (see equation 4.20). Thus, the analysis can be carried out by using (nc-1) dimensional form.

If there is continuity of mass and energy fluxes at the interface between phases "A" and "B" we have, in matrix form,

$$< \mathbf{N} >= \mathbf{k}^A < \left(\mathbf{y}^A - \mathbf{y}^{A,I}\right) > + (\sum_{i=1}^{nc} \mathbf{N}_i) < \mathbf{y}^A > \quad (4.31.a)$$

$$< \mathbf{N} >= \mathbf{k}^B < \left(\mathbf{y}^{B,I} - \mathbf{y}^B\right) > + (\sum_{i=1}^{nc} \mathbf{N}_i) < \mathbf{y}^B > \quad (4.31.b)$$

where "<>" is used to indicate that the mathematical operation is on the first (nc - 1) components of an nc-vector, the superscripts indicate the part of the system referred, and **k** is defined as a (nc-1)(nc-1) matrix.

The energy transfer rate is also made up of a diffusive contribution and a convective term. We use the already introduced heat transfer coefficient:

$$e = Ah^A(T^A - T^{A,I}) + H^A(\mathbf{f} \bullet \mathbf{x}'^A) \qquad (4.32.a)$$

$$e = Ah^B(T^{B,I} - T^B) + H^B(\mathbf{f} \bullet \mathbf{x}'^B) \qquad (4.32.b)$$

where H^A and H^B contains the specific total enthalpy, the mass flow is related to the fluxes by $\mathbf{f} = A\mathbf{MN}$, being A the area of the interface, and •

indicates the inner product. Note that mass fraction **x** is used to be dimensionally consistent.

Then, some considerations are taken in relation to the interface. In the absence of any better information, it is common to assume that equilibrium prevails at the interface [Krishna and Taylor, 1986]. Thus, the usual equations of phase equilibrium relates the mole fractions on each side of the interface. Krishna and Taylor, [1986] use the equilibrium ratios as follows:

$$K_i y_i^{A,I} = y_i^{B,I}; \quad i = 1,2,\ldots,nc \tag{4.33.a}$$

$$\sum_{i=1}^{nc} y_i^{A,I} = 1; \quad \sum_{i=1}^{nc} y_i^{B,I} = 1 \tag{4.33.b}$$

where $K_i = K_i(T^I, \mathbf{y}^{A,I}, \mathbf{y}^{B,I}, P)$ are the equilibrium ratios.

There exist various methods to evaluate the matrices of multicomponent diffusive mass transfer coefficients $\mathbf{k}^A$ and $\mathbf{k}^B$, the heat transfer coefficients h^B and h^A, and the equilibrium ratios matrix **K**. Krishna and Taylor [1986] give a review on the methods used to estimate the mass transfer coefficients, some of which are based on solutions to the Stefan-Maxwell equations.

The two film model is used in the examples A.4 and A.5 of the appendix. Note that the conversion from mass fraction to mol fraction is required to be consistent with the model above described.

4.6.7.- Modelling Reactive Systems

The stoichiometric equation of a chemical reaction involving the chemical species A_i with stoichiometric coefficients σ_i may be written as:

$$\sum_{i=1}^{nc} \sigma_i A_i = 0 \tag{4.34}$$

where σ_i is negative for reactants, positive for products, and zero for non-modified species.

Then, the rate of a chemical reaction is defined as the common value obtained by dividing the molar production rate of the chemical species with their respective stoichiometric coefficient [Reklaitis, 1983], i.e.

$$r = \frac{R_i}{\sigma_i}, \quad \sigma_i \neq 0 \tag{4.35}$$

where r is the rate of reaction and R_i is the molar production rate of species i.

The question arising at this point is: How can we evaluate the rate of reaction?. The rate of reaction is characterised by a rate equation whose form may be suggested by theoretical considerations or it may simply be an empirical expression. In practice, the values of the constants of the rate equation can only be found by experiment; predictive methods are still inadequate at present unless the assumption of instantaneous equilibrium is considered. It is precisely the experimental nature of reactive systems which limits our possibilities of generalization.

The rate of a chemical reaction is normally affected by many variables. While in homogeneous reactions it is possible to think of pressure, temperature and composition as the only involved variables, in heterogeneous reactions the problem becomes more complex.

In the study of chemical reactions, the first step is very often the proposal of appropriate mechanisms to show how some elementary steps may be combined to produce the overall reactions. Thus, the rate of reaction may be the result of incorporating the contribution of each "pseudoreaction".

Chemists assert that, strictly speaking, all reactions should be considered reversible. This means that whenever a reaction is deemed to occur, its reverse should be deemed to occur as well. Sometimes, however, the occurrence rate of the reverse reaction is very small. For the purpose of modelling, one often neglects the occurrence of those reactions that are regarded to proceed at very small rates.

The following are examples of the description of the rate of reaction to be included in a general package for process model building:

- A reaction is reversible and follows the mass action law
- A reaction is irreversible and follows the mass action law
- A reaction obeys an instantaneous equilibrium
- A reaction follows different kinetics.
- A reaction rate is given by an empirical expression.

The mass action law states that the rate of reaction is proportional to the product of all the molar species concentrations, each raised to a power given by the corresponding stoichiometric coefficient in the reactant complex. This sort of model is used even for complex reactions. For instance, if we consider the reaction,

$$2[A](L) + [B](L) = 2[C](L) \qquad (4.36)$$

then the rate of reaction, considering a reversible mass action model, may be written as,

$$r = Kr_1 c_C^2 - Kr_2 c_A^2 c_B \qquad (4.37)$$

where c's are the concentration of the chemical species in the phase and Kr's are constants of proportionality.

The constants of proportionality in the mass action model may follow the Arrhenius law for dependence on the temperature:

$$Kr = A \bullet e^{\frac{-E}{R \bullet T}} \qquad (4.38)$$

where A is a frequency factor, E is the activation energy, R is the gas constant, and T is the temperature.

Another general model, also to be considered here, is that where the rate of reaction is proportional to the concentration of one or more components of the

reaction raised to a power which is completely empirical. This case is called the polynomial kinetics. For instance, if we suppose that the reaction above follows an order one based on the chemical species [C] then the model takes the form:

$$r = Krc_C \tag{4.39}$$

Finally, the instantaneous equilibrium implies that the degree of advancement of a chemical reaction is characterised by an equation of the form:

$$Kr_e = \prod_{j=1}^{nc} c_j^{\sigma_j} \tag{4.40}$$

where Kr_e is the equilibrium constant.

If the reaction is carried out in more than one phase the empirical formalism is even higher. However, some models are developed on some theoretical basis. For instance, the simple heterogeneous reaction

$$A = R$$

on the surface of a solid catalysis has been modelled by applying Langmuir isotherms to generate the rate equation termed Langmuir-Hinshelwood-Hougen-Watson (L-H-H-W) [Froment and Bischoff, 1979]:

$$r = \frac{Kr_1 \bullet (c_A - \frac{c_R}{Kr_2})}{1 + Kr_3 \bullet c_A + Kr_4 \bullet c_R} \tag{4.41}$$

where Kr's are constants which are experimentally determined, and C's are molar concentrations of the chemical species in the fluid phase.

At present we have only implemented the general cases analysed above. It is believed, however, that implementing particular models is not a difficult task.

The models of some reactive systems have been developed in the appendix: example A.6 considers a homogeneous reaction, example A.7 considers a heterogeneous reaction, and example A.9 considers a multiple reactive system.

A problem may arise in the evaluation of the rate of reaction if one of the phases involved in the reaction disappears. A conditional statement is introduced in the model to overcome this problem (see examples in the appendix).

4.7.- Additional Equations

Under the title of additional equations we group the equations which are obtained from imposing constraints on grouped phases and the ancillary equations, steps 6 and 7 of the procedure 4.1.

In chapter two we also discussed the possibility of imposing some conditions on a group of phases grouped in a process. The step 6 of the procedure 4.1 refers to the inclusion of these equations into the model. Then, the task to perform is just look for the characteristics imposed in each process and provide the appropriate equation. For instance, if constancy of the sum of volumes was required then an equation considering the sum of volumes equal to a parameter should be incorporated.

The numerical values provided by the user are indicated as a set of equations. However, this should be internally dealt with as a problem of assignment.

Finally, there is a group of equations which are named here ancillary. They include the relations between the mass fraction and the masses of each chemical species and the equations to relate the density of phases with their respective occupied volume.

4.8.- Incorporating Procedures

The final step in the procedure 4.1 implies the incorporation of those equations which are part of a computing procedure. In general, a model may have three

kinds of procedures: flashes, controllers, and properties. The inputs and outputs are explicitly indicated in the argument list in order to identify the degrees of freedom of the model.

The controllers are modelled as a sort of black box where, given some inputs, we compute the outputs.

The user is expected to declare all the possible phases in equilibrium. Then, a flash procedure is indicated to evaluate the mass distribution in those phases which coexist at equilibrium. This procedure has the following form:

flash{input: $\mathbf{m}^r$, U^r, P^r; output: T^r, s^A, $\mathbf{m}^A$,U^A,P^A, T^A, d^A s^B, $\mathbf{m}^B$,U^B, P^B, T^B, d^B ...}

where the superscript *A* and *B* refer to the phases in equilibrium, and *r* refers to the region, *s* indicates the state of the phase: vapour, liquid, or solid, **m** is the nc-vector of mass, *U* is the internal energy, *P* is the pressure, *d* is the density, and *T* is the temperature.

We preferred the use of a flash procedure rather than leaving the equations describing the equilibrium because we expect the flash procedure to give consistent values when any of the phases disappears. In addition, a specialised procedure can be implemented in order to improve the speed in achieving the numerical solution. The standard thermodynamic state (temperature, pressure, and masses of each species) is not used in this case because pressure and temperature become dependent in the single chemical species case. The nc-vector of mass, internal energy, pressure, and temperature of each phase will be evaluated and properly indicated in the flash procedure.

The sequence of the arguments listed in the flash procedure is in order of increasing density. The user could verify if the status (vapour, liquid, or solid) of the phases correspond to his expectation by observing the output of the "*s*" variables in the flash procedure. For instance, if the flash above detects liquid and vapour then the phase *A* is indicated to be vapour and phase *B* liquid.

A problem arising here concerns the identification of the order of the phases according to the user's point of view. Here, we adopt the following convention: the user should label his phases in equilibrium starting with "S", "L", or "V" to

indicate solid, liquid, or vapour respectively; however, if the phases do not start with any of these letters (S, L, or V) then the alpha-numerical order is considered. For instance, if the user declares that phases "L1" and "V1" are in equilibrium then the order given in the flash procedure is "V1" then "L1" and the solution of the model will provide the actual status of the phases, i.e. vapour or liquid; if the phases "Pa" and "Pb" are in equilibrium then the order of increasing density is "Pa", "Pb"; etc.

We proposed linkage to a properties package. Then, the properties required by the model are just indicated and the inputs are the state of the phases as in the flash above (see examples in the appendix). We left the incorporation of the properties at the end in order to perform a search to find all the properties required in the model.

4.9.- Methods for Determining Sets of Free Variables

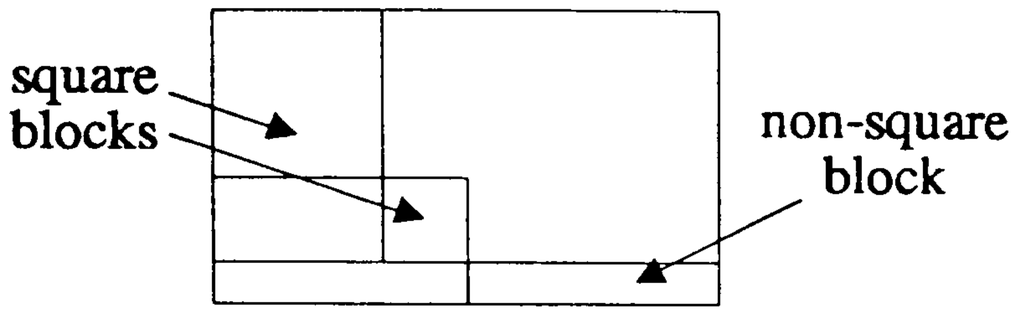
An issue arising here concerns the decision about which variables have to be fixed once the mathematical model has been generated. In this section we describe one current technique for determining the free variables and, then, a procedure for selecting one set of free variables is proposed.

Several techniques for finding the free variables have been used in different commercial flowsheeting packages like SpeedUp. The structure of the system constitutes the basic information for this analysis in all these techniques.

The analysis depends on the way in which the variables occur in the equations. This information is encapsulated in the occurrence matrix "**M**" whose elements are defined by,

$$\mathbf{M}_{i,j} = 1 \text{ if variable j occurs in equation i,}$$
$$= 0 \text{ otherwise.}$$

The rectangular system represented by **M** is rearranged to a block triangular form to identify square blocks which must be solved independently. It may produce a structure of the type,



Any attempt to specify a variable belonging to a square block gives an overspecified system. Thus, the appropriate potential free variables are those remaining in the last block. The block triangular form can be generated using any of the methods suggested by [Duff, 1981; Sargent and Westerberg, 1964; Steward, 1965].

For instance, let us consider the following system of four equations in six variables given in [Pantelides, 1989].

$$f_1(x_1, x_2, x_3, x_5) = 0 \qquad (4.42.a)$$
$$f_2(x_2, x_4) = 0 \qquad (4.42.b)$$
$$f_3(x_1, x_2, x_6) = 0 \qquad (4.42.c)$$
$$f_4(x_2, x_4) = 0 \qquad (4.42.d)$$

The corresponding occurrence matrix of equations in system (4.42) is given in the following Figure 4.2.

	VARIABLES					
	x_1	x_2	x_3	x_4	x_5	x_6
Eq a	1	1	1		1	
Eq b		1		1		
Eq c	1	1				1
Eq d		1		1		

Figure 4.2 *Incidence Matrix of System (4.42)*

Appropriate permutation of the rows and columns of the incidence matrix in Figure 4.2 leads to the block triangular form given in Figure 4.3.

	VARIABLES					
	x_2	x_4	x_1	x_3	x_5	x_6
Eq b	1	1				
Eq d	1	1				
Eq a	1		1	1	1	
Eq c	1		1			1

Figure 4.3 *Block Triangular Matrix of System (4.42)*

Then, it is clear that we cannot specify either x_2 or x_4 but we can specify any of the following: x_1, x_3, x_5, and x_6. Therefore, all appropriate free variables are contained in the set $\{x_1, x_3, x_5, x_6\}$.

However, there is a problem which arises with this set of free variables. Since there are two degrees of freedom, we have to select only two variables from this set. If we specify x_1 in the system (4.42) any attempt to specify x_6 would cause the system to become singular. To overcome this problem the procedure of block triangularisation has to be repeated "one variable at a time". For instance, specification of x_1 in system (4.42) is equivalent to introducing another equation of the form $f(x_1) = 0$. The following block triangularisation is given in Figure 4.4 where it is evident that x_6 cannot be specified given x_1, but either x_3 or x_5 can be specified.

	VARIABLES					
	x_2	x_4	x_1	x_6	x_3	x_5
Eq b	1	1				
Eq d	1	1				
			1			
Eq c	1		1	1		
Eq a	1		1		1	1

Figure 4.4 *Block Triangular Matrix of System (4.42) After Fixing* x_1

4.9.1.- A Procedure to Suggest a Single Set of Free Variables

The uncertainty about which variables should be fixed is increased if the number of free variables is much greater than the degrees of freedom of the system. From a user's point of view, it would be beneficial if a single set of free variables were automatically generated. Then, the number of variables contained in the set should be equal to the number of degrees of freedom. Obviously, users would still have the option of fixing any of the free variables suggested by current block triangularisation techniques and performing the "one variable at a time" verification.

The problem can be better described making use of some terminology from graph theory. By a graph we will mean a finite set V together with a subset E of the set of pairs of distinct members of V which is represented as G= (V,E). The members of V are known as vertices and the members of E are known as edges. An edge incident with vertices v and w is written {v,w}. A set $M \subseteq E$ is a matching if no vertex $v \in V$ is incident with more than one edge in M. The graph G=(V,E) is bipartite if the set of vertices V can be partitioned into two sets X and Y, such that each edge of G joins a vertex in X with a vertex in Y. In a system of equations, an element of X may correspond to a variable and an element of Y may correspond to an equation.

If the system of equations is not redundant there must be at least one possible assignment of output variables to the equations. The problem is then formulated as finding a "maximum matching" which is a matching of maximum cardinality. In fact, the cardinality must equalise the number of equations, otherwise the equations contain redundancies [Sargent, 1978].

A bipartite graph for the system (4.42), where a maximum matching is indicated in dark edges, is shown in Figure 4.5.

Various algorithms have been reported in the literature to find a maximum matching. Sargent [1978] states the problem of finding an optimum maximum matching. However, faster algorithms are available if any maximum matching is satisfactory, see Hopcroft and Karp [1973], or Duff [1981].

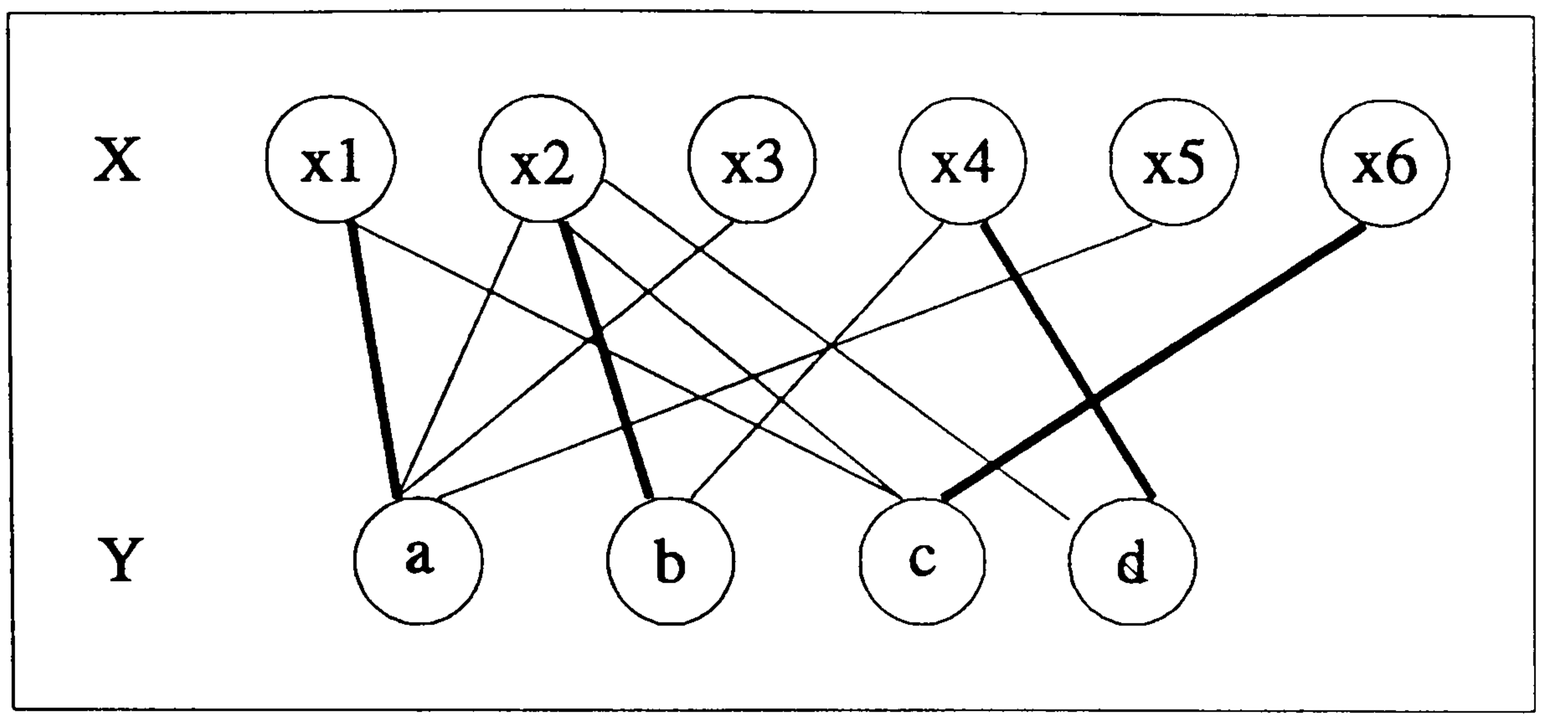


Figure 4.5 *Bipartite Graph*

The single set of free variables is then composed of those variables which are not a part of the maximum matching.

CHAPTER FIVE

AN OBJECT-ORIENTED IMPLEMENTATION

"With computers the problem may look trivial"

An object-oriented implementation referred to as the prototype has been developed in the course of this research. The program is used as a stand alone tool since the models generated follow a syntax which is not compatible with any particular software. In the future we hope to incorporate the possibility of solving the set of differential algebraic equations to produce a more useful package. At present, the prototype is only used to demonstrate the automatic generation of process systems models.

This chapter is devoted to the description of the prototype. We start discussing the important concepts and the characteristics of object-oriented programs; then, we describe the main features of the prototype.

5.1.- The Object-Oriented Technique

Object-oriented programming has been an increasingly popular methodology for software development. The term was first used to describe the Smalltalk programming environment developed at Xerox Palo Alto Research Centre in the early 70's [Deutsch and Goldberg, 1991]. The basic idea of this philosophy is very much supported by our natural tendency to put everything we observe in a category or class. Objects are the basic elements incorporated in the software; they correspond to entities in the real world.

Though the power of the object-oriented style has become widely appreciated, there are different views of what object-oriented should be. It is not possible to give a single and clear definition of object-oriented programming. In general, the object-oriented concepts presented in this section will be based on the Smalltalk point of view [Smalltalk, 1991].

Objects are protected data structures containing two basic elements which are kept apart in conventional programming languages: data and code, see Figure 5.1. Data refers to the private memory of an object where its own internal state is maintained. This information consists of instance variables which, in the Smalltalk programming language, contain either pointers or bytes. Code refers to a set of mechanisms for altering the state of an object. In some programming languages like Fortran or Pascal, the code may refer to a set of subroutines or functions; in Smalltalk a subroutine code is called a method.

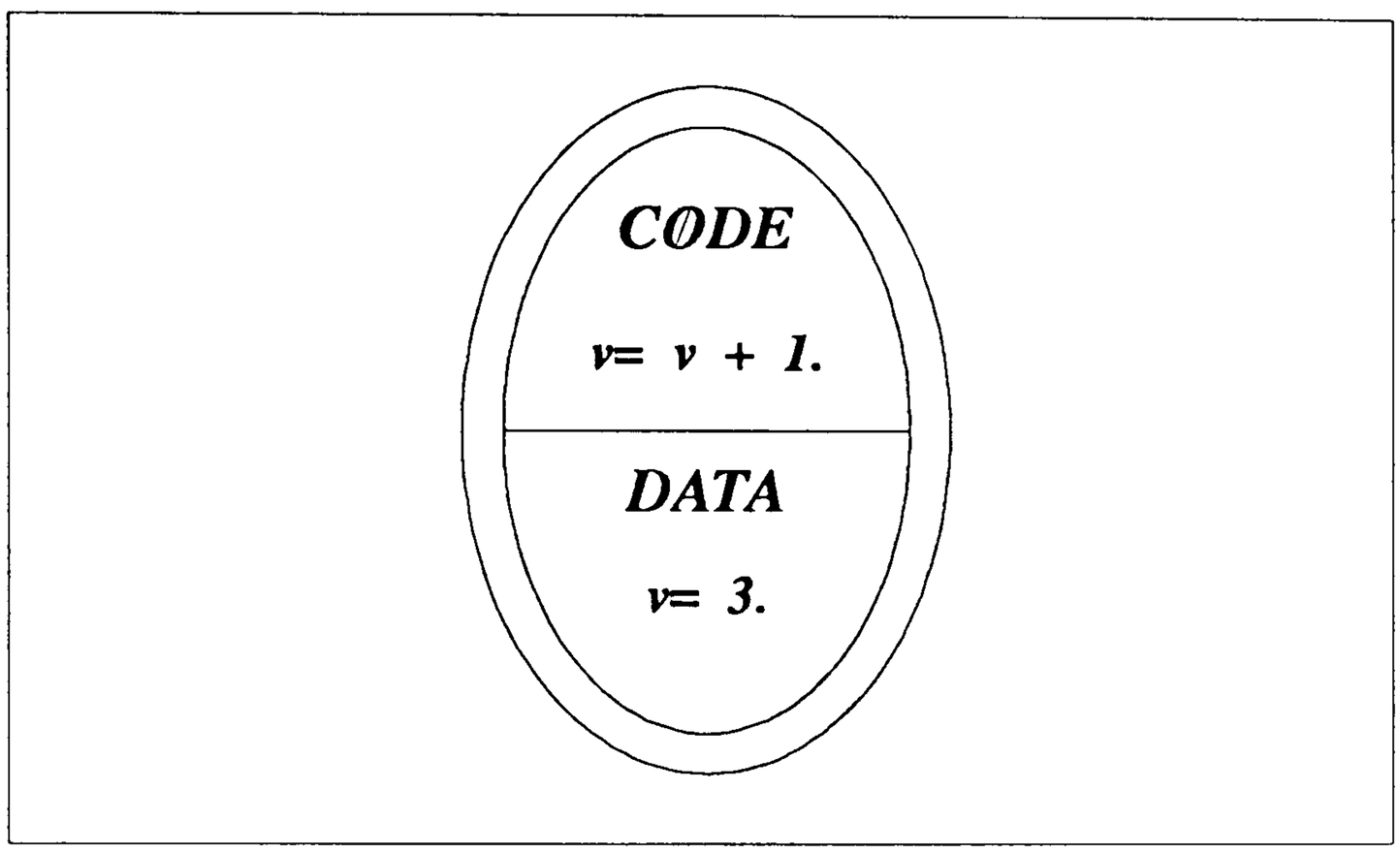


Figure 5.1 *Object Representation*

A programming language is considered to support the object-oriented style if it provides most of the following characteristics [Pascoe, 1986]:

• Data abstraction. The programmer defines an abstract data type consisting of an internal representation that is manipulated and accessed through an abstract interface of access procedures. This means that data is hidden and no object can directly read or modify the state of any other object so that data is visible only from within a localised set of procedures or methods. The data stored inside an object is accessible to other objects only via messages. As a result, the access to any part of the state of an object is only possible if the object itself permits it.

When a message is received, the object invokes a body of code, a method, local to itself and typically generates a response. The strict enforcement of data hiding is known as encapsulation [Micallef, 1988].

• Memory management is done automatically. An object has a unique identity but its state may change throughout its life-time. Objects can be created, they can respond to stimuli from the environment and they can be terminated when they are no longer needed. Thus, the memory space is reclaimed automatically by the system when there is no other object referring to the object.

• Classes are used to describe objects. The idea of a class associated with the concept of data abstraction has been given in Butler and Corbin [1990]. The concept of an object, previously outlined and shown in Figure 5.1, suggests that every object "carries around" with it its code. This would be quite expensive in terms of storage space and speed. Reusability has been the solution proposed in the object-oriented philosophy to overcome this problem [Micallef, 1988]. Thus, many objects may share the same code in some way. This is an important attribute which supports the use of Classes so that objects can be specialisations of other objects. Creating a specialisation of an existing class is called subclassing. Then, the new class is a subclass of the existing class, and the existing class is the superclass of the new class. Individual objects are considered "instances" of a given class and share the same functionality. Then, objects of the same class have the same structure, possess their own state, and share the same program code defined within the class.

• Inheritance. It refers to the mechanism which enables programmers to reuse the code. The inheritance mechanism supported in object-oriented programming languages can be identified in two directions: instances of a class and subclasses. A subclass inherits instance variables and methods from its superclass. Instances of a class inherit the structure from the class to which they belong.

• Polymorphism and Dynamic Binding. A problem may exist in some programming languages if a particular instance variable is used to stack different type of elements, i.e. integer, string, etc. The solution given to this problem, in the object-oriented languages, is the use of dynamic binding. Dynamic binding is a facility for generalising procedure calls; it allows, in the context of the call, to decide precisely which operation is performed according to the type of recipient. Thus, different objects respond in different ways to the same message if the

responses to the message have been programmed. This property results in the idea of polymorphism which means literally the ability to take many forms, i.e. declared variables may take different types and each answer is a function of the actual type. Thus, an instance variable can just be activated with a logic value, or, it can be fixed with a numerical value.

- Multiple Inheritance. It means that a class may inherit from more than one superclass.

Nowadays, the object-oriented philosophy is not exclusive to programming languages. Utilisation of object-oriented concepts in diverse areas has received increasing attention in the last few years. Developments with object-oriented conceptualisation can be summarised as follows:

- Object-oriented Programming is a method of implementation in which programs are organised as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes. This is achieved when the implemented program is written within any of the object-oriented programming languages.

- Object-oriented Design is a method where the characteristics of the system under design are conceived using the object-oriented technique.

- Object-oriented Analysis is a method of analysis in which requirements are examined from the perspective of the classes and objects found in the vocabulary of the problem domain.

Obviously, the object-oriented approach does not provide the solution to every problem. However, many practical problems can be naturally structured into an object-oriented environment. In particular, it was observed that the object-oriented formulation fits well the approach for process model-building presented in this thesis.

5.2.- Object-Oriented Programming Languages

Having observed that the approach can be naturally described in an object-oriented technique, a question arises about what programming language should be used. A survey of most of the object-oriented programming languages in use can be found in Saunders [1989]. In this section we give a short overview of three of the most important object-oriented programming languages.

Smalltalk

In general terms, Smalltalk is an integrated programming environment which uses a bitmapped display, mouse, and windows [Smalltalk, 1991]. It is also the origin of most of the object-oriented terminology used today. Smalltalk is totally object-oriented in the sense that everything is an object and all computing is done by sending messages. Some of the advantages of using Smalltalk are that it is excellent for rapid prototyping and good for building graphical user interfaces.

Lisp Extensions

Lisp itself is not an object-oriented language but it is easily extended to support object-oriented programming. One of the more used ones is Flavors which supports multiple inheritance and it is dynamic in the sense that new classes (flavors) can be created at run time [Allen et al, 1984].

C++

This is an extension to C that supports object-oriented programming with multiple inheritance [Stroustrup, 1988]. It is intended to be an improvement to C. The use of C++ is rapidly increasing, especially within industry, perhaps because of its origin in C and it is supported by a few influential organisations. It supports multiple inheritance and the classes are not considered as objects.

A deeper comparison of some selected programming languages is found in Wolczko [1990]. Based on his analysis, Smalltalk seemed to be a good option to develop the prototype for this study. Thus, the prototype is a computer program written in Smalltalk/v for Windows which is a programming environment for IBM compatible personal computers.

5.3.- Introduction to the Prototype

How was the implementation developed? There is a large number of ways to implement any computer system. The design strategy currently used by most organisations tends to be a rather informal version of the top-down strategy where the designer tries to design the major parts of the system first, then breaks these parts into smaller parts, and so forth. Very often, however, the strategy used to code the modules tends to be rather random. The way in which the program code was built followed an incremental implementation approach combined with the technique of prototyping, the "Build-Then-Try" school. The incremental implementation strategy can be paraphrased in the following manner:

1) Design, code and test one module by itself.
2) Add another module.
3) Test and debug the combination.
4) Repeat steps 2 and 3.

The prototype provides an environment where the process system can be described in the terms discussed in the previous chapters. The general structure can be divided into two parts which are discussed in the following sections: one external part which corresponds to the definition of the interface user-code, and one internal part which contemplates the structure of the classes.

5.4.- The Interface Architecture

It is not our purpose in the implementation, however, to oblige users to learn the object-oriented philosophy. Hence, while the prototype presented in this section was programmed in an object-oriented environment and the internal design follows the object-oriented design, the interaction with the user excludes the object-oriented terminology. Indeed, it was believed that one of the problems in most of the object-oriented applications is precisely that the code obliges the user to think in this style which sometimes may be too complex. The aim of any interface architecture should be to generate a user-friendly system.

We have created a new class called **ProcessSystem** as a subclass of **ViewManager** to inherit a collection of useful variables and methods which implement the basic structure and behaviour of application windows in

Smalltalk/V. Then, the first action upon running the prototype consists of opening a window which interfaces with the user. The interface is then an instance of the new class. Various methods have been developed in order to detect some mistakes like declaring one entity more than once, and to draw the appropriate figure on the screen since the graphic representation has been strongly supported in the present implementation.

Though the development of the methods of the class **ProcessSystem** requires some ability and hard work, the internal structure is rather simple: the class contains a set of methods which are activated by proper messages.

The purpose of this section is to describe the capabilities of the interface, perhaps avoiding the users manual level but giving a taste of the system, rather than describing the methods.

The first step of developing the interface is to define what a window that helps solve the problem would look like. Thus, the proposed window interface was designed to contain three sections: menus, a graphical pane, and a list pane. The structure of this window is shown in Figure 5.2.

Two non-standard menus have been defined in this application. They are labelled "Help" and "MoPS". Figure 5.3 shows the design and the elements contained in the two menus.

The menus are displayed by clicking on the proper item with the left button of the mouse. Then, the subsequent actions are activated by releasing the mouse while indicating the proper item. The messages included in the "Help" menu and the reply as a result of their activation are explained in the following list:

- *Read Representation:* The file should contain the definition of the system which appears on the graphical and list panes. It is recommended to read only files which have been generated by the system.

- *Save Representation:* The information of the process representation is properly saved for future reincorporation into the system.

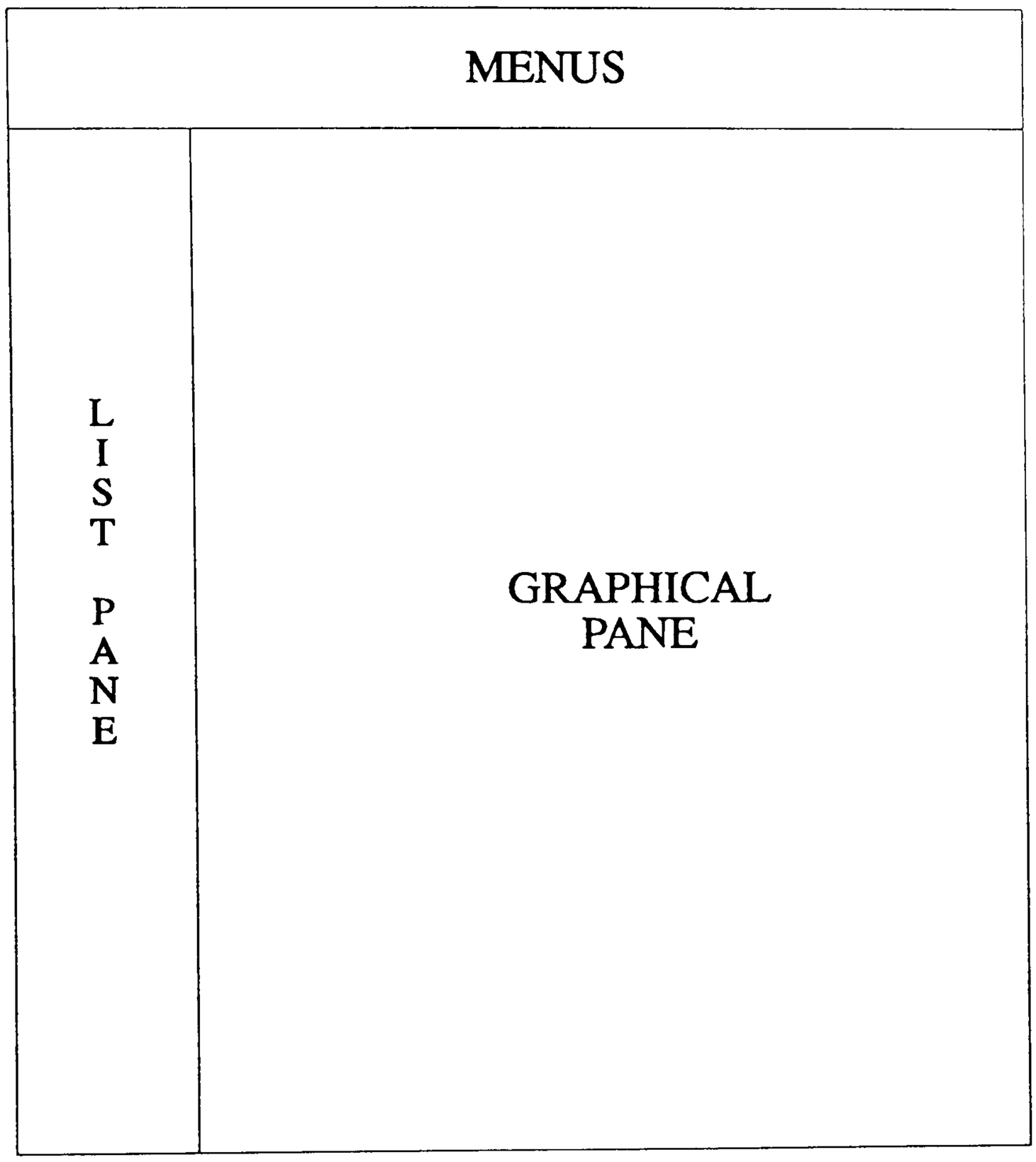


Figure 5.2 *The Window Model*

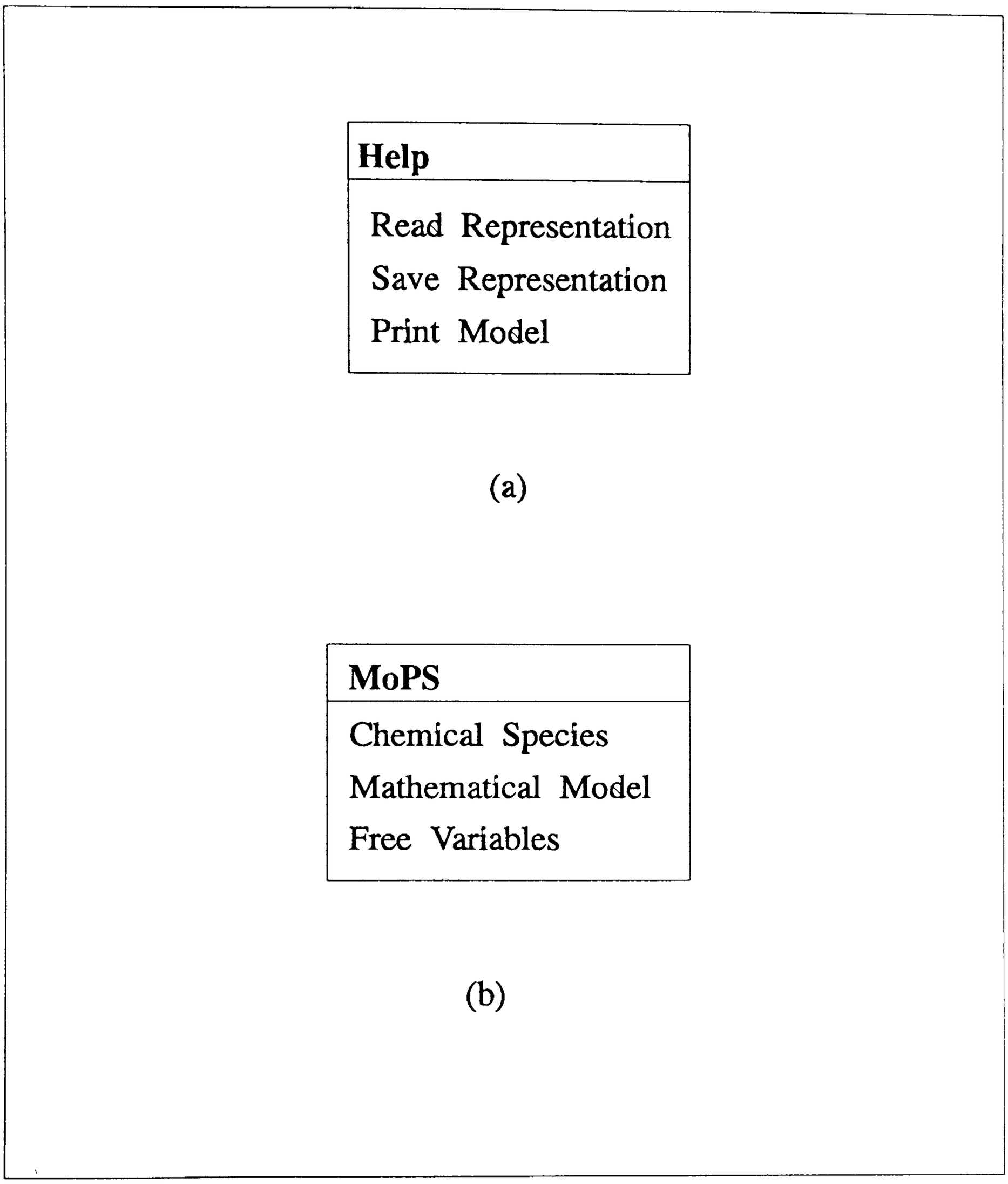


Figure 5.3 *Menus Adopted in the Window Model*

• *Print Model:* The model may be directly printed or transferred to a file which can be edited in order to document the solved problem in whatever form the user prefers. In addition to the mathematical model, the printing includes the nomenclature and a report of the number of variables, the number of equations, the number of initial conditions required, and the degrees of freedom.

The label "MoPS" in the second menu stands for Modelling Process Systems. The messages included in this menu are as follows:

• *Chemical Species:* Chemical components should be defined according to the identification of those species in some properties package. The user will be prompted to define the chemical species.

• *Mathematical Model:* Once the representation is believed to hold the process system being modelled, the mathematical model may be produced on request by the user. The user sends the message to generate the model by clicking on this item. The reply is a window box containing the generated model.

• *Free Variables:* A set of suggested free variables is produced on users request. The result is a window containing a list of the free variables.

The graphic pane includes another menu. This menu appears after clicking on the graphic pane with the left button of the mouse. The menu shown in Figure 5.4 will appear on the screen. Note that the items shown in the menu correspond to the elements of the process representation. Thus, the selection of any of these items will activate an interactive dialogue to incorporate the required information into the system. Then, the entity will be incorporated into both the graphic pane and the list pane. In order to make the entities accessible from every object in the environment, the programming language imposes the condition that their names begin with a capital letter.

• *Phase:* After clicking on this item, the user will be prompted to provide the declarative information of the phase. The form of declaration corresponds to that provided in chapter three. Subsequently, a circle indicating the new phase and its corresponding name, written inside the circle, are displayed in the position where the left mouse button has been released. In addition, the name is also displayed in the list pane which will allow future modifications.

• *Reservoir:* As in the phase item above, a click on this item will generate a dialogue box where the name of the reservoir should be defined. Then, an ellipse to indicate the reservoir is displayed in the position where the left mouse button has been released.

• *Process:* A process is represented by a rectangle. Therefore, it requires two points as references in the pane. The operation to draw a process starts when the left button of the mouse is pressed. This point will correspond to one of the corners of the rectangle. Holding down the mouse button, drag the cursor to another point and release the button. The program will consider this second point as the other corner of the rectangle. Then, the user is prompted to define the process indicating the name and the characteristics he wants to impose. The system automatically detects if the process definition corresponds in fact to a vessel based on the attributes which have been activated. In addition, an automatic procedure detects which phases are included in the process.

• *Connection:* Selection of this item will prompt users to define the name of the connection, the two elements interconnected, the sampling mechanism if it applies, and the transfer mechanism. The two interconnected elements must exist in the system. As a result, a solid or a dashed line will be drawn on the screen.

• *Reaction:* Selection of this item will prompt users to define a reaction where a name, the stoichiometric reaction, and a model to evaluate the reaction are specified. If the reaction is heterogeneous then a solid line will be drawn on the screen to interconnect the phases involved.

• *Controller:* Selection of this item will prompt users to define the controller according to the language provided in chapter three. As a result, a triangle is drawn where the left mouse button was released and dot lines will be drawn on the screen connecting the entities related to the input(s) and output(s).

An interactive mechanism to detect some errors is also implemented. For instance: a message is displayed indicating a mistake if either of the two interconnected elements does not exist, the name of the entities must start with a capital letter, etc.

Omissions of properties of phases are not considered errors. Thus, if the program detects that a transfer mechanism requires the value of a property which has not

been activated, the program will incorporate and activate the property into the model.

The list pane also includes another menu. This menu is activated by clicking on the list pane with the left mouse button. Then, the menu shown in Figure 5.5 will show up. The user can activate any of the items of this menu by releasing the left mouse button on the item of interest. The result is explained below:

- *Definition:* Activating this item, the definition of the selected entity is displayed and the user can modify the contents.

- *Coordinates:* This item is used to modify the point of display of the selected entity.

- *Remove:* The selected entity will be removed if the deletion is confirmed.

The prototype described in this section is in fact an environment for aiding the engineer with process model-building. The high degree of flexibility provided by the graphical user interface is such that users can describe the modelling system starting from the basic entities: phases and reservoirs, or starting from global views: processes. In addition, the graphical user interface also provides rapid access to the topology of all parts of the system facilitating the examining and probably the understanding of the functionality and behaviour of the system.

Once the user believes that the information about the system is completely established, the generation of the model is reduced to a simple click on the proper item of the menu "MoPS". Then, the description is automatically translated into the object-oriented environment and the procedure to generate the model, already discussed in the chapter four, is performed.

The actual interface, as presented on the computer screen, is given in Figure 5.6. Some entities have also been defined to show the appearance of the entities.

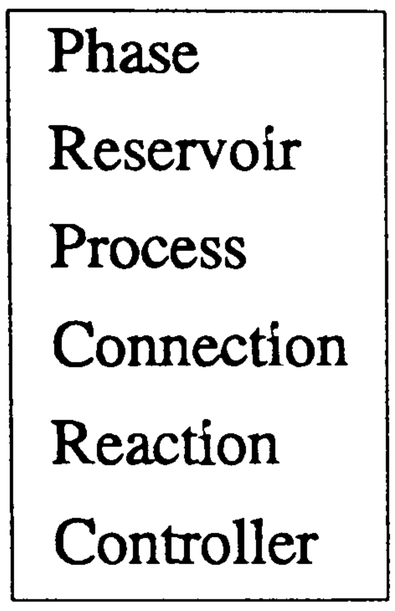


Figure 5.4 *The Menu Assigned to the Graphical Pane*

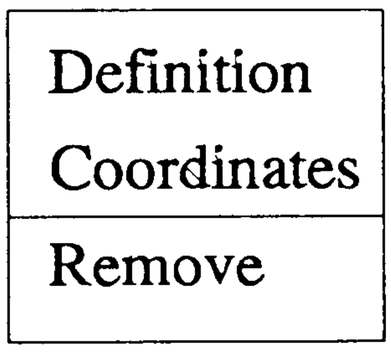


Figure 5.5. The Menu Assigned to the List Pane

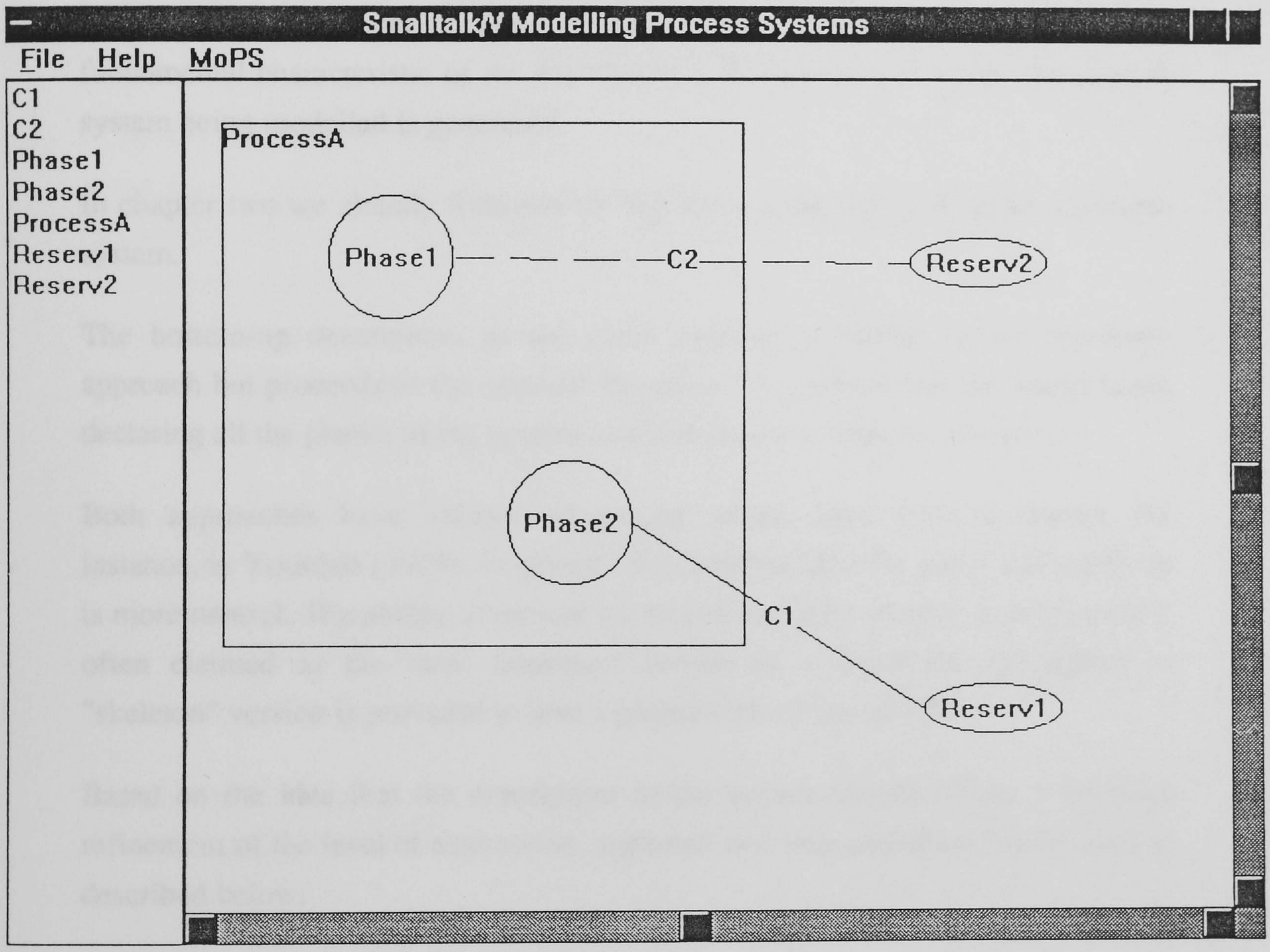


Figure 5.6 *The Actual Interface*

5.5.- A Procedure to Generate the System Representation

The process of modelling might start by placing a symbol, as a block diagram, on the drawing space which is the portion of the window that is reserved for the layout of the symbols, the connections, etc. This drawing space is of virtually unlimited size.

In general, there are two structured ways to describe a process, they are called the top-down and the bottom-up approaches. In practice, however, the user can make a choice between them or even consider mixed or random approaches. The fundamental characteristic of the description is the sequence in which the process system being modelled is generated.

In chapter two we already followed the top-down approach to describe a process system.

The bottom-up description, as the name implies, is similar to the top-down approach but proceeds in the opposite direction. This means that we would begin declaring all the phases of the system and then continue with the processes.

Both approaches have various advantages which have been discussed, for instance, in Yourdon [1979]. In general, it is believed that the top-down approach is more natural. The ability to present users with an early version of the system is often claimed as the most important benefit of a top-down description; a "skeleton" version is provided to give a global view of the system.

Based on the idea that the description of the system should follow a stepwise refinement of the level of abstraction, a general two-step procedure can be used as described below:

<u>First Step</u>

- Draw, in the proper position, the rectangles representing the processes, subprocesses, and vessels. Then, provide the declarative representation to generate the properties involved in each entity.

- Identify the phases in each process and define them inside the respective process if they are a part of any of the processes. Declare the properties of interest for modelling.

- Identify and declare the reservoirs.

Second Step

- Identify the connections and declare them with the sampling mechanism and the transfer model.

- Incorporate the declaration of controllers.

- Incorporate the declaration of reactions.

In general, a different order of declaration yields the same structure. However, connections, reactions, and controllers cannot be declared if any of the involved entities has not been declared. Indeed, a condition imposed by the system is that connections, reactions, and controllers will exist provided the involved phases and/or processes exist in the system. Thus, whenever the removal of one entity is confirmed, all its related connections, reactions, and controllers will also be removed.

Once an entity is declared it remains in the system unless it is explicitly removed by the user. The sizes of the symbols are automatically evaluated and the programming environment shows every entity as soon as it is declared. The position and the definition itself can be changed by selecting the proper item in the list pane.

Though the package is in fact under development, the essential idea has been implemented. The graphic representation provides the user with a visual picture of the process system. All the examples given in the appendix show the window which contains the declared entities in order to demonstrate the versatility of the prototype.

5.6.- The Internal Structure of the Prototype

As a result of the interaction with the user, the system generates sets which contain the data related to the phases, reservoirs, processes, reactions, controllers, and connections, as well as the names of the different chemical species of the system being modelled. Then, this information is translated into an object-

oriented structure so that each entity provided by the user is internally manipulated as an object which is an instance of one of the previously defined classes. In principle, the user could execute all these messages himself; however, this activity could be tedious.

Both, the translation into the object-oriented structure and the generation of the mathematical model are automatically performed by clicking on the appropriate item of the menu shown in Figure 5.4.b. The actual codified method is called "*model*" and contains, in essence, the computing program of Procedure 4.1. The translation into the internal structure is performed by the method *translate* of the class **ProcessSystem** which is activated by *model*.

The various classes which have been identified during the development of the prototype are shown in Figure 5.7. These classes form a hierarchy where the root class **Object** provides the common behaviour for all objects. They are discussed in more detail below.

Phase

This class defines the protocol for generating the data structure of phases. The properties of interest for the model are contained in its defined instance variables.

Reservoir

This class contains the variables related to the reservoirs of the process system being modelled.

Connection

The purpose of this class is to provide a common protocol for the connections. Elements are not added or removed as instances of this class.

SpecifiedEnergy, HeatConduction, HeatConvection, HeatRadiation, BulkFlow, Equilibration, TurbulentFlow, LaminarFlow, TwoFilmModel,and **PressureDropDriven**

These classes are subclasses of the class **Connection** and provide the protocol for the different kinds of connections.

Process

This class provides the data structure for processes.

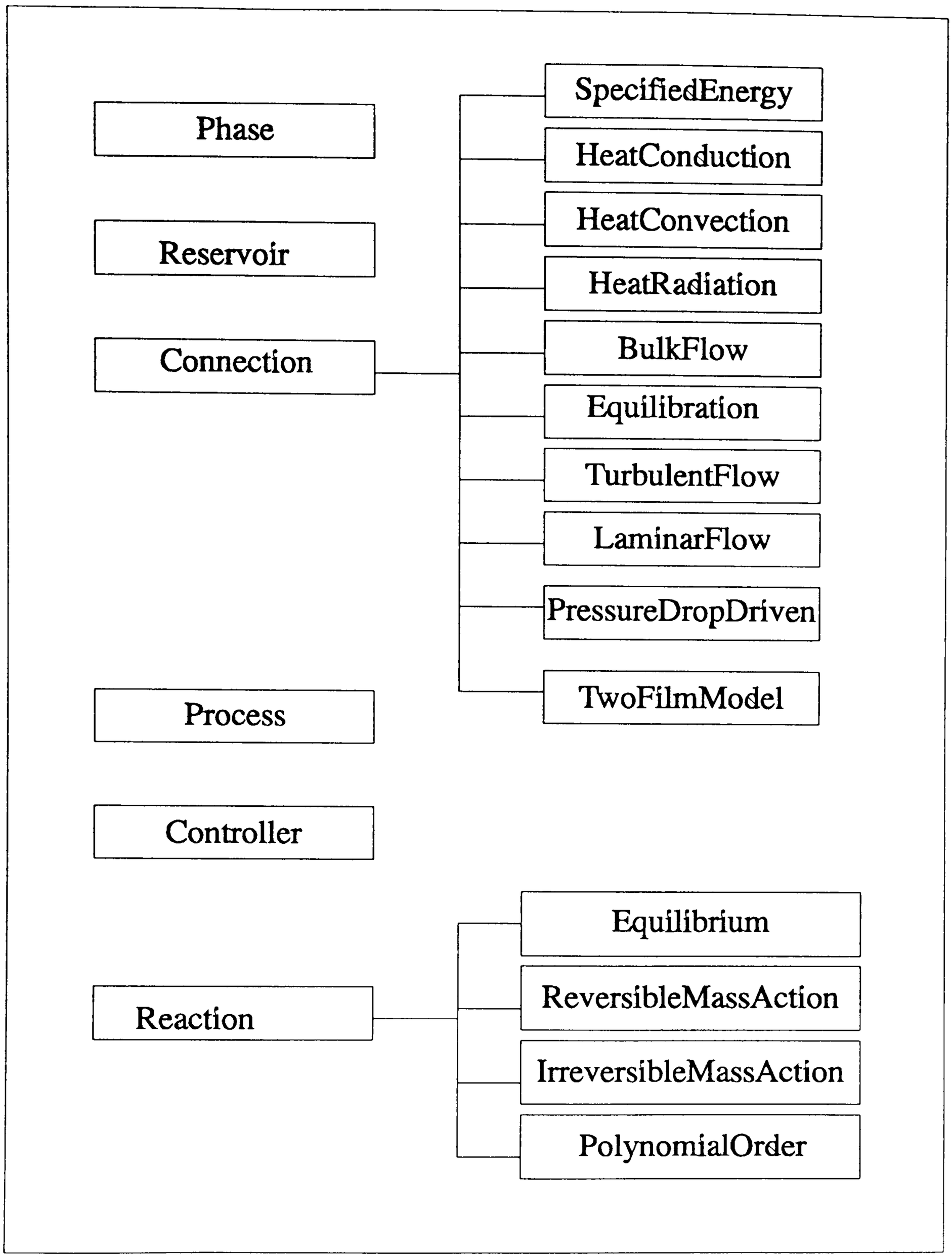


Figure 5.7 *The Hierarchy of the Classes in the Prototype*

Controller

This class provides the protocol to generate the procedures related to the controllers of the process system.

Reaction

This class provides the common protocol for reactions.

Equilibrium, ReversibleMassAction, IrreversibleMassAction, PolynomialOrder

These classes are subclasses of class **Reaction** and provide the protocol to generate the rate transfer equations.

It is obvious that the classes above defined do not include all the cases which may appear in process systems (e.g. a different kind of reaction); however, we believe that all the primitive classes have been incorporated. This means that any additional class will be a subclass of the existing classes. In fact, we have detected only three feasible points for expanding the capabilities of the prototype. These points concern the inclusion of subclasses of the classes connection, controller, and reaction. Thus, the prototype seems to be flexible in the sense that it can include additions/modifications without affecting the general structure.

In Smalltalk, the creation of an object is achieved by sending a message to a class. A common route used to create instances of the classes in Figure 5.7 is the evaluation of the message *translate: aString*. This method extracts from *aString* the name of the object and the instance variables declared by the user. For instance, the phase *V1* in the example A.1 of the appendix is created as an instance of the class **Phase** by evaluating the following message:

Phase translate:'V1= {P, V, T}'

The above message creates *V1* as an instance of the class **Phase** and it also activates the indicated instance variables.

Each class responds in a different way to the *translate* message. While the class **Phase** straightforwardly create an instance of its class, classes **Reaction** and **Connection** create instances of their subclasses. For instance, the connection C1

for the case study A.1 of the appendix is created as an object by sending the following message:

Connection translate:'C1= {R1,S1;-phaseSelective;TurbulentFlow}'

The above message creates an object called *C1* as an instance of the class **TurbulentFlow**.

The objects thus created inherit all the instance variables and respond to those methods defined in their classes and superclasses. Indeed, the prototype provides single inheritance, which means that a class has only one superclass. Those instance variables which are not activated remain initialised to the object **nil**. The object **nil** is the sole instance of class **UndefinedObject** and it is always assigned as a value of the instance variables of all new objects.

The user does not explicitly declare the entities grouped by a process since the interface provides the view of these entities. Then, the method called *getEntities: aProcess* is performed for each process in order to detect and incorporate this information into the appropriate objects.

The objects created as instances of the classes shown in Figure 5.7 are contained in so called global variables. These objects are accessible to any other object via previously defined messages.

In order to automate the translation, all the Smalltalk source code is incorporated into a temporal instance of the class **ReadWriteStream** which is subsequently executed.

Once the entities have been translated, the method *model* of the class **ProcessSystem** continues its execution according to the steps indicated in Procedure 4.1. The regions are detected via the message *getRegions* which, in essence, contains the Smalltalk code of Procedure 4.2.

The equations are written into the global variable *Model* which is an instance of the class **ReadWriteStream**. This variable is automatically initialised each time a new model is developed.

The first set of equations is generated by sending the following message to each region:

massBalance: species with: connections with: reactions

where *species* is a string containing the chemical species of the system.

The above method has been programmed for phases and processes and it works as indicated in Procedure 4.3. In order to incorporate the rate transfer terms, step 1.2.1 of Procedure 4.3, this method sends the following message to each connection:

aConnection connectMassIn: aRegion

The rate transfer is then incorporated, provided *aConnection* connects *aRegion*. The search includes looking into each of the phases contained in the region.

Similarly, the reaction terms are added to the model, provided a reaction occurs in one of the phases of the region. The following method has been programmed:

aReaction reactsPhase: aPhase with: species

The energy balance equations are generated by sending the following message to each region:

energyBalance: species with: connections

The above method contains in fact the Smalltalk code of Procedure 4.4. It sends the following message to each connection to incorporate the transfer terms:

aConnection connectEnergyIn: aRegion

The rate transfer is then incorporated provided *aConnection* relates to a phase contained in the region.

All the differential equations of the model will be completely generated by the end of this step. Then, we move on to the generation of the transfer rate characterising equations. We start by getting the equations related to the reactions. The method which has been programmed in each subclass of the class **Reaction** (i.e. **Equilibrium, ReversibleMassAction**) has the same name:

model: species

Thus, every reaction as an object responds to the same message above, but generates a different equation.

The different connections also respond to the same message in their own way to generate the transfer rate equations. For instance, the transfer rate equation for the connection *C1* is generated by the following message:

C1 model: species

The following step, according to Procedure 4.1, consists of incorporating the group of those equations from constraints imposed on the phases and those equations where the mass fractions are introduced. The following methods achieve these goals:

getFixedValues
fractionMol

The first method is activated when instances of the class **Phase** or **Process** receive the message. The second method is used for each phase and region contained in the system being modelled.

Finally, the set of procedures is generated as follows: the procedures for control are generated by sending the message *model:species* to each instance of the class **Controller**; the flash procedures are generated by sending the message *getFlash:*

species to each region; and the properties procedures are generated by sending the message *getProperties: species* to each phase.

A point worth mentioning here is that the method *getProperties: species* performs a search in the mathematical model (the global variable *Model*) to detect the properties that the model requires. In addition, it looks into the properties which have been activated as a request of the user.

It might have been noticed, and confirmed in the examples of the appendix, that the model generation follows a sequential order, as given in Procedure 4.1.

5.7.- Some Additional Issues of the Model Generation

The management and control of names for regions and variables in the modelling package is also considered the responsibility of the computer program. Thus, a reasonable standard has been developed. As pointed out by Simonyi and Heller [1991], it may help in the development and maintenance of the software.

The question arising here is: What is the form used to represent a mathematical expression? The answer to this question is addressed in the following discussion.

It is well known that mathematical expressions contain symbols which represent mathematical operations and variables. The type of representation used for algebraic expressions affects significantly the required computation time and storage [Pantelides, 1988]. Unfortunately, there are not many possibilities of representation in computer interfaces. Computers allow only expressions written in the form of a string.

A string is a finite-length sequence of symbols from a given alphabet [Aho et al, 1974]. An empty string is one with no symbols. In this study strings will be enclosed in single quotes.

Three types of mathematical representation have been distinguished in a string: prefix, infix, and suffix [Rosen, 1991]. If x and y are two variables, and z is a mathematical operator, then the string xzy is an infix notation, the string zxy is a prefix notation, and the string xyz is a suffix notation. The prefix form is said to be in Polish notation while the suffix form is known as reverse Polish notation.

The properties of the three notations have been discussed by Hernández-Sosa [1980]. The prefix or suffix notation seems to improve the symbolic manipulation in the numerical solution of the mathematical expression. However, in this work we will use the infix notation to describe the mathematical model because of the familiarity of this form in the engineering community.

One characteristic of the infix notation is that it requires parentheses to represent mathematical expressions. Let us consider the mathematical expression below,

$$\frac{(x+y)^2}{z} \tag{5.1}$$

Then, it is represented in infix notation as:

'(x + y) ** 2 / z' (5.2.)

Note that if we remove the parentheses in (5.2) the expression will not represent (5.1).

Once the form of the string has been established, the following question concerns the name of the variables. One important standard to be enforced is the use of abbreviations. The objectives for abbreviating the words in a name are to produce shorter meaningful names, and to provide easy look-up in a model. Additional effort must be made to prevent synonyms and homonyms. Appropriate names are important for the successful generation of the model. In particular, duplication of names must not be allowed. Thus, names of variables must be unique, descriptive, and understandable. Generally, it has been considered poor practice to abbreviate a string of words to only one letter of the word describing an attribute. However, exceptions to this rule are widely known.

Variables describing physical properties of the phases are named using both roots: the name of the entity and the name of the attribute. For convenience, we follow the inverse order, i.e. the first part of the name identifies the attribute and the second part identifies the name of the entity. Recently, this form has been named the Hungarian convention [Simonyi and Heller, 1991]. Abbreviations for the attribute's name follow the symbols given in the Table 2.1 complemented by the Table 5.1. For instance, the variable required to identify the pressure in the region

named "Liquid" is "PLiquid", and the variable "kVapour" stands for the thermal conductivity in the region named "Vapour".

The symbol "£" has been selected to represent, in a string form, the mathematical operator of the derivative d/dt.

Those variables associated with connections, controllers, reactions, and reservoirs are named in a similar form to phases. Table 5.2 gives the symbols referring to those attributes characterising the sort of connections included in this study.

Finally, the expressions generated may include vector representation. If a variable represents a vector but the operation is carried out with only one element of the vector, the number of the variable in the array is indicated between parentheses immediately after the name of the variables. For instance, x(2) indicates the second element in the array x. A vector with its elements is indicated by writing its elements, separated by a comma, between parentheses. For instance, the vector x= (1.5, 2.0, 3.4)' contains three elements with corresponding values x(1)= 1.5, x(2)= 2., and x(3)= 3.4.

5.8.- Summary

This chapter has structured the modelling approach presented in the previous chapters into the object-oriented approach. The graphic interface was also described in detail.

It was satisfying to realise that the graphical description of a process is an implementable issue. It is believed that graphical interfaces achieve the high priority goal of producing user-friendly programs.

Our experience indicates positive results in terms of speed when prototyping in Smalltalk. However, disasters happen very often without any apparent reason, making the development more difficult.

Though the present status of the code is just the initial step to produce a simulation package, it may be attractive to develop software for the numerical solution of the mathematical model within the object-oriented technique.

Property	Symbol
Rate of Reaction	r
Internal Energy	U
Mass	m
Constants of reaction	K1, K2
Mol fraction	xm

Table 5.1 Complementary Symbols of Phase Attributes

Property	Symbol
Mass flowrate	f
Energy flowrate	e
Heat transfer coefficient	h
Area of the interface	A
Total flow	w
Parameter associated with a valve	C
Parameter of pressure drop driven flow	E
Boltzman and emissivity constant	Ce
Mass Transfer Coefficient	Mt

Table 5.2 Typical Set of Symbols of Variables Related to Connections

CHAPTER SIX

GENERAL CONCLUSIONS AND FUTURE WORK

"Certainly I owe you an explanation,
and you shall have it. But you will permit
me to handle the matter in my own way"

A.Conan Doyle, The case-book of S Holmes

6.1.- Conclusions

In this thesis, a systematic approach to the modelling of lumped parameter processes and its implementation as a computer-aided system have been presented. We have demonstrated that it is possible to automate the build-up of dynamic mathematical models based on the description of the process system in terms of more elementary operations, without considering any particular geometry of the units being modelled.

Two main general parts have been identified: the description of the problem and the build-up of the model. In the particular case of lumped parameter systems, it was possible to keep the separation between these two parts during all the modelling process.

An effort was made to find a process decomposition convenient for both the user and the model-building process. Thus, the proposed decomposition of the process system, based on the phases and supported by the underlying principles of thermodynamics, provides a convenient way to achieve our goals of modelling. The accurate description of the process in physical terms is considered the responsibility of the user.

The definition language developed uses a keyword-based representation which permits the description of the process system in a rather simplified way. The language is specifically oriented to process systems. The process description for a given plant can be easily created, maintained, and modified using the language. The utility of the language has been demonstrated by the wide range of case studies given in the appendix.

The definition language presents a clear evolution from "unit operations" to "elementary physical and chemical phenomena" where the user employs terms which lie close to the normal terminology customary in the discussion of these specific topics.

Once the process system has been described, the problem of modelling was reduced to the utilization of the conservation laws, coupled with the laws dictating the rates of transfer, together with the relations imposed via processes. It has been demonstrated that the degree of automation of the model build-up is such that the construction of the model can be performed by just a "click" on the appropriate item.

Insights into some aspects of the numerical solution were considered necessary prior to the model generation in order to avoid poorly formulated models. Unfortunately, some of the inconsistencies of models cannot be prevented at this stage since they appear as a result of selection of the free variables, e.g. the high index problem.

A second area investigated concerned the programming language which could make use of the more detailed and general representation of the process system. Thus, we found that the object-oriented technique fits well the approach for process model-building presented in this thesis. In particular, it was observed that the language used, Smalltalk, provides an excellent environment for prototyping. Indeed, it was satisfying to verify that the graphical representation proposed in this thesis can be implemented into a computer code. The implementation demonstrates the benefits of the object-oriented technique in both programming language and design.

It was also observed that graphic environments provide a rigorous and coherent conceptual framework for a new generation of modelling systems.

6.2.- Future Work

We consider that the present approach is a potentially powerful technique for the automatic generation of process models based on the description of simpler operations. However, many issues remain to be resolved for modelling. For

instance, the work of connections between processes composed of sub-processes as explained in Section 3.6.1 has yet to be done.

The user interface developed for the prototype was sufficient for testing and development purposes. However, it is likely that more study of graphic interfaces will result in better environments.

There exist specific models which have been broadly tested which the user may want to incorporate in a simpler way. Therefore, tools should be developed that permit the introduction, in a simple way, of externally developed models.

While the prototype seems to achieve our goals of modelling, the numerical solution of the equations has not yet been implemented. One of the various activities which are required for making the prototype a real package for simulation is the development of a language for describing the operations for simulation, or the adaptation of the system into one of the already existing simulation packages. For instance, it may be a good idea to combine the present system with the language for simulation proposed by Barton [1992].

Finally, the problem of modelling distributed parameter systems remains as a challenge.

REFERENCES

"I could perhaps suggest that the set should be valued by an expert"

A.Conan Doyle, The case-book of S Holmes

A.V. Aho, J.E. Hopcroft, and J.D. Ullman (1974). "The Design and Analysis of Computer Algorithms", Addison-Wesley Pub. Co.

E.M. Allen, R.H. Trigg, and R.J. Wood (1983). "The Maryland Artificial Intelligence Group Franz Lisp Environment", TR-1226, University of Maryland, College Park, Maryland.

M. Andersson (1990). "Omola- An Object-oriented Language for Model Representation", Department of Automatic Control, Lund Institute of Technology, Lund Sweden.

R. Balzhiser, and M.R. Samuels (1977). "Engineering Thermodynamics", Prentice-Hall Inc.

P. Barton (1992). "The Modelling and Simulation of Combined Discrete/ Continuous Processes", PhD thesis on preparation, University of London.

R.B. Bird, W.E. Stewart, and E.N. Lightfoot (1960). "Transport Phenomena", John Wiley & Sons.

K.E. Brenan, S.L. Campbell, and L.R. Petzold (1989). "Numerical Solution of Initial-value Problems in Differential-algebraic Equations", North-Holland.

British Standard (1977). "Graphical Symbols for General Engineering", part 1, "Piping Systems and Plant", BS 1553.

G.F. Butler and M.J. Corbin (1990). "Object-oriented Simulation in Fortran-77", RAE Working Paper MM 388/89, Listing Revised: March 1990.

I.T. Cameron (1983). "Large Scale Transient Analysis of Processes-The State of the Art-", Rev.Latinoam.Ing.Quim.Quim.Apl., vol 13, pp. 215-228.

J.M. Coulson, J.F. Richardson, J.R. Backhurst and J.H. Harker (1990). "Chemical Engineering, Fluid Flow, Heat Transfer and Mass Transfer", vol. 1, fourth edition, Pergamon Press.

J.A. De Leeuw Den Bouter, F.A. Perris, D.W.T. Rippin, and F.A.P. Theunissen (1983). "Sample Problems for Flowsheeting Packages", Comp.Chem.Engng, vol 7, pp. 65-71.

D.J. Deutsch (1980). "Process Piping Systems", McGraw-Hill Pub. Co.

L.P. Deutsch and A. Goldberg (1991). "Smalltalk Yesterday, Today and Tomorrow", Byte, August 1991, pp 108-115.

I.S. Duff (1981). "On algorithms for Obtaining a Maximum Transversal", ACM Transactions on Mathematical Software, Vol. 7, No. 3, September.

J. Fabre and A. Liné (1992). "Modelling of Two-phase Slug Flow", in: J.L. Lumley, M.V. Dyke, and H.L. Reed (Eds), "Annual Review of Fluid Mechanics", Vol. 24, Annual Reviews Inc.

M. Feinberg (1979). "On Gibbs' Phase Rule", Archive for Rational Mechanics and Analysis, vol. 70, pp. 219-234.

G.F. Froment and K.B. Bischoff (1979). "Chemical Reactor Analysis and Design", John Wiley & Sons.

C.W. Gear (1988). "Differential-algebraic Equations Index Transformations", SIAM J. Sci. Stat. Comput, vol 9, No. 1, pp. 39-47.

J.W. Gibbs (1878). "On the Equilibrium of Heterogeneous Substances", Trans. Conn. Acad., 3, pp 343-524. In "The Scientific Papers of J.W. Gibbs", vol 1, New York, Dover Publications 1961.

E. Hairer, C. Lubich, and M. Roche (1989). "The Numerical Solution of Differential-algebraic Systems by Runge-Kutta Methods", Lecture Notes in Mathematics edited by A. Dold, B. Eckmann, and F. Takens, No. 1409, Springer-Verlag.

R. Hernández-Sosa (1980). "Symbolic Algebraic Techniques in Computer-aided Process Design", PhD Thesis, University of London, Imperial College.

D.M. Himmelblau, and K.B. Bischoff (1968). "Process Analysis and Simulation: deterministic systems", John Wiley&Sons.

J.O. Hirshfelder, C.F. Curtiss, and R.B. Bird (1954). "Molecular Theory of Gases and Liquids", New York, Wiley.

J.E. Hopcroft, and R.M. Karp (1973). "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs", SIAM J. Comput., vol 2, No. 4, pp. 225-231.

R. Jarvis (1992). Private communication.

R. Krishna, and G.L. Standart (1976). "A Multicomponent Film Model Incorporating a General Matrix Method of Solution to the Maxwell-Stefan Equations", AIChE Journal, vol 22, No. 2, pp. 383-389.

R. Krishna, and G.L. Standart (1979). "Mass and Energy Transfer in Multicomponent Systems", Chem. Eng.Commun., vol 3, pp. 201-275.

R. Krishna, and R. Taylor (1986). "Multicomponent Mass Transfer: Theory and Applications", in N.P. Cheremisinoff (Ed), 1986, "Handbook of Heat and Mass Transfer", Vol 2, pp. 259-432.

R. Krishnamurthy, and R. Taylor (1985). "A Nonequilibrium Stage Model of Multicomponent Separation Processes Part I: Model Description and Method of Solution", AIChE Journal, vol 31, No. 3, pp 449-455.

R. Krishnamurthy, and R. Taylor (1985a). "A Nonequilibrium Stage Model of Multicomponent Separation Processes Part II: Comparison with Experiment", AIChE Journal, vol 31, No. 3, pp 456-465.

A. Kroner, P. Holl, W. Marquardt, and E.D. Gilles (1990). "Diva-An Open Architecture for Dynamic Simulation", Computers Chem. Engng, vol 14, No. 11, pp. 1289-1295.

W.H. McAdams (1958). "Heat Transmission", McGraw-Hill.

W. Marquardt (1991). "Dynamic Process Simulation- Recent Progress and Future Challenges", in Y. Arkun and W.H. Ray (Eds). "Chemical Process Control-CPCIV", CACHE AIChE, Proceeding of the Fourth International Conference on Chemical Process Control.

B. Meyssami, and O.A. Asbjornsen (1989). "Process Modelling from First Principles- Method and Automation", Proc. 1989 Summer Computer Simulation Conference, July 24-27, Austin, Texas, USA.

J. Micallef (1988). "Encapsulation, Reusability and Extensibility in Object-oriented Programming Languages", Journal of Object-oriented Programming, vol 1, pp. 12-51.

M. Modell and R.C. Reid (1983). "Thermodynamics and its Applications", Second Ed, Prentice-Hall.

M.N. Ozisik (1985). "Heat Transfer A Basic Approach"; Mc Graw-Hill Book Co.

C.C. Pantelides (1988). "SpeedUp-Recent Advances in Process Simulation", Comput. Chem. Engng, vol 12, No. 7, pp. 745-755.

C.C. Pantelides (1988a). "Symbolic and Numerical Techniques for the Solution of Large Systems of Nonlinear Algebraic Equations", PhD Thesis, University of London, Imperial College.

C.C. Pantelides (1989). "Advanced Techniques for Computer-aided Design and Operation of Chemical Processes, An Intensive Course", Universidad de Chile.

G.A. Pascoe (1986). "Elements of Object-oriented Programming", Byte, August, pp. 139-144.

J.D. Perkins (1983). "Equation-oriented Flowsheeting", in A.W. Westerberg and H.H. Chien (Eds), "Proceedings of the Second International Conference on Foundations of Computer-aided Process Design", Colorado, June 19-24, 1983, pp. 309-367.

J.D. Perkins (1986). "Survey of Existing Systems for the Dynamic Simulation of Industrial Processes", Modeling, Identification and Control, vol 7, No. 2, pp. 71-81.

J.D. Perkins, and G.W. Barton (1987). "Modelling and Simulation in Process Operation", Foundations of Computer Aided Process Design, CACHE Publications.

R. Perry, and C.H. Chilton (1974). "Chemical Engineers'Handbook", fifth ed., Mc Graw-Hill Book Co.

P.C. Piela (1989). "ASCEND: An Object-oriented Computer Environment for Modelling and Analysis", PhD Thesis, Carnegie-Mellon University.

P.C. Piela, T.G. Epperly, K.M. Westerberg and A.W. Westerberg (1991). "Ascend: an Object-oriented Computer Environment for Modeling and Analysis: The Modeling Language", Computers Chem.Engng, vol 15, No.1, pp. 53-72.

J.W. Ponton, and P.J. Gawthrop (1991). "Systematic Construction of Dynamic Models for Phase Equilibrium Processes", Computers Chem. Engng, vol 15, No. 12, pp. 803-808.

H.A. Preisig, T.Y. Lee, F. Little, and B. Wright (1989). "On the Representation of Life-support System Models", 19th Intersociety Conference on Environmental Systems, San Diego, California, July 24-26.

H.A. Preisig, T.Y. Lee, and F. Little (1990). "A Prototype Computer-aided Modelling Tool for Life-support System Models", 20th International Conference on Environmental Systems, July 9-12, Williamsburg, VA.

W.F. Ramirez (1989). "Computational Methods for Process Simulation", Butterworths Series in Chemical Engineering.

G.V. Reklaitis (1983). "Introduction to Material and Energy Balances", John Wiley & Sons.

E.M. Rosen (1980). "Steady State Chemical Process Simulation: A State-Of-The-Art Review"; in R.G. Squires and G.V. Reklaitis (Eds), "Computer Applications to Chemical Engineering"; ACS Symposium Series.

K.H. Rosen (1991). "Discrete Mathematics and its Applications", Second Edition, Mc Graw-Hill.

R.W.H. Sargent (1978). "The Decomposition of Systems of Procedures and Algebraic Equations", in G.A. Watson (Ed) Lecture Notes in Mathematics (630), Proceedings, Biennial Conference, Dundee 1977, Springer-Verlag, pp. 158-178

R.W.H. Sargent (1983). "Advances in Modelling and Analysis of Chemical Process Systems", Computers and Chemical Engng, Vol. 7, No. 4, pp. 219-237.

R.W.H. Sargent (1989). "Distillation", Lecture Notes, Imperial College.

R.W.H. Sargent, and A.W. Westerberg (1964). "SpeedUp in Chemical Engineering Design", Trans. Instn.Chem.Engrs, Vol 42, pp. T190-T197.

J.H. Saunders (1989). "A Survey of Object-oriented Programming Languages", JOOP, March/April, pp. 5-11

C. Simonyi and M. Heller (1991). "The Hungarian Revolution", Byte, August, pp. 131-138.

Smalltalk/V Windows (1991). Tutorial and Programming Handbook, Digitalk.

D.B. Spalding, and E.H. Cole (1973). "Engineering Thermodynamics. An Introductory Text", Edward Arnold Ltd.

G. Stephanopoulos, J. Johnston, T. Kriticos, R. Lakshmanan, M. Mavrovouniotis, and C. Siletti (1987). "Design-kit: an Object-oriented Environment for Process Engineering", Comput. Chem. Engng, Vol 11, No. 6, pp. 655-674.

G. Stephanopoulos, G. Henning, and H. Leone (1990). "Model.LA. A Modeling Language for Process Engineering-I. The Formal Framework", Computers Chem. Engng, vol 14, No. 8, pp. 813-846.

G. Stephanopoulos, G. Henning, and H. Leone (1990a). "Model.LA. A Modeling Language for Process Engineering-II. Multifaceted Modeling of Processing Systems", Computers Chem. Engng, Vol 14, No. 8, pp. 847-869.

D.V. Steward (1965). "Partitioning and Tearing Systems of Equations", J.SIAM Numer. Anal., Ser. B, vol 2, No. 2, pp. 345-365.

B. Stroustrup (1988). "What is Object-oriented Programming?", IEEE Software, May, pp. 10-20.

M. Wolczko (1990). "A Comparison of Object-oriented Languages and Environments", A Short Course on "Object-oriented Techniques for Engineers" EASE, University of Manchester.

E. Yourdon and L. Constantine (1979). "Structured Design: Fundamentals of a Discipline of Computer Program and System Design", Prentice-Hall.

APPENDIX: Experiments on Process Modelling

"The trouble with doing something right the first time is that nobody appreciates how difficult it was"

Sun WorkStation

This appendix contains a set of examples which have been solved using the prototype described in chapter five. These examples are intended to demonstrate the versatility of the approach presented in the thesis and its implementation in a computer code.

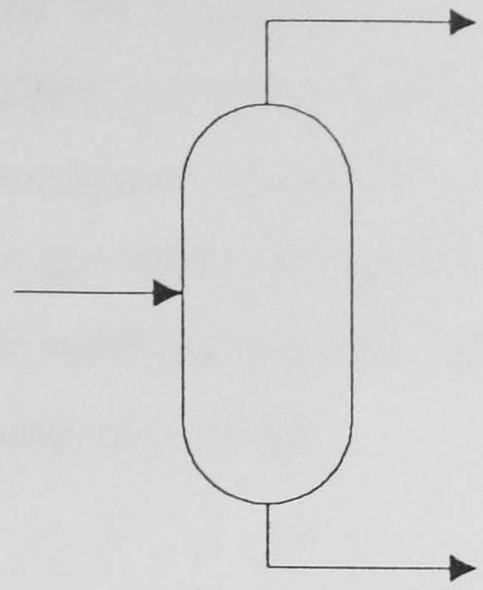
Two types of examples are given, the first type corresponds to the development of models for new units and the second to complete flowsheets. With regards to the details of the solution, a set of free variables is proposed in each example in order to show the possibility of solving the mathematical model. The numerical solution of the examples is not trivial but it is beyond the goals of this research.

The declarative representation has been included for each example.

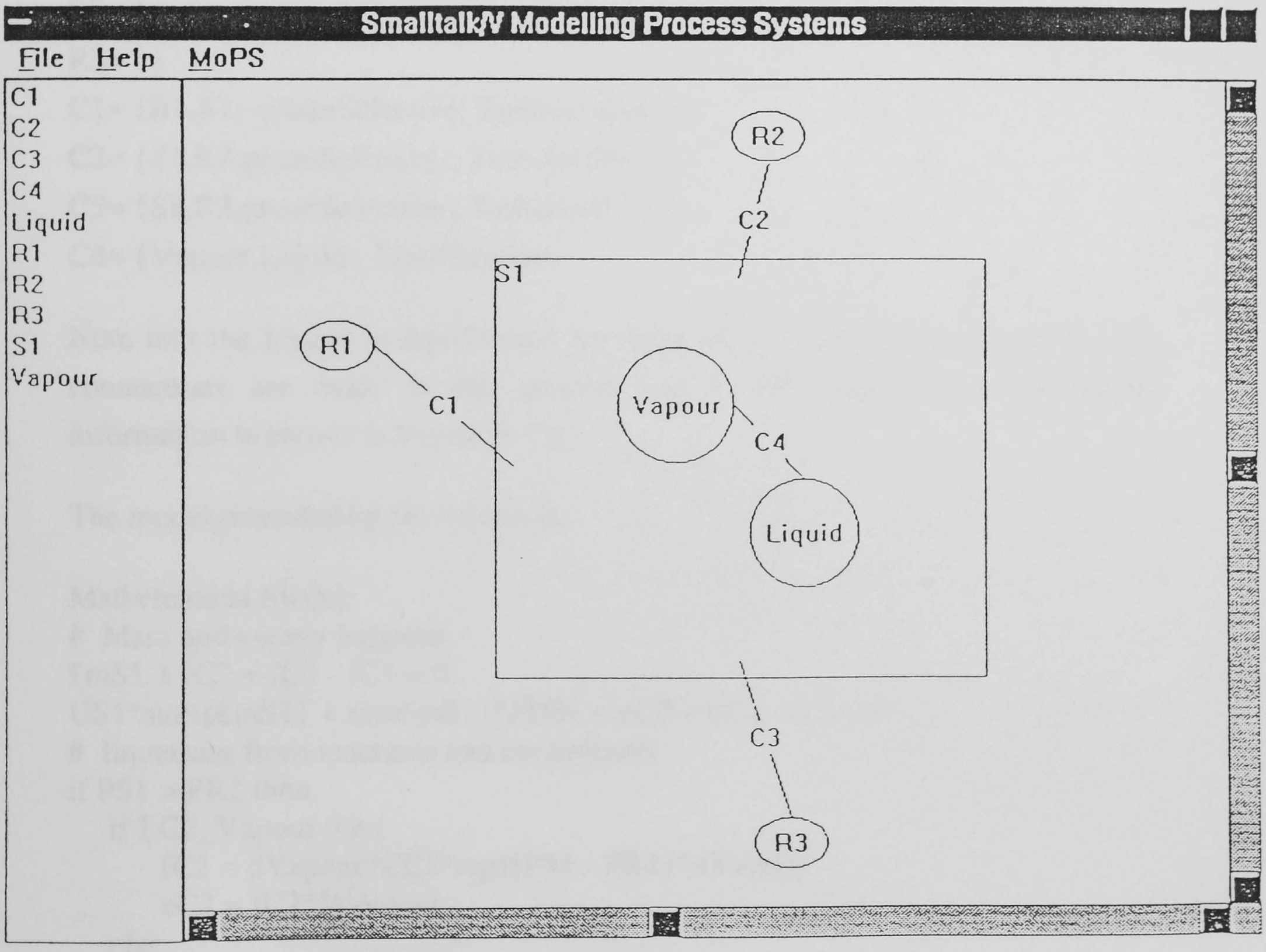
A.I- Modelling Units

Example A.1 *Flash Tank:*

This section starts with one of the simplest units often found in process systems: the flash tank. Under ordinary operation a steady flow is fed into the tank where two phases, liquid and vapour, may coexist and stream out from the top and bottom of the tank as seen in Figure A.1.a. Modelling this simple tank becomes complex if we consider some of the extreme operation conditions. For instance, the inlet flow can be decreased/increased at certain times or its condition modified; then, it may happen that either of the two phases disappears. In that condition, both outputs will have the same phase.



(a)



(b)

Figure A.1 *Modelling a Flash Tank*

In order to model this system we suppose that each stream has turbulent flow. According to the model described in chapter four, reverse flow is not allowed, i.e. check valves make this assumption feasible. We also assume four chemical species {A B C D} which are declared via the chemical species item as described in chapter five. Thus, the following declarative statements have been incorporated through the activation of appropriate items:

```
Vapour= {P, T, V}
Liquid= {P, T, V}
S1= {Vapour, Liquid; sumV, P}
R1= {}
R2= {}
R3= {}
C1= {R1,S1; -phaseSelective; TurbulentFlow}
C2= {S1,R2;phaseSelective-; TurbulentFlow}
C3= {S1,R3;phaseSelective-; TurbulentFlow}
C4= {Vapour,Liquid;; Equilibration}
```

Note that the phases in equilibrium are declared as a process (S1) and that the connections are made to this process, not to the phases. The topological information is shown in Figure A.1.b.

The model provided by the system is,

```
Mathematical Model:
#  Mass and energy balances
£mS1 + fC2 + fC3 - fC1 = 0
US1*sum{£mS1} + sum{mS1}*£US1 + eC2 + eC3 - eC1 = 0
#  Equations from reactions and connections
if PS1 > PR2 then
    if LC2_Vapour then
        fC2 = dVapour*CC2*sqrt{PS1 - PR2}*xVapour
        eC2 = fC2*HVapour
    else
        fC2 = dLiquid*CC2*sqrt{PS1 - PR2}*xLiquid
        eC2 = fC2*HLiquid
    endif
else
    fC2 = 0
    eC2 = 0
endif
if PS1 > PR3 then
```

```
    if LC3_Vapour then
        fC3 = dVapour*CC3*sqrt{PS1 - PR3}*xVapour
        eC3 = fC3*HVapour
    else
        fC3 = dLiquid*CC3*sqrt{PS1 - PR3}*xLiquid
        eC3 = fC3*HLiquid
    endif
else
    fC3 = 0
    eC3 = 0
endif
if PR1 > PS1 then
    fC1 = dR1*CC1*sqrt{PR1 - PS1}*xR1
    eC1 = sum{fC1}*HR1
else
    fC1 = 0
    eC1 = 0
endif
#  Additional equations
VVapour + VLiquid = VS1
VVapour*dVapour = sum{mVapour}
VLiquid*dLiquid = sum{mLiquid}
mVapour= xVapour*sum{mVapour}
mS1 = xS1*sum{mS1}
mLiquid= xLiquid*sum{mLiquid}
#  Procedures for flashes, controllers and properties
flash{input: mS1, US1, PS1, output: TS1, sVapour, mVapour, UVapour,
        PVapour, TVapour, dVapour, sLiquid, mLiquid, ULiquid, PLiquid,
        TLiquid, dLiquid}
propH{input: xVapour, UVapour, PVapour; output: HVapour}
propH{input: xLiquid, ULiquid, PLiquid; output: HLiquid}
propd{input: xR1, UR1, PR1; output: dR1}
propH{input: xR1, UR1, PR1; output: HR1}
```

Nomenclature:
CC1 is parameter of flowrate in C1
CC2 is parameter of flowrate in C2
CC3 is parameter of flowrate in C3
dLiquid is density in Liquid
dR1 is density in R1
dVapour is density in Vapour
eC1 is transfer energy in C1
eC2 is transfer energy in C2
eC3 is transfer energy in C3
fC1 is nc-vector of mass transfer rate in C1
fC2 is nc-vector of mass transfer rate in C2
fC3 is nc-vector of mass transfer rate in C3

HLiquid is specific mass enthalpy in Liquid
HR1 is specific mass enthalpy in R1
HVapour is specific mass enthalpy in Vapour
LC2_Vapour is a logical variable related to C2_Vapour
LC3_Vapour is a logical variable related to C3_Vapour
mLiquid is nc-vector of mass in Liquid
mS1 is nc-vector of mass in S1
mVapour is nc-vector of mass in Vapour
PLiquid is pressure in Liquid
PR1 is pressure in R1
PR2 is pressure in R2
PR3 is pressure in R3
PS1 is pressure in S1
PVapour is pressure in Vapour
sLiquid is status (V, L, or S) of phase Liquid
sVapour is status (V, L, or S) of phase Vapour
TLiquid is temperature in Liquid
TS1 is temperature in S1
TVapour is temperature in Vapour
ULiquid is internal energy in Liquid
UR1 is internal energy in R1
US1 is internal energy in S1
UVapour is internal energy in Vapour
VLiquid is volume in Liquid
VS1 is volume in S1
VVapour is volume in Vapour
xLiquid is nc-vector of mass fraction in Liquid
xR1 is nc-vector of mass fraction in R1
xS1 is nc-vector of mass fraction in S1
xVapour is nc-vector of mass fraction in Vapour

The System has:
chemical species: A B C D
58 equations
72 variables
14 degrees of freedom and
requires 5 initial conditions

The simulation problem can be solved by fixing the following set of

Free Variables:

CC1	CC2	
CC3	xR1	*
dR1	PR1	
PR2	VS1	

PR3 LC2_Vapour
LC3_Vapour

* The variable is a nc-vector

Note that logical variables appear in both the mathematical model and the set of free variables. At the moment, it is expected that the user will be able to relate this variable to some condition of the process. Observe that the vapour phase will stream out if the logical condition is satisfied. The condition, for instance, may be: LC2_Vapour= (sum{mVapour} > *0*).

Example A.2 *A Boiling Vessel-Case I:*

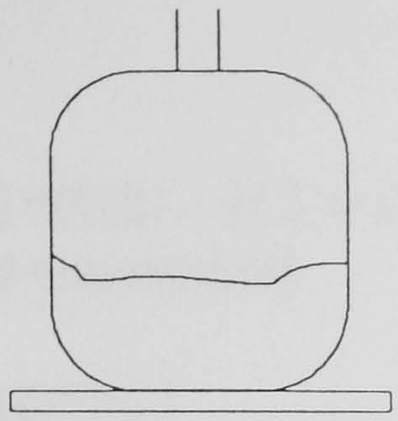
Here we consider a liquid to be evaporating in a closed vessel as shown in Figure A.2.a. As boiling proceeds, the composition of the liquid and the pressure in the vessel changes but the volume occupied by the two phases is constant. It seems appropriate to consider that the phases within the vessel coexist in equilibrium. The vapour phase is the only phase streaming out and it is considered as a laminar flow.

For modelling purposes we consider four chemical species: {A B C D}. The declarative statements are as follows:

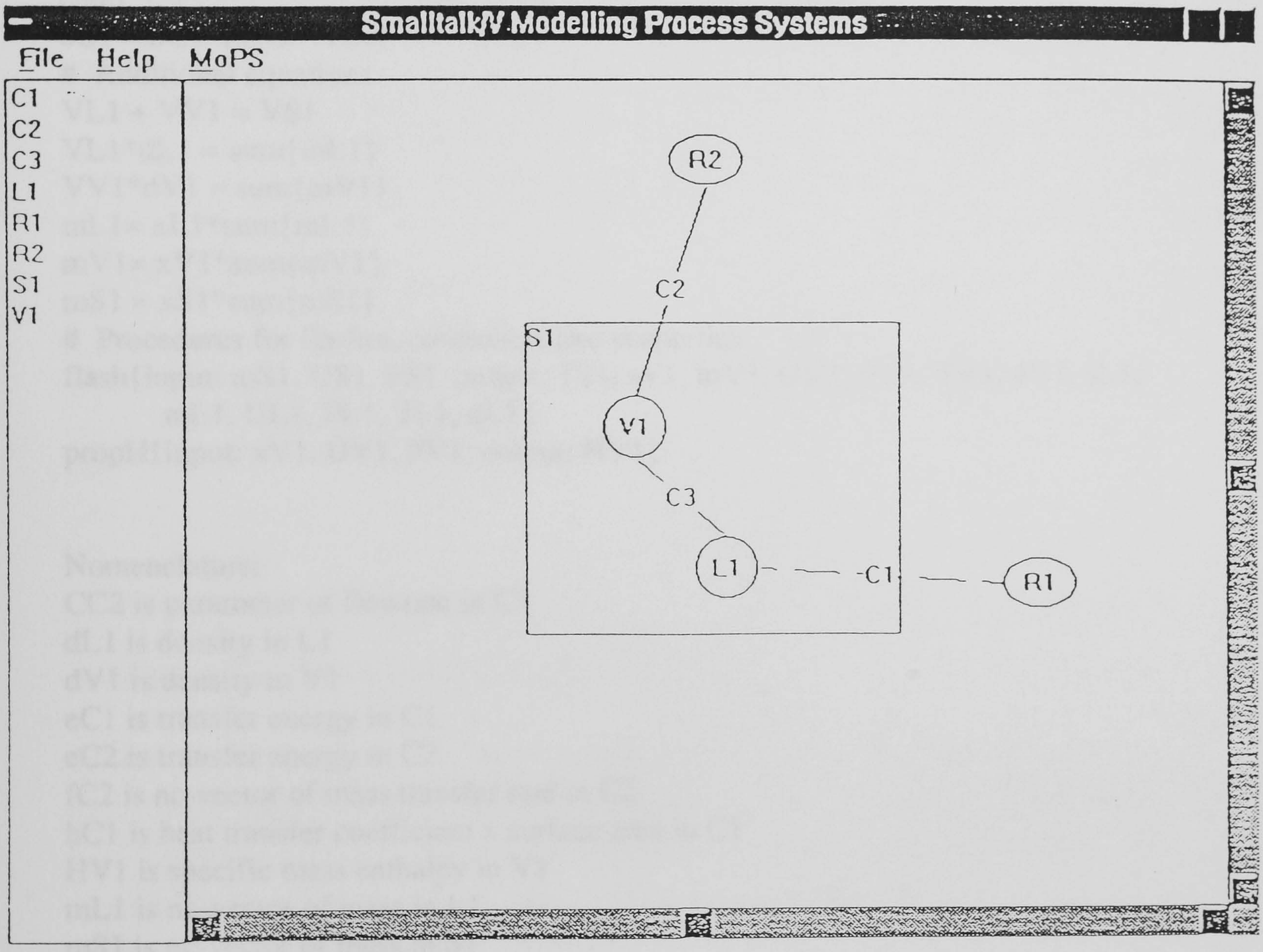
V1= {P, T, V}
L1= {P, T, V}
R1= { }
R2= { }
S1= {V1, L1; sumV, P}
C1= {R1, L1;; HeatConvection}
C2= {V1, R2; ;LaminarFlow}
C3= {L1, V1;; Equilibration}

The topological information is shown in Figure A.2.b.

The solution provided by the system is,



(a)



(b)

Figure A.2 *Boiling Vessel Case I*

Mathematical Model:
Mass and energy balances
£mS1 + fC2 = 0
US1*sum{£mS1} + sum{mS1}*£US1 - eC1 + eC2 = 0
# Equations from reactions and connections
if PV1 > PR2 then
    fC2 = dV1*CC2*(PV1 - PR2)*xV1
    eC2 = sum{fC2}*HV1
else
    fC2 = 0
    eC2 = 0
endif
eC1 = hC1*(TR1 - TL1)
# Additional equations
VL1 + VV1 = VS1
VL1*dL1 = sum{mL1}
VV1*dV1 = sum{mV1}
mL1= xL1*sum{mL1}
mV1= xV1*sum{mV1}
mS1 = xS1*sum{mS1}
# Procedures for flashes, controllers and properties
flash{input: mS1, US1, PS1, output: TS1, sV1, mV1, UV1, PV1, TV1, dV1, sL1,
        mL1, UL1, PL1, TL1, dL1}
propH{input: xV1, UV1, PV1; output: HV1}

Nomenclature:
CC2 is parameter of flowrate in C2
dL1 is density in L1
dV1 is density in V1
eC1 is transfer energy in C1
eC2 is transfer energy in C2
fC2 is nc-vector of mass transfer rate in C2
hC1 is heat transfer coefficient x surface area in C1
HV1 is specific mass enthalpy in V1
mL1 is nc-vector of mass in L1
mS1 is nc-vector of mass in S1
mV1 is nc-vector of mass in V1
PL1 is pressure in L1
PR2 is pressure in R2
PS1 is pressure in S1
PV1 is pressure in V1
sL1 is status (V, L, or S) of phase L1
sV1 is status (V, L, or S) of phase V1
TL1 is temperature in L1
TR1 is temperature in R1
TS1 is temperature in S1
TV1 is temperature in V1

UL1 is internal energy in L1
US1 is internal energy in S1
UV1 is internal energy in V1
VL1 is volume in L1
VS1 is volume in S1
VV1 is volume in V1
xL1 is nc-vector of mass fraction in L1
xS1 is nc-vector of mass fraction in S1
xV1 is nc-vector of mass fraction in V1

The System has:
 chemical species: A B C D
 46 equations
 51 variables
 5 degrees of freedom and
 requires 5 initial conditions

The proposed set of free variables is,

CC2
hC1
VS1
PR2
TR1

Example A.3 *A Boiling Vessel-Case II:*

As one increases the diameter of the vapour outlet pipe in the previous example, the pressure difference (PV1 - PR2) decreases to zero but the constant of the valve (CC2) goes to infinity to maintain a finite and continuous vaporisation. Here we consider the limiting case of an open vessel as shown in Figure A.3.

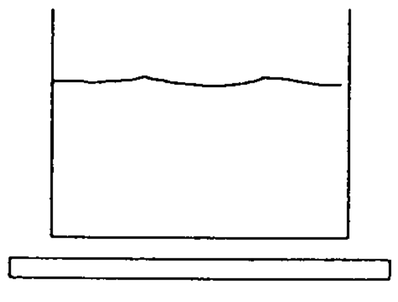


Figure A.3 *Boiling Vessel Case II*

The topological information is similar to the one given in Figure A.2.a. In this case, however, the pressure PV1 is essentially constant and equals to PR2. If we assume that there is sufficient rate of evaporation to make counter-diffusion of vapour from the reservoir negligible, then we can assume that there is a layer of vapour above the liquid in equilibrium with it, and C2 becomes a bulk flow to determine. In this case, the rate of evaporation will become independent of the volume of vapour so that we can specify a minimum value as given below.

For modelling purposes we consider four chemical species: {A B C D} and the declarative statements are as follows:

```
V1= {P, T, V= 1.e-03}
L1= {P, T, V}
S1= {V1, L1; P}
R1= {}
R2= {}
C1= {R1, L1;; HeatConvection}
C2= {V1,R2;; BulkFlow}
C3= {V1, L1;; Equilibration}
```

The solution provided by the system is,

Mathematical Model:

```
#  Mass and energy balances
£mS1 + fC2 = 0
US1*sum{£mS1} + sum{mS1}*£US1 - eC1 + eC2 = 0
#  Equations from reactions and connections
if wC2 > 0 then
    fC2 = wC2*xV1
    eC2 = sum{fC2}*HV1
else
    fC2 =  wC2*xR2
    eC2 = sum{fC2}*HR2
endif
eC1 = hC1*(TR1 - TL1)
#  Additional equations
VL1*dL1 = sum{mL1}
VV1*dV1 = sum{mV1}
VV1 = 1.e-03
mL1= xL1*sum{mL1}
mV1= xV1*sum{mV1}
```

```
mS1 = xS1*sum{mS1}
# Procedures for flashes, controllers and properties
flash{input: mS1, US1, PS1, output: TS1, sV1, mV1, UV1, PV1, TV1, dV1, sL1,
        mL1, UL1, PL1, TL1, dL1}
propH{input: xV1, UV1, PV1; output: HV1}
```

Nomenclature:
dL1 is density in L1
dV1 is density in V1
eC1 is transfer energy in C1
eC2 is transfer energy in C2
fC2 is nc-vector of mass transfer rate in C2
hC1 is heat transfer coefficient x surface area in C1
HR2 is specific mass enthalpy in R2
HV1 is specific mass enthalpy in V1
mL1 is nc-vector of mass in L1
mS1 is nc-vector of mass in S1
mV1 is nc-vector of mass in V1
PL1 is pressure in L1
PS1 is pressure in S1
PV1 is pressure in V1
sL1 is status (V, L, or S) of phase L1
sV1 is status (V, L, or S) of phase V1
TL1 is temperature in L1
TR1 is temperature in R1
TS1 is temperature in S1
TV1 is temperature in V1
UL1 is internal energy in L1
US1 is internal energy in S1
UV1 is internal energy in V1
VL1 is volume in L1
VV1 is volume in V1
wC2 is total rate of flow in C2
xL1 is nc-vector of mass fraction in L1
xR2 is nc-vector of mass fraction in R2
xS1 is nc-vector of mass fraction in S1
xV1 is nc-vector of mass fraction in V1

The System has:
 chemical species: A B C D
 46 equations
 54 variables
 8 degrees of freedom and
 requires 5 initial conditions

A proposed set of free variables to solve the system is,

hC1
HR2
TR1
PS1
xR2 *

* The variable is a nc-vector

Example A.4 *A Boiling Vessel-Case III:*

If the rate of evaporation is small then there exists counter-diffusion. A typical case is the evaporation of water from a lake to the atmosphere. In this case, the two-film model will be used to determine the rate of evaporation at the surface of the liquid. It is convenient to remind you that we use "<>" to indicate that the mathematical operation is on the first (nc-1) components of an nc-vector.

We also consider four chemical species, {A B C D}, and the declarative statements are as follows:

L1= {P, T, V}
R1= {}
R2= {}
C1= {R1, L1;; HeatConvection}
C2= {L1, R1;; TwoFilmModel}

The topological information is shown in Figure A.4.

The solution provided by the system is,

Mathematical Model:
Mass and energy balances
£mL1 + fC2 = 0
UL1*sum{£mL1} + sum{mL1}*£UL1 + eC2 - eC1 = 0
Equations from reactions and connections
M*xmL1 = xL1*sum{M*xmL1}
<NC2> = MtL1_C2*<(xmL1 - xiL1)> + sum{NC2}*<xmL1>
sum{xiL1} = 1
M*xmR2 = xR2*sum{M*xmR2}
<NC2> = MtR2_C2*<(xiR2 - xmR2)> + sum{NC2}*<xmR2>

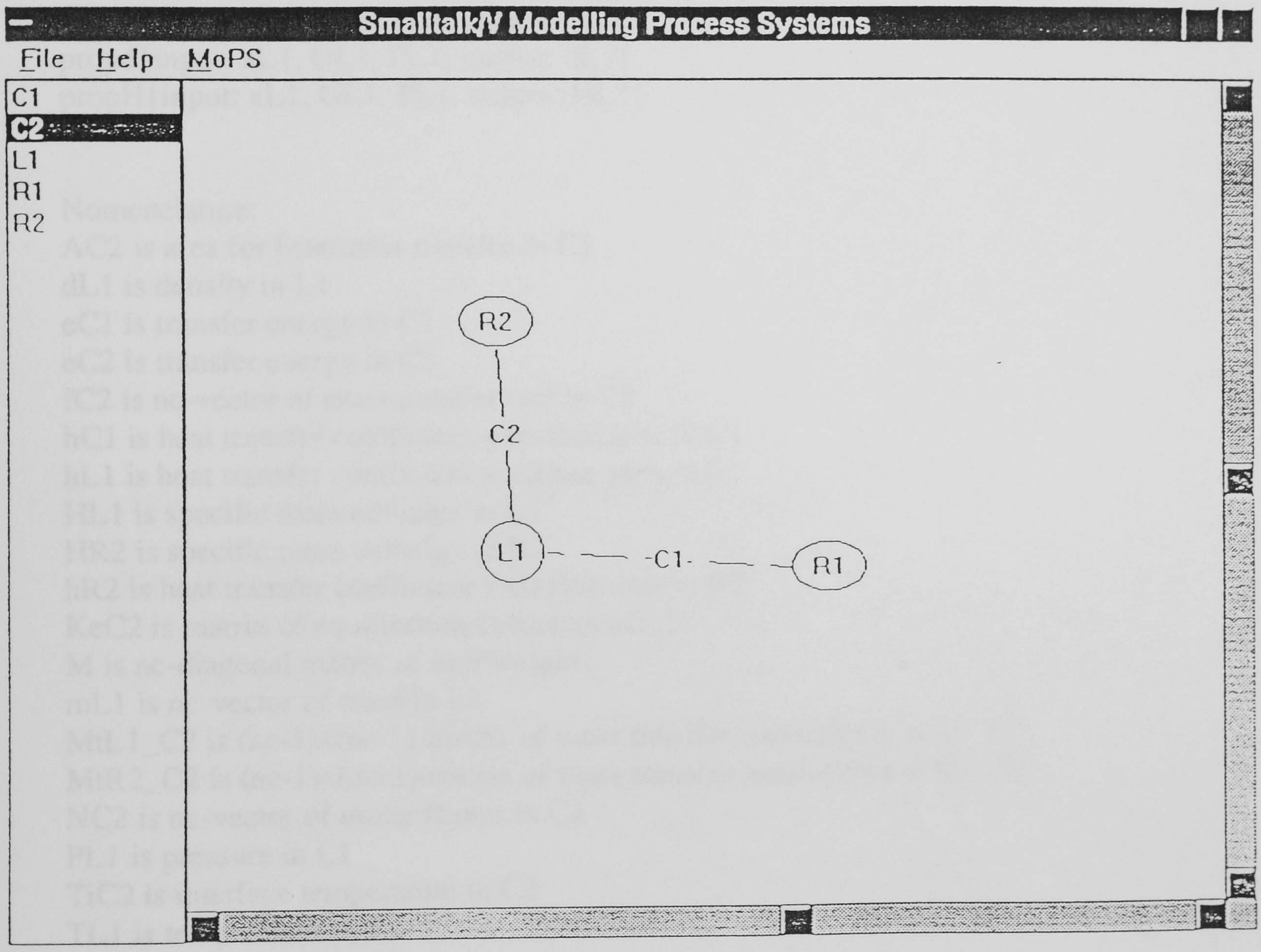


Figure A.4 *Boiling Vessel Case III*

```
sum{xiR2} = 1
fC2 = AC2*M*NC2
eC2 = hL1*(TiC2- TL1) + HL1*xL1'*fC2
eC2 = hR2*(TR2- TiC2) + HR2*xR2'*fC2
KeC2*xiL1 = xiR2
KeC2 = KeC2{PL1, TiC2, xiL1, xiR2}
eC1 = hC1*(TR1 - TL1)
#  Additional equations
VL1*dL1 = sum{mL1}
mL1= xL1*sum{mL1}
#  Procedures for flashes, controllers and properties
propT{input: xL1, UL1, PL1; output: TL1}
propd{input: xL1, UL1, PL1; output: dL1}
propH{input: xL1, UL1, PL1; output: HL1}
```

Nomenclature:
AC2 is area for heat/mass transfer in C2
dL1 is density in L1
eC1 is transfer energy in C1
eC2 is transfer energy in C2
fC2 is nc-vector of mass transfer rate in C2
hC1 is heat transfer coefficient x surface area in C1
hL1 is heat transfer coefficient x surface area in L1
HL1 is specific mass enthalpy in L1
HR2 is specific mass enthalpy in R2
hR2 is heat transfer coefficient x surface area in R2
KeC2 is matrix of equilibrium constants in C2
M is nc-diagonal matrix of mol weights
mL1 is nc-vector of mass in L1
MtL1_C2 is (nc-1)x(nc-1)-matrix of mass transfer coefficients in L1_C2
MtR2_C2 is (nc-1)x(nc-1)-matrix of mass transfer coefficients in R2_C2
NC2 is nc-vector of molar fluxes in C2
PL1 is pressure in L1
TiC2 is interface temperature in C2
TL1 is temperature in L1
TR1 is temperature in R1
TR2 is temperature in R2
UL1 is internal energy in L1
VL1 is volume in L1
xiL1 is nc-vector of interface mol fraction in L1
xiR2 is nc-vector of interface mol fraction in R2
xL1 is nc-vector of mass fraction in L1
xmL1 is nc-vector of mol fraction in L1
xmR2 is nc-vector of mol fraction in R2
xR2 is nc-vector of mass fraction in R2

The System has:
chemical species: A B C D
44 equations
78 variables
34 degrees of freedom and
requires 5 initial conditions

The proposed set of free variables is,

AC2	M **
PL1	TR1
hR2	hL1
xL1	PR2
TR2	xR2 *
MtL1_C2 ***	MtR2_C2 ***

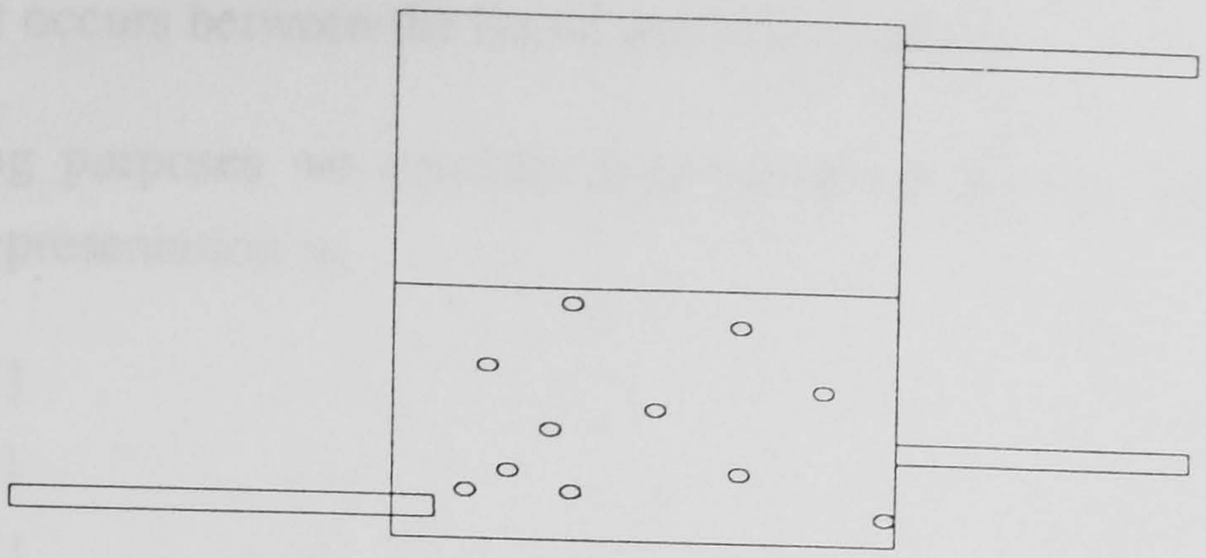
* The variable is a nc-vector

** The variable is a diagonal matrix

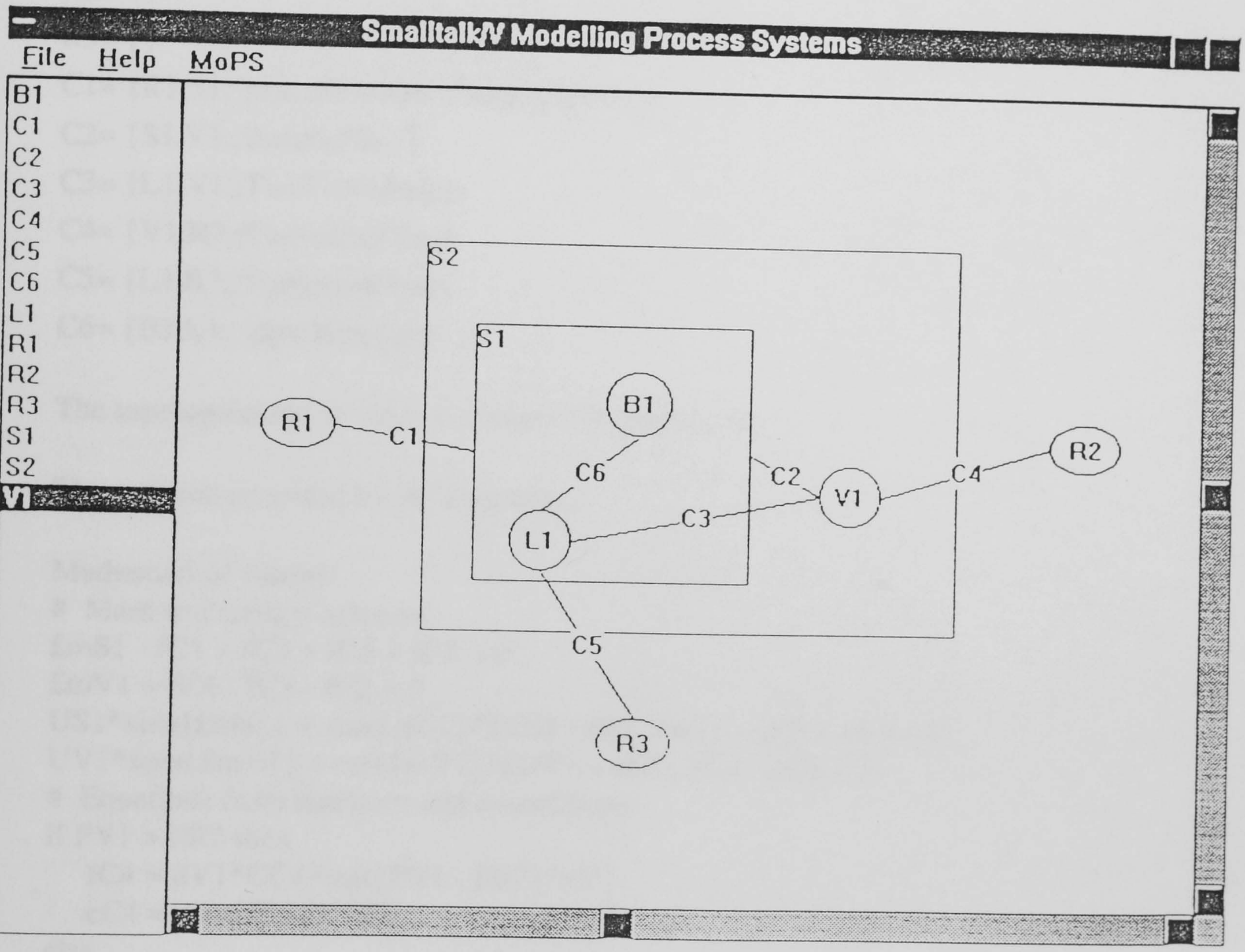
*** The variable is a (nc-1)*(nc-1)-matrix

Example A.5 *A Flash Vessel with Bubbles Streaming into it:*

In this example we show the definition of a process which contains another process. Here we consider that a stream bubbles into a vessel as shown in Figure A.5.a. Thus, three phases are contained in the vessel: the bubbles, the liquid phase, and the vapour phase. It is assumed that the bubbles and the liquid phase are in physical equilibrium. Then, the connection defined as C1 connects a mixture of bubbles and liquid which we define as process S1. We assume that none of the phases disappears and that we know a transfer law "BubbleFlow" which determines the rate of flow from the bubble phase to the vapour phase. The movement of the bubbles through the liquid phase is due to force acting on the bubbles. This force may come from the density difference between the bubbles and the liquid, and the Archimedes' principle may be used to relate the force to the mass of the fluid displaced by the particle and the acceleration from the external force. However, there are various effects which affect the practical results: the bubbles may change shape as they move through the liquid phase, wall effects, etc [Fabre and Liné, 1992]. Since the properties of both bubbles and liquid are required we consider that the bubble flow connection is from the process S1 to the vapour phase.



(a)



(b)

Figure A.5 *A Flash Vessel with Bubbles Streaming into it*

Mass transfer occurs between the liquid and vapour phases.

For modelling purposes we consider four chemical species: {A B C D}. The declarative representation is,

B1= {P, T, V}
L1= {P, T, V}
V1= {P, T, V}
S1= {B1, L1; }
S2= {V1, L1, B1; P, sumV}
R1= {}
R2= {}
R3= {}
C1= {R1,S1;-phaseSelective;TurbulentFlow}
C2= {S1,V1;;BubbleFlow}
C3= {L1,V1;;TwoFilmModel}
C4= {V1,R2;;TurbulentFlow}
C5= {L1,R3;;TurbulentFlow}
C6= {B1,L1;; Equilibration}

The topological information is shown in Figure A.5.b.

The solution provided by the system is,

Mathematical Model:
Mass and energy balances
£mS1 - fC1 + fC3 + fC5 + fC2 = 0
£mV1 + fC4 - fC3 - fC2 = 0
US1*sum{£mS1} + sum{mS1}*£US1 - eC1 + eC3 + eC5 + eC2 = 0
UV1*sum{£mV1} + sum{mV1}*£UV1 + eC4 - eC3 - eC2 = 0
# Equations from reactions and connections
if PV1 > PR2 then
    fC4 = dV1*CC4*sqrt{PV1 - PR2}*xV1
    eC4 = sum{fC4}*HV1
else
    fC4 = 0
    eC4 = 0
endif
M*xmL1 = xL1*sum{M*xmL1}
<NC3> = MtL1_C3*<(xmL1 - xiL1)> + sum{NC3}*<xmL1>
sum{xiL1} = 1
M*xmV1 = xV1*sum{M*xmV1}
<NC3> = MtV1_C3*<(xiV1 - xmV1)> + sum{NC3}*<xmV1>

```
sum{xiV1} = 1
fC3 = AC3*M*NC3
eC3 = hL1*(TiC3- TL1) + HL1*xL1'*fC3
eC3 = hV1*(TV1- TiC3) + HV1*xV1'*fC3
KeC3*xiL1 = xiV1
KeC3 = KeC3{PL1, TiC3, xiL1, xiV1}
fC2 = fC2{mB1, PB1, dB1, dL1, mL1}
eC2 = sum{fC2}*HB1
if PR1 > PS1 then
    fC1 = dR1*CC1*sqrt{PR1 - PS1}*xR1
    eC1 = sum{fC1}*HR1
else
    fC1 = 0
    eC1 = 0
endif
if PL1 > PR3 then
    fC5 = dL1*CC5*sqrt{PL1 - PR3}*xL1
    eC5 = sum{fC5}*HL1
else
    fC5 = 0
    eC5 = 0
endif
#  Additional equations
PS1 = PS2
PV1 = PS2
VV1 + VL1 + VB1 = VS2
VB1*dB1 = sum{mB1}
VL1*dL1 = sum{mL1}
VV1*dV1 = sum{mV1}
mS1 = xS1*sum{mS1}
mB1= xB1*sum{mB1}
mL1= xL1*sum{mL1}
mV1= xV1*sum{mV1}
#  Procedures for flashes, controllers and properties
flash{input: mS1, US1, PS1, output: TS1, sB1, mB1, UB1, PB1, TB1, dB1, sL1,
        mL1, UL1, PL1, TL1, dL1}
propH{input: xB1, UB1, PB1; output: HB1}
propH{input: xL1, UL1, PL1; output: HL1}
propT{input: xV1, UV1, PV1; output: TV1}
propd{input: xV1, UV1, PV1; output: dV1}
propH{input: xV1, UV1, PV1; output: HV1}
propd{input: xR1, UR1, PR1; output: dR1}
propH{input: xR1, UR1, PR1; output: HR1}
```

Nomenclature:
AC3 is area for heat/mass transfer in C3
CC1 is parameter of flowrate in C1

CC4 is parameter of flowrate in C4
CC5 is parameter of flowrate in C5
dB1 is density in B1
dL1 is density in L1
dR1 is density in R1
dV1 is density in V1
eC1 is transfer energy in C1
eC2 is transfer energy in C2
eC3 is transfer energy in C3
eC4 is transfer energy in C4
eC5 is transfer energy in C5
fC1 is nc-vector of mass transfer rate in C1
fC2 is nc-vector of mass transfer rate in C2
fC3 is nc-vector of mass transfer rate in C3
fC4 is nc-vector of mass transfer rate in C4
fC5 is nc-vector of mass transfer rate in C5
HB1 is specific mass enthalpy in B1
hL1 is heat transfer coefficient x surface area in L1
HL1 is specific mass enthalpy in L1
HR1 is specific mass enthalpy in R1
HV1 is specific mass enthalpy in V1
hV1 is heat transfer coefficient x surface area in V1
KeC3 is matrix of equilibrium constants in C3
M is nc-diagonal matrix of mol weights
mB1 is nc-vector of mass in B1
mL1 is nc-vector of mass in L1
mS1 is nc-vector of mass in S1
MtL1_C3 is (nc-1)x(nc-1)-matrix of mass transfer coefficients in L1_C3
MtV1_C3 is (nc-1)x(nc-1)-matrix of mass transfer coefficients in V1_C3
mV1 is nc-vector of mass in V1
NC3 is nc-vector of molar fluxes in C3
PB1 is pressure in B1
PL1 is pressure in L1
PR1 is pressure in R1
PR2 is pressure in R2
PR3 is pressure in R3
PS1 is pressure in S1
PS2 is pressure in S2
PV1 is pressure in V1
sB1 is status (V, L, or S) of phase B1
sL1 is status (V, L, or S) of phase L1
TB1 is temperature in B1
TiC3 is interface temperature in C3
TL1 is temperature in L1
TS1 is temperature in S1
TV1 is temperature in V1
UB1 is internal energy in B1
UL1 is internal energy in L1

UR1 is internal energy in R1
US1 is internal energy in S1
UV1 is internal energy in V1
VB1 is volume in B1
VL1 is volume in L1
VS2 is volume in S2
VV1 is volume in V1
xB1 is nc-vector of mass fraction in B1
xiL1 is nc-vector of interface mol fraction in L1
xiV1 is nc-vector of interface mol fraction in V1
xL1 is nc-vector of mass fraction in L1
xmL1 is nc-vector of mol fraction in L1
xmV1 is nc-vector of mol fraction in V1
xR1 is nc-vector of mass fraction in R1
xS1 is nc-vector of mass fraction in S1
xV1 is nc-vector of mass fraction in V1

The System has:
chemical species: A B C D
108 equations
145 variables
37 degrees of freedom and
requires 10 initial conditions

A solution to the above model can be obtained if the following variables are fixed:

CC1	CC4
CC5	PR3
PR1	PR2
AC3	hL1
hV1	dR1
xR1 *	VS2
MtL1_C3 **	MtV1_C3 **
M ***	

* The variable is a nc-vector

** The variable is a (nc-1)x(nc-1)-matrix

*** The variable is a diagonal matrix

Example A.6 *A Homogeneous Reactor:*

This example illustrates the capability of modelling a homogeneous reaction. It also introduces the declaration of a controller.

The model of a continuous stirred tank reactor is required for simulation. A stream feeds a mixture where the reactants "A" and "B" are provided. Then, the reaction

$$A + 2B = 2D$$

is carried out. The reaction is assumed second order with respect to chemical component B. Another stream is extracted from the reactor containing an output flow. The reactor exchanges heat in such a way that the temperature is controlled by manipulation of the stream into the coolant jacket as shown in Figure A.6.a.

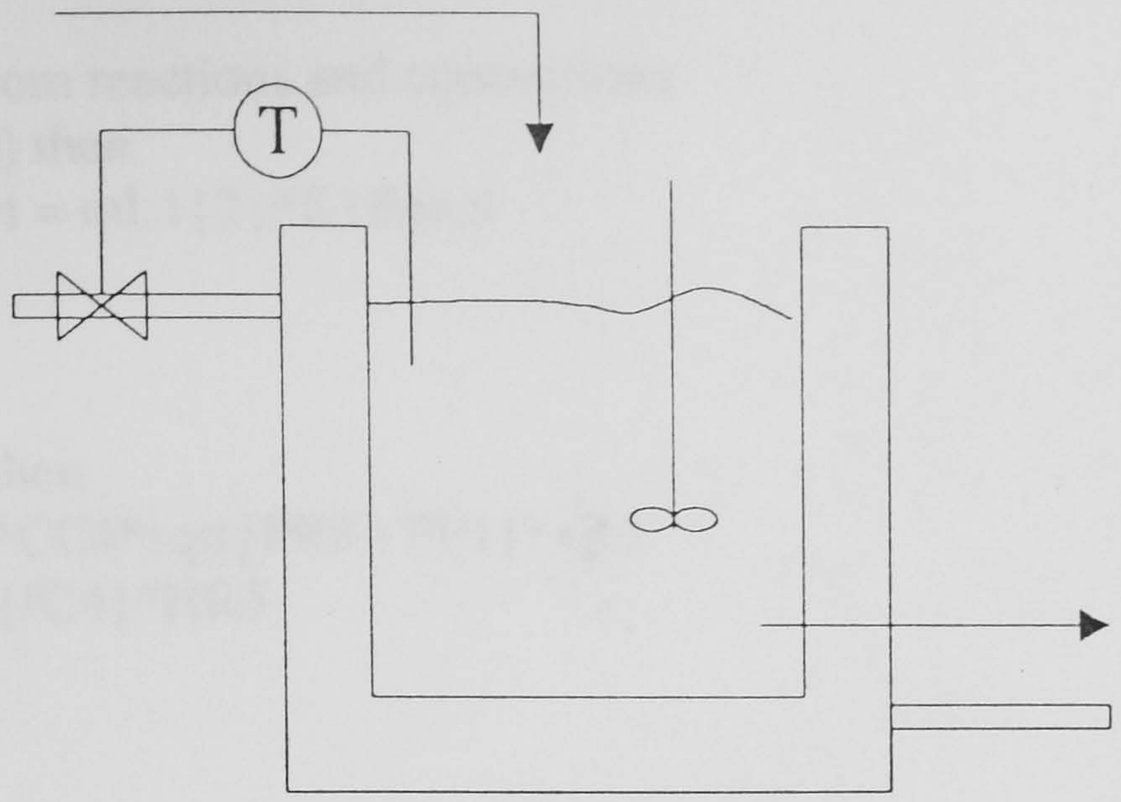
For modelling purposes we consider five chemical species: {A B C D E}. The declarative representation is as follows:

L1= {P, T, V}
P1= {P, T, V}
R1= { }
R2= { }
R3= { }
R4= { }
C1= {R1, L1;; TurbulentFlow}
C2= {L1, R2;; TurbulentFlow}
C3= {P1, L1; ; HeatConvection}
C4= {R3, P1;; TurbulentFlow}
C5= {P1, R4;; TurbulentFlow}
C6= { input: T in L1; output: C1}
React= { [A](L1)+2[B](L1)=2[D](L1); order_2_on_[B](L1)}

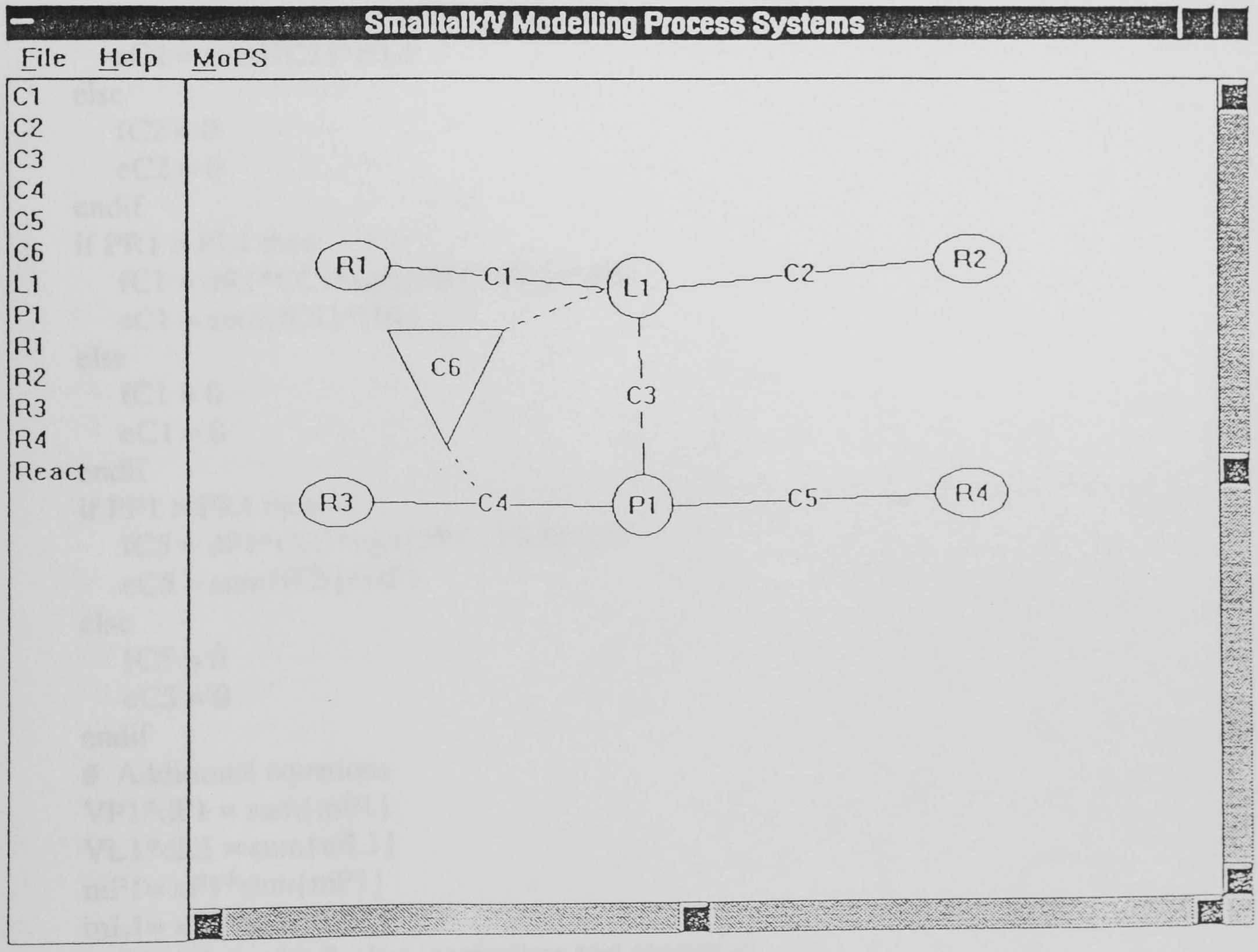
The topological information is shown in Figure A.6.b.

The solution provided by the system is,

Mathematical Model:
Mass and energy balances
£mP1 - fC4 + fC5 = 0
£mL1 + fC2 - fC1- rReact*M*(-1, -2, 0, 2)' = 0
UP1*sum{£mP1} + sum{mP1}*£UP1 - eC4 - eC3 + eC5 = 0
UL1*sum{£mL1} + sum{mL1}*£UL1 + eC3 + eC2 - eC1 = 0



(a)



(b)

Figure A.6 *A Homogeneous Reactor*

```
# Equations from reactions and connections
if (mL1{2}> 0) then
    VL1*rReact = mL1{2}*K1React
else
    rReact = 0
endif
if PR3 > PP1 then
    fC4 = dR3*CC4*sqrt{PR3 - PP1}*xR3
    eC4 = sum{fC4}*HR3
else
    fC4 = 0
    eC4 = 0
endif
eC3 = hC3*(TL1 - TP1)
if PL1 > PR2 then
    fC2 = dL1*CC2*sqrt{PL1 - PR2}*xL1
    eC2 = sum{fC2}*HL1
else
    fC2 = 0
    eC2 = 0
endif
if PR1 > PL1 then
    fC1 = dR1*CC1*sqrt{PR1 - PL1}*xR1
    eC1 = sum{fC1}*HR1
else
    fC1 = 0
    eC1 = 0
endif
if PP1 > PR4 then
    fC5 = dP1*CC5*sqrt{PP1 - PR4}*xP1
    eC5 = sum{fC5}*HP1
else
    fC5 = 0
    eC5 = 0
endif
# Additional equations
VP1*dP1 = sum{mP1}
VL1*dL1 = sum{mL1}
mP1= xP1*sum{mP1}
mL1= xL1*sum{mL1}
# Procedures for flashes, controllers and properties
control{input: TL1; output: CC4}
propT{input: xP1, UP1, PP1; output: TP1}
propd{input: xP1, UP1, PP1; output: dP1}
propH{input: xP1, UP1, PP1; output: HP1}
propT{input: xL1, UL1, PL1; output: TL1}
propd{input: xL1, UL1, PL1; output: dL1}
```

propH{input: xL1, UL1, PL1; output: HL1}
propd{input: xR3, UR3, PR3; output: dR3}
propH{input: xR3, UR3, PR3; output: HR3}
propd{input: xR1, UR1, PR1; output: dR1}
propH{input: xR1, UR1, PR1; output: HR1}

Nomenclature:
CC1 is parameter of flowrate in C1
CC2 is parameter of flowrate in C2
CC4 is parameter of flowrate in C4
CC5 is parameter of flowrate in C5
dL1 is density in L1
dP1 is density in P1
dR1 is density in R1
dR3 is density in R3
eC1 is transfer energy in C1
eC2 is transfer energy in C2
eC3 is transfer energy in C3
eC4 is transfer energy in C4
eC5 is transfer energy in C5
fC1 is nc-vector of mass transfer rate in C1
fC2 is nc-vector of mass transfer rate in C2
fC4 is nc-vector of mass transfer rate in C4
fC5 is nc-vector of mass transfer rate in C5
hC3 is heat transfer coefficient x surface area in C3
HL1 is specific mass enthalpy in L1
HP1 is specific mass enthalpy in P1
HR1 is specific mass enthalpy in R1
HR3 is specific mass enthalpy in R3
K1React is constant of reaction in React
M is nc-diagonal matrix of mol weights
mL1 is nc-vector of mass in L1
mP1 is nc-vector of mass in P1
PL1 is pressure in L1
PP1 is pressure in P1
PR1 is pressure in R1
PR2 is pressure in R2
PR3 is pressure in R3
PR4 is pressure in R4
rReact is rate of reaction in React
TL1 is temperature in L1
TP1 is temperature in P1
UL1 is internal energy in L1
UP1 is internal energy in P1
UR1 is internal energy in R1
UR3 is internal energy in R3
VL1 is volume in L1

VP1 is volume in P1
xL1 is nc-vector of mass fraction in L1
xP1 is nc-vector of mass fraction in P1
xR1 is nc-vector of mass fraction in R1
xR3 is nc-vector of mass fraction in R3

The System has:
chemical species: A B C D E
61 equations
89 variables
28 degrees of freedom and
requires 12 initial conditions

A suggested set of free variables is:

CC1	PR4
CC2	VL1
CC5	PR3
dR1	VP1
K1React	xR1 *
dR3	M **
PR1	xR3 *
hC3	PR2

* The variable is a nc-vector
** The variable is a diagonal matrix

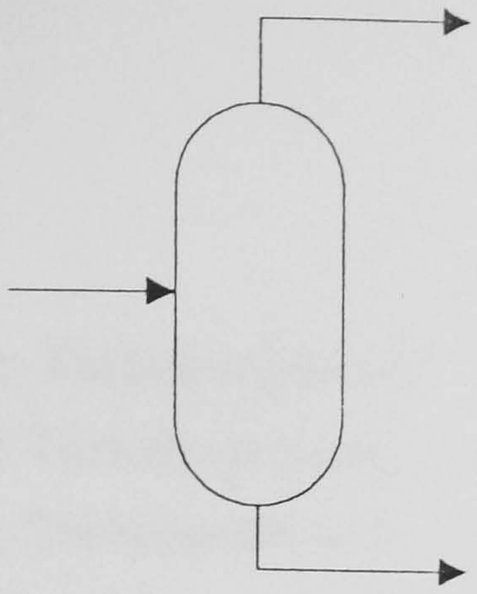
Example A.7: *A Heterogeneous Reactor:*

This example adds another characteristic to the flash tank shown in Figure A.7.a. We assume that the following heterogeneous reaction occurs within the tank:

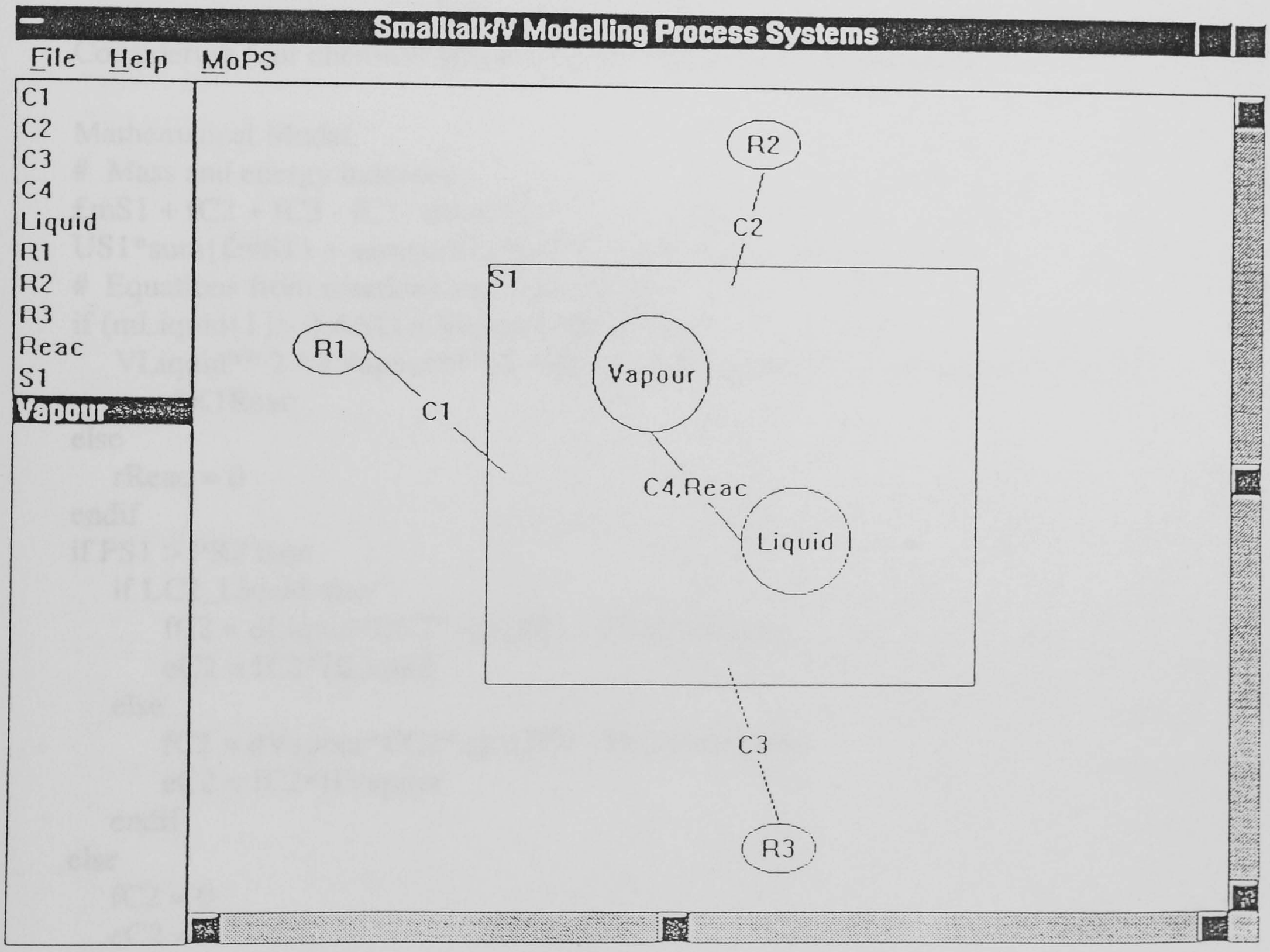
$$A_{(Liquid)} + B_{(Vapour)} = C_{(Liquid)}$$

The reaction is assumed to follow a kinetics of second order with respect to chemical component A and order 1/2 with respect to chemical component B. The declarative representation is as follows:

Vapour= {P, T, V}
Liquid= {P, T, V}
S1= {Vapour, Liquid; sumV, P}



(a)



(b)

Figure A.7 *A Heterogeneous Reactor*

R1= { }
R2= { }
R3= { }
C1= {R1,S1; -phaseSelective; TurbulentFlow}
C2= {S1,R2;phaseSelective-; TurbulentFlow}
C3= {S1,R3;phaseSelective-; TurbulentFlow}
C4= {Vapour,Liquid;; Equilibration}
Reac= {[A](Liquid) + [B](Vapour)=[C](Liquid);
order_2_on_[A](Liquid)&order_1/2_on_[B](Vapour) }

The topological representation is given in Figure A.7.b.

Considering four chemical species, {A B C D}, the model generated is:

Mathematical Model:

```
#  Mass and energy balances
£mS1 + fC2 + fC3 - fC1- rReac*M*(-1, -1, 1, 0)' = 0
US1*sum{£mS1} + sum{mS1}*£US1 + eC2 + eC3 - eC1 = 0
#  Equations from reactions and connections
if (mLiquid{1}> 0 AND mVapour{2}> 0) then
    VLiquid** 2 *VVapour** 1/2 *rReac = mLiquid{1}** 2 *mVapour{2}** 1/2
        *K1Reac
else
    rReac = 0
endif
if PS1 > PR2 then
    if LC2_Liquid then
        fC2 = dLiquid*CC2*sqrt{PS1 - PR2}*xLiquid
        eC2 = fC2*HLiquid
    else
        fC2 = dVapour*CC2*sqrt{PS1 - PR2}*xVapour
        eC2 = fC2*HVapour
    endif
else
    fC2 = 0
    eC2 = 0
endif
if PS1 > PR3 then
    if LC3_Liquid then
        fC3 = dLiquid*CC3*sqrt{PS1 - PR3}*xLiquid
        eC3 = fC3*HLiquid
    else
        fC3 = dVapour*CC3*sqrt{PS1 - PR3}*xVapour
        eC3 = fC3*HVapour
```

```
    endif
else
    fC3 = 0
    eC3 = 0
endif
if PR1 > PS1 then
    fC1 = dR1*CC1*sqrt{PR1 - PS1}*xR1
    eC1 = sum{fC1}*HR1
else
    fC1 = 0
    eC1 = 0
endif
#  Additional equations
VLiquid + VVapour = VS1
VVapour*dVapour = sum{mVapour}
VLiquid*dLiquid = sum{mLiquid}
mVapour= xVapour*sum{mVapour}
mS1 = xS1*sum{mS1}
mLiquid= xLiquid*sum{mLiquid}
#  Procedures for flashes, controllers and properties
flash{input: mS1, US1, PS1, output: TS1, sVapour, mVapour, UVapour,
        PVapour, TVapour, dVapour, sLiquid, mLiquid, ULiquid, PLiquid,
        TLiquid, dLiquid}
propH{input: xVapour, UVapour, PVapour; output: HVapour}
propH{input: xLiquid, ULiquid, PLiquid; output: HLiquid}
propd{input: xR1, UR1, PR1; output: dR1}
propH{input: xR1, UR1, PR1; output: HR1}
```

Nomenclature:
CC1 is parameter of flowrate in C1
CC2 is parameter of flowrate in C2
CC3 is parameter of flowrate in C3
dLiquid is density in Liquid
dR1 is density in R1
dVapour is density in Vapour
eC1 is transfer energy in C1
eC2 is transfer energy in C2
eC3 is transfer energy in C3
fC1 is nc-vector of mass transfer rate in C1
fC2 is nc-vector of mass transfer rate in C2
fC3 is nc-vector of mass transfer rate in C3
HLiquid is specific mass enthalpy in Liquid
HR1 is specific mass enthalpy in R1
HVapour is specific mass enthalpy in Vapour
K1Reac is constant of reaction in Reac
LC2_Liquid is a logical variable related to C2_Liquid
LC3_Liquid is a logical variable related to C3_Liquid

M is nc-diagonal matrix of mol weights
mLiquid is nc-vector of mass in Liquid
mS1 is nc-vector of mass in S1
mVapour is nc-vector of mass in Vapour
PLiquid is pressure in Liquid
PR1 is pressure in R1
PR2 is pressure in R2
PR3 is pressure in R3
PS1 is pressure in S1
PVapour is pressure in Vapour
rReac is rate of reaction in Reac
sLiquid is status (V, L, or S) of phase Liquid
sVapour is status (V, L, or S) of phase Vapour
TLiquid is temperature in Liquid
TS1 is temperature in S1
TVapour is temperature in Vapour
ULiquid is internal energy in Liquid
UR1 is internal energy in R1
US1 is internal energy in S1
UVapour is internal energy in Vapour
VLiquid is volume in Liquid
VS1 is volume in S1
VVapour is volume in Vapour
xLiquid is nc-vector of mass fraction in Liquid
xR1 is nc-vector of mass fraction in R1
xS1 is nc-vector of mass fraction in S1
xVapour is nc-vector of mass fraction in Vapour

The System has:
chemical species: A B C D
59 equations
78 variables
19 degrees of freedom and
requires 5 initial conditions

The model can be solved if the following set of variables is fixed:

CC1	CC2
CC3	xR1 *
PR1	dR1
K1React	PR2
PR3	LC2_Liquid
LC3_Liquid	VS1
M **	

* The variable is a nc-vector
** The variable is a diagonal matrix

A.II.- Modelling Flowsheets

The following examples show that the modelling approach proposed in the thesis can be extended to the case of flowsheets. Two cases were chosen, the first illustrates the case of a process system without reaction and the second includes the definition of reactions.

Example A.8 *The CHCLF2 Liquefaction Process:*

Suppose that the model of the process to liquefy monochlorodifluoromethane is requested for simulation. The process flowsheet is given in Figure A.8 as proposed in [Modell and Reid, 1983]. First, the hot, high pressure process gas is cooled by heat transfer in exchanger A and in doing so it is expected to precool to as close to ambient temperature as possible. The gas is further cooled in exchanger B, then passes through the expansion valve and into a liquid-vapour separator. Liquid is removed and saturated vapour is used as the cooling medium in exchanger B. The valve is operated by a controller to maintain a specified upstream pressure. Figure A.9 shows the graphical representation of the process. Note that Figure A.8 includes the terminology of the entities in order to explain how A.8 maps into A.9.

We assume that none of the phases disappears and the flow is always continuous and turbulent. The declarative representation is as follows:

P1= {P,T,V}
P2= {P,T,V]
P3= {P,T,V}
L1= {P,T,V}
V1= {P,T,V}
S1= {L1,V1; P, sumV}
R1= {}
R2= {}
R3= {}
R4= {}
C1= {R1, P1;; TurbulentFlow}
C2= {P1, P2;; TurbulentFlow}

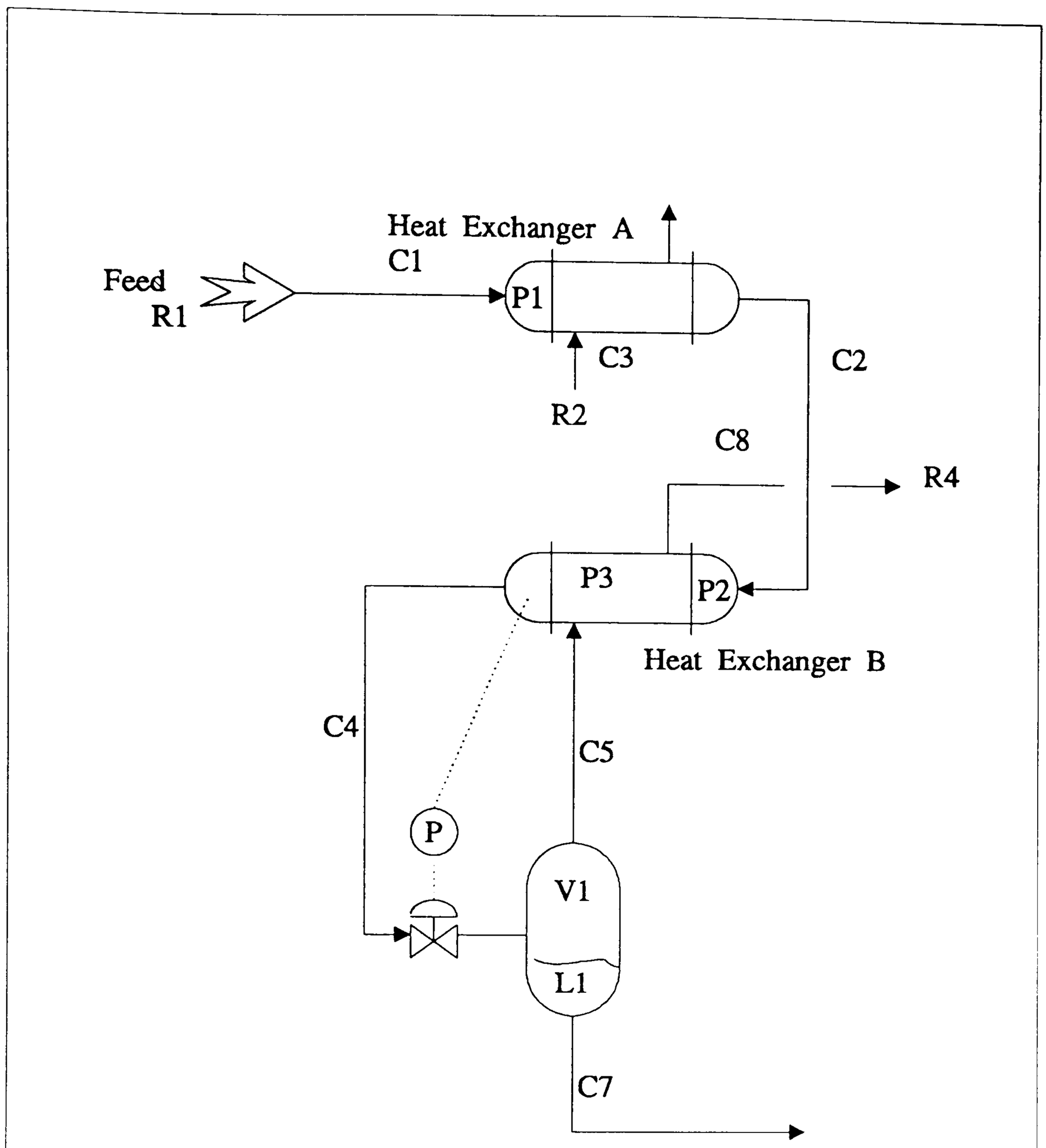


Figure A.8 *Flowsheet for Monochlorodifluoromethane*

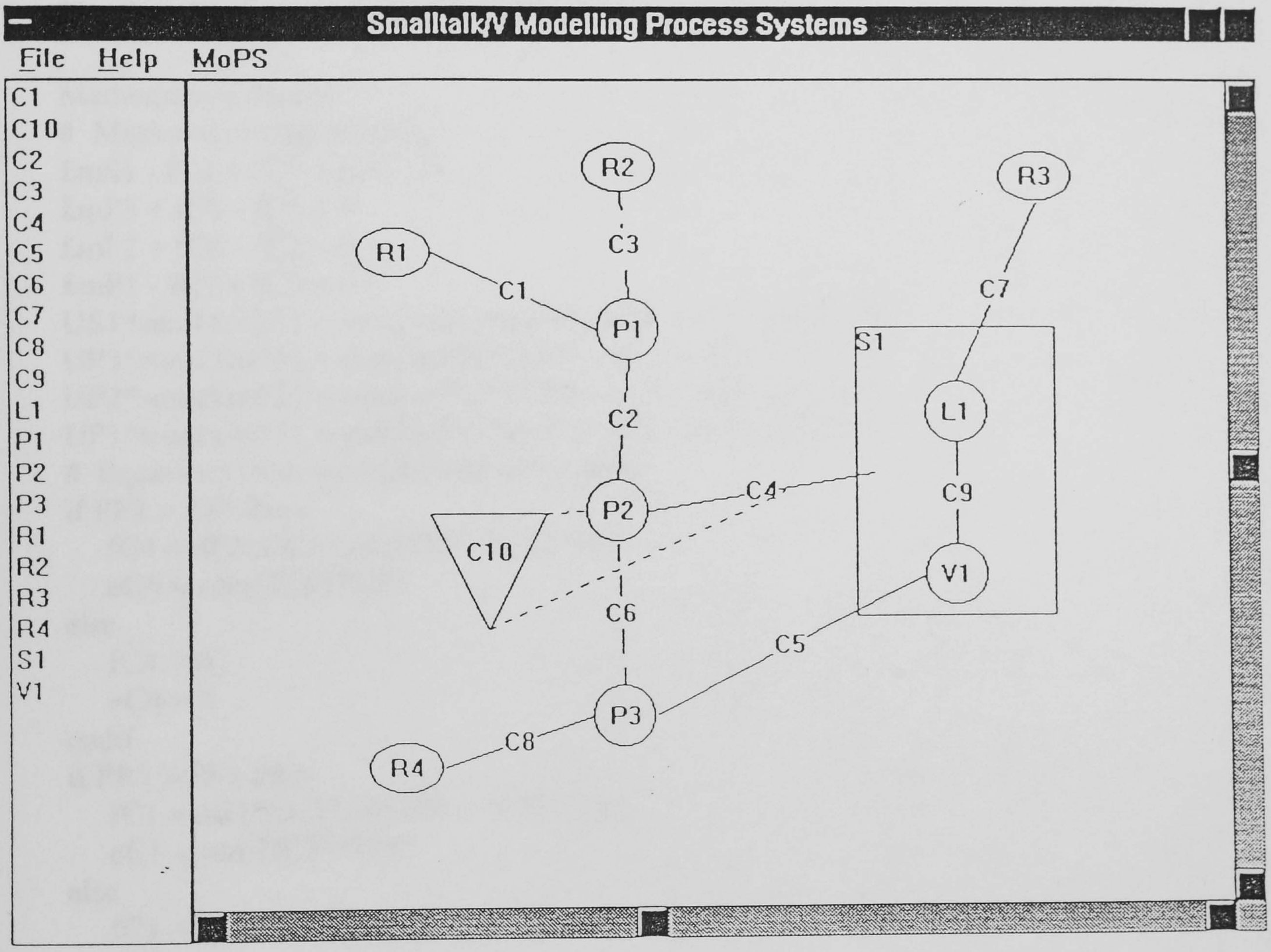


Figure A.9 *Modelling the CHClF2 Process*

C3= {R2, P1;; HeatConvection}
C4= {P2, S1;-phaseSelective; TurbulentFlow}
C5= {V1,P3;; TurbulentFlow}
C6= {P3, P2;; HeatConvection}
C7= {L1, R3;; TurbulentFlow}
C8= {P3, R4;; TurbulentFlow}
C9= {L1,V1;; Equilibration}
C10={input: P in P2; output: C4}

The model generated by the system is,

Mathematical Model:
Mass and energy balances
£mS1 - fC4 + fC7 + fC5 = 0
£mP3 + fC8 - fC5 = 0
£mP2 + fC4 - fC2 = 0
£mP1 - fC1 + fC2 = 0
US1*sum{£mS1} + sum{mS1}*£US1 - eC4 + eC7 + eC5 = 0
UP3*sum{£mP3} + sum{mP3}*£UP3 + eC6 + eC8 - eC5 = 0
UP2*sum{£mP2} + sum{mP2}*£UP2 + eC4 - eC6 - eC2 = 0
UP1*sum{£mP1} + sum{mP1}*£UP1 - eC1 - eC3 + eC2 = 0
# Equations from reactions and connections
if PP2 > PS1 then
    fC4 = dP2*CC4*sqrt{PP2 - PS1}*xP2
    eC4 = sum{fC4}*HP2
else
    fC4 = 0
    eC4 = 0
endif
if PR1 > PP1 then
    fC1 = dR1*CC1*sqrt{PR1 - PP1}*xR1
    eC1 = sum{fC1}*HR1
else
    fC1 = 0
    eC1 = 0
endif
eC6 = hC6*(TP3 - TP2)
eC3 = hC3*(TR2 - TP1)
if PP3 > PR4 then
    fC8 = dP3*CC8*sqrt{PP3 - PR4}*xP3
    eC8 = sum{fC8}*HP3
else
    fC8 = 0
    eC8 = 0
endif

```
if PP4 > PP3 then
    fC5 = dP4*CC5*sqrt{PP4 - PP3}*xP4
    eC5 = sum{fC5}*HP4
else
    fC5 = 0
    eC5 = 0
endif
if PP1 > PP2 then
    fC2 = dP1*CC2*sqrt{PP1 - PP2}*xP1
    eC2 = sum{fC2}*HP1
else
    fC2 = 0
    eC2 = 0
endif
if PL1 > PR3 then
    fC7 = dL1*CC7*sqrt{PL1 - PR3}*xL1
    eC7 = sum{fC7}*HL1
else
    fC7 = 0
    eC7 = 0
endif
#  Additional equations
VL1 + VP4 = VS1
VP2*dP2 = sum{mP2}
VP4*dP4 = sum{mP4}
VP1*dP1 = sum{mP1}
VP3*dP3 = sum{mP3}
VL1*dL1 = sum{mL1}
mP2= xP2*sum{mP2}
mS1 = xS1*sum{mS1}
mP4= xP4*sum{mP4}
mP1= xP1*sum{mP1}
mP3= xP3*sum{mP3}
mL1= xL1*sum{mL1}
#  Procedures for flashes, controllers and properties
flash{input: mS1, US1, PS1, output: TS1, sP4, mP4, UP4, PP4, TP4, dP4, sL1,
        mL1, UL1, PL1, TL1, dL1}
control{input: PP2; output: CC4}
propT{input: xP2, UP2, PP2; output: TP2}
propd{input: xP2, UP2, PP2; output: dP2}
propH{input: xP2, UP2, PP2; output: HP2}
propH{input: xP4, UP4, PP4; output: HP4}
propT{input: xP1, UP1, PP1; output: TP1}
propd{input: xP1, UP1, PP1; output: dP1}
propH{input: xP1, UP1, PP1; output: HP1}
propT{input: xP3, UP3, PP3; output: TP3}
propd{input: xP3, UP3, PP3; output: dP3}
propH{input: xP3, UP3, PP3; output: HP3}
```

propH{input: xL1, UL1, PL1; output: HL1}
propd{input: xR1, UR1, PR1; output: dR1}
propH{input: xR1, UR1, PR1; output: HR1}

Nomenclature:
CC1 is parameter of flowrate in C1
CC2 is parameter of flowrate in C2
CC4 is parameter of flowrate in C4
CC5 is parameter of flowrate in C5
CC7 is parameter of flowrate in C7
CC8 is parameter of flowrate in C8
dL1 is density in L1
dP1 is density in P1
dP2 is density in P2
dP3 is density in P3
dP4 is density in P4
dR1 is density in R1
eC1 is transfer energy in C1
eC2 is transfer energy in C2
eC3 is transfer energy in C3
eC4 is transfer energy in C4
eC5 is transfer energy in C5
eC6 is transfer energy in C6
eC7 is transfer energy in C7
eC8 is transfer energy in C8
fC1 is nc-vector of mass transfer rate in C1
fC2 is nc-vector of mass transfer rate in C2
fC4 is nc-vector of mass transfer rate in C4
fC5 is nc-vector of mass transfer rate in C5
fC7 is nc-vector of mass transfer rate in C7
fC8 is nc-vector of mass transfer rate in C8
hC3 is heat transfer coefficient x surface area in C3
hC6 is heat transfer coefficient x surface area in C6
HL1 is specific mass enthalpy in L1
HP1 is specific mass enthalpy in P1
HP2 is specific mass enthalpy in P2
HP3 is specific mass enthalpy in P3
HP4 is specific mass enthalpy in P4
HR1 is specific mass enthalpy in R1
mL1 is nc-vector of mass in L1
mP1 is nc-vector of mass in P1
mP2 is nc-vector of mass in P2
mP3 is nc-vector of mass in P3
mP4 is nc-vector of mass in P4
mS1 is nc-vector of mass in S1
PL1 is pressure in L1
PP1 is pressure in P1

PP2 is pressure in P2
PP3 is pressure in P3
PP4 is pressure in P4
PR1 is pressure in R1
PR3 is pressure in R3
PR4 is pressure in R4
PS1 is pressure in S1
sL1 is status (V, L, or S) of phase L1
sP4 is status (V, L, or S) of phase P4
TL1 is temperature in L1
TP1 is temperature in P1
TP2 is temperature in P2
TP3 is temperature in P3
TP4 is temperature in P4
TR2 is temperature in R2
TS1 is temperature in S1
UL1 is internal energy in L1
UP1 is internal energy in P1
UP2 is internal energy in P2
UP3 is internal energy in P3
UP4 is internal energy in P4
UR1 is internal energy in R1
US1 is internal energy in S1
VL1 is volume in L1
VP1 is volume in P1
VP2 is volume in P2
VP3 is volume in P3
VP4 is volume in P4
VS1 is volume in S1
xL1 is nc-vector of mass fraction in L1
xP1 is nc-vector of mass fraction in P1
xP2 is nc-vector of mass fraction in P2
xP3 is nc-vector of mass fraction in P3
xP4 is nc-vector of mass fraction in P4
xR1 is nc-vector of mass fraction in R1
xS1 is nc-vector of mass fraction in S1

The System has:
chemical species: CH2ClF2
61 equations
78 variables
17 degrees of freedom and
requires 8 initial conditions

The proposed set of free variables is:

CC1		CC2
CC5		CC7
CC8		dR1
hC3		hC6
xR1	*	PR1
VS1		PR3
PR4		TR2
VP1		VP2
VP3		

*The variable is a nc-vector

Example A.9 *The Methanol Synthesis Process:*

The methanol synthesis process is one of the problems suggested by the European Federation of Chemical Engineers [De Leeuw den Bouter et al, 1983]. A fluid mixture is fed into a mixer and from there into the reactor, see Figure A.10. Two reversible reactions are carried out assuming that both achieve the equilibrium. The reactions are:

$$CO + 2H_2 = CH_3OH,$$

and

$$CO + H_2O = CO_2 + H_2$$

For the purpose of modelling we assume six chemical species: {CO2 H2 CH3OH CO H2O A B} where A and B are inert. We also assume that the pressure of the reactor may vary and the reverse flow occurs between the reactor and the separator. The declarative representation is,

Mixer= {P, T, V}
Reactor= {P, T, V}
Vflash= {P, T, V}
Lflash= {P, T, V}
Feed= { }
Prodv= { }
Prodl= { }
S1= {Vflash, Lflash; P, sumV}

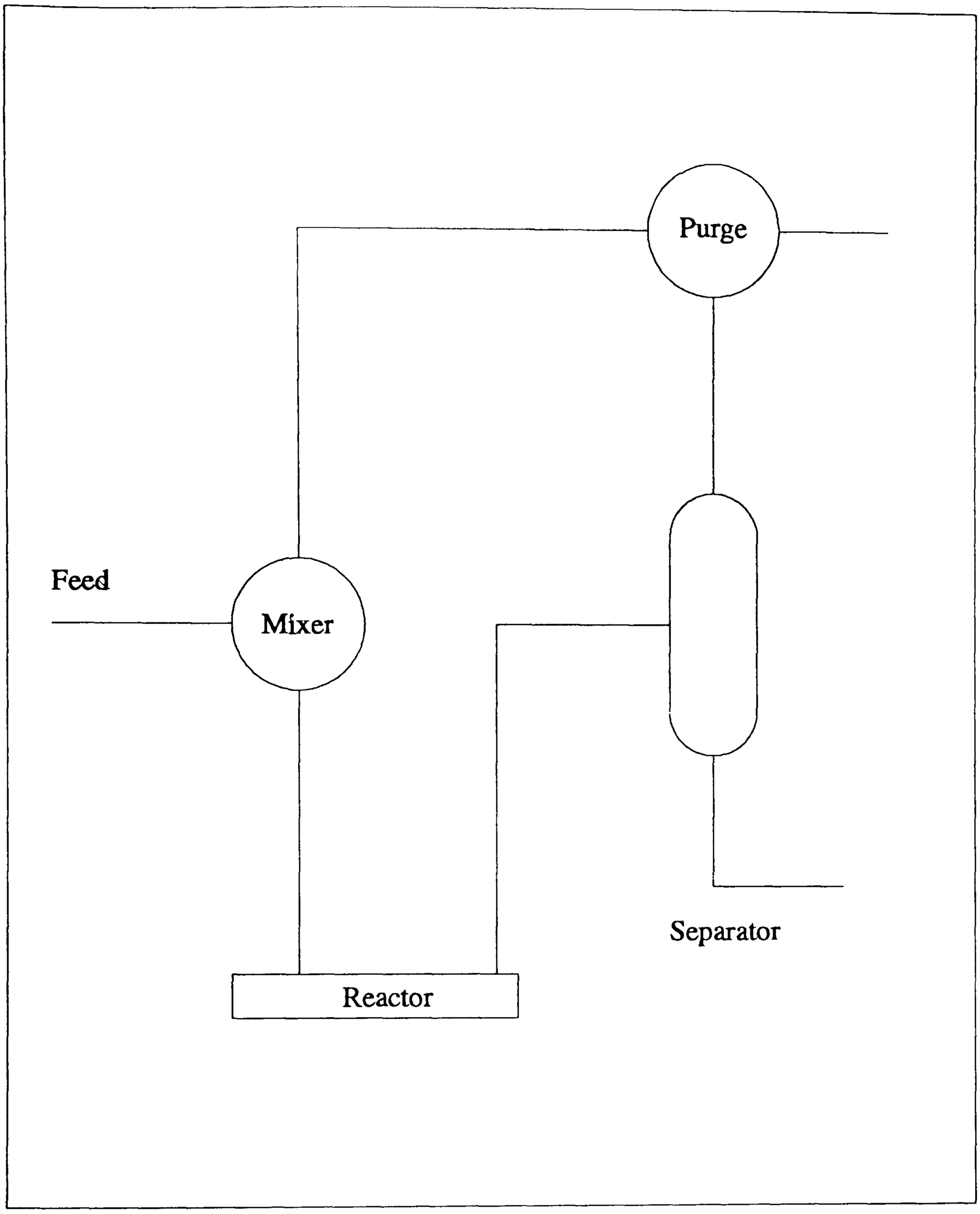


Figure A.10 *Flowsheet for Methanol Synthesis Process*

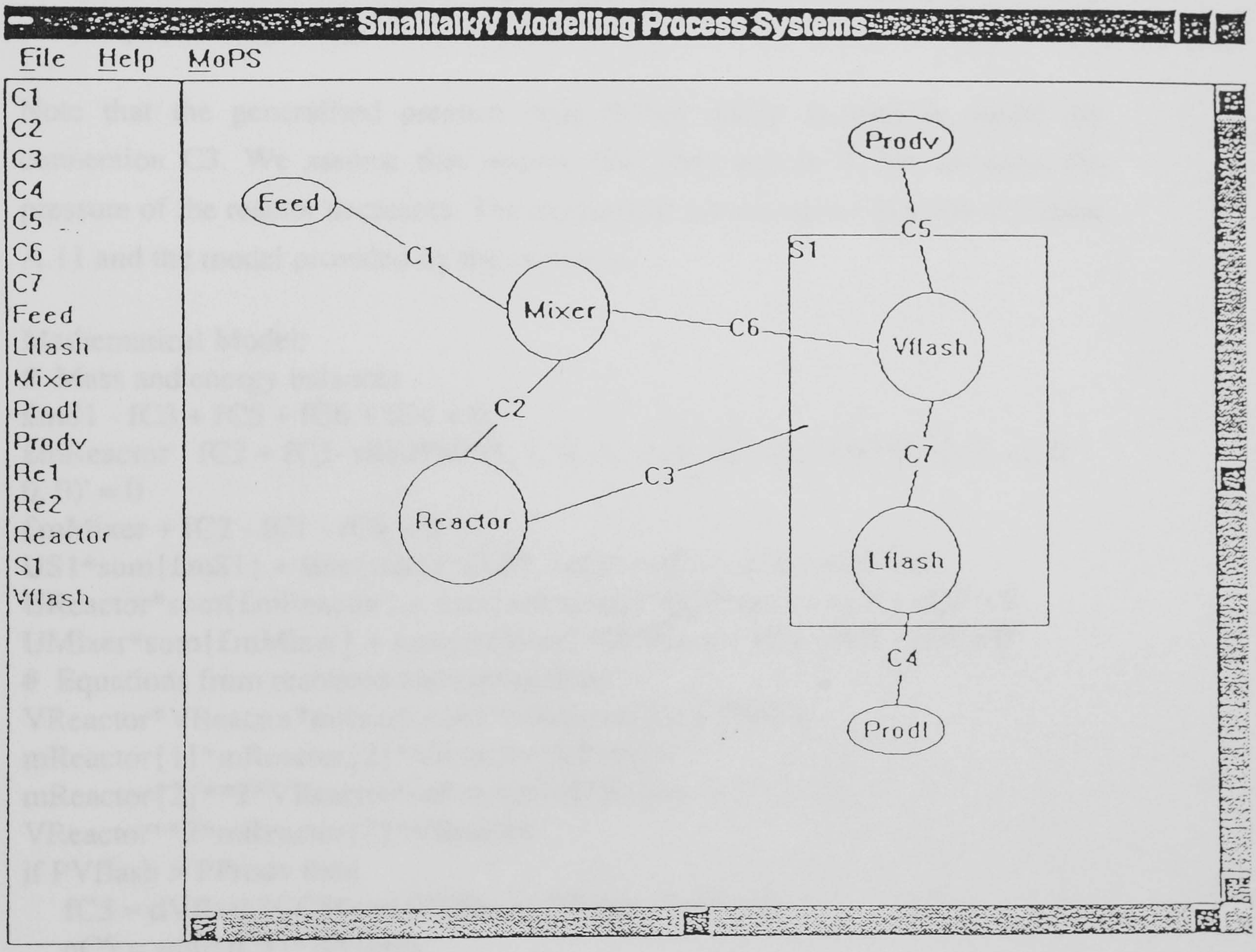


Figure A.11 *Representation of the Methanol Synthesis Process*

C1= {Feed,Mixer;; TurbulentFlow}
C2= {Miser, Reactor;; TurbulentFlow}
C3= {Reactor, S1;-phaseSelective; PressureDropDriven}
C4= {Lflash, Prodl;; TurbulentFlow}
C5= {Vflash, Prodv;; TurbulentFlow}
C6= {Vflash, Mixer;; BulkFlow}
C7= {Vflash,Lflash;; Equilibration}
Re1= {[CO](Reactor)+2[H2](Reactor)=[CH3OH](Reactor); Equilibrium}
Re2= {[CO](Reactor)+[H2O](Reactor)=[CO2](Reactor)+[H2](Reactor);
Equilibrium}

Note that the generalised pressure drop driven model is used to model the connection C3. We assume that reverse flow may appear if, for instance, the pressure of the reactor decreases. The topological representation is given in Figure A.11 and the model provided by the system is,

Mathematical Model:

```
# Mass and energy balances
£mS1 - fC3 + fC5 + fC6 + fC4 = 0
£mReactor - fC2 + fC3- rRe2*M*(1, 1, 0, -1, -1, 0, 0)'- rRe1*M*(0, -2, 1, -1, 0,
0, 0)' = 0
£mMixer + fC2 - fC1 - fC6 = 0
US1*sum{£mS1} + sum{mS1}*£US1 - eC3 + eC5 + eC6 + eC4 = 0
UReactor*sum{£mReactor} + sum{mReactor}*£UReactor - eC2 + eC3 = 0
UMixer*sum{£mMixer} + sum{mMixer}*£UMixer + eC2 - eC1 - eC6 = 0
# Equations from reactions and connections
VReactor*VReactor*mReactor{4}*mReactor{5}*K1Re2 =
mReactor{1}*mReactor{2}*VReactor*VReactor
mReactor{2}**2*VReactor*mReactor{4}*K1Re1 =
VReactor**2*mReactor{3}*VReactor
if PVflash > PProdv then
    fC5 = dVflash*CC5*sqrt{PVflash - PProdv}*xVflash
    eC5 = sum{fC5}*HVflash
else
    fC5 = 0
    eC5 = 0
endif
if PMixer > PReactor then
    fC2 = dMixer*CC2*sqrt{PMixer - PReactor}*xMixer
    eC2 = sum{fC2}*HMixer
else
    fC2 = 0
    eC2 = 0
endif
```

```
if PLflash > PProdl then
    fC4 = dLflash*CC4*sqrt{PLflash - PProdl}*xLflash
    eC4 = sum{fC4}*HLflash
else
    fC4 = 0
    eC4 = 0
endif
if PFeed > PMixer then
    fC1 = dFeed*CC1*sqrt{PFeed - PMixer}*xFeed
    eC1 = sum{fC1}*HFeed
else
    fC1 = 0
    eC1 = 0
endif
if wC6 > 0 then
    fC6 = wC6*xVflash
    eC6 = sum{fC6}*HVflash
else
    fC6 =  wC6*xMixer
    eC6 = sum{fC6}*HMixer
endif
if PReactor - PS1> EC3 then
    fC3 = dReactor*CC3*sqrt{EC3*(PReactor - PS1)}*xReactor
    eC3 = sum{fC3}*HReactor
else if PReactor > PS1 then
    fC3 = dReactor*CC3*(PReactor - PS1)*xReactor
    eC3 = sum{fC3}*HReactor
else if PS1 - PReactor> EC3 then
    if LC3_Vflash then
        fC3 =  - dVflash*CC3*sqrt{EC3*(PS1 - PReactor)}*xVflash
        eC3 = sum{fC3}*HVflash
   else
        fC3 =  - dLflash*CC3*sqrt{EC3*(PReactor - PS1)}*xLflash
        eC3 = sum{fC3}*HLflash endif
else  if LC3_Vflash then
        fC3 =  - dVflash*CC3*(PS1 - PReactor)*xVflash
        eC3 = sum{fC3}*HVflash
   else
        fC3 =  - dLflash*CC3*(PReactor - PS1)*xLflash
        eC3 = sum{fC3}*HLflash
   endif
endif
#  Additional equations
VVflash + VLflash = VS1
VLflash*dLflash = sum{mLflash}
VMixer*dMixer = sum{mMixer}
VReactor*dReactor = sum{mReactor}
VVflash*dVflash = sum{mVflash}
```

```
mLflash= xLflash*sum{mLflash}
mMixer= xMixer*sum{mMixer}
mReactor= xReactor*sum{mReactor}
mVflash= xVflash*sum{mVflash}
mS1 = xS1*sum{mS1}
#  Procedures for flashes, controllers and properties
flash{input: mS1, US1, PS1, output: TS1, sVflash, mVflash, UVflash, PVflash,
        TVflash, dVflash, sLflash, mLflash, ULflash, PLflash, TLflash, dLflash}
propH{input: xLflash, ULflash, PLflash; output: HLflash}
propT{input: xMixer, UMixer, PMixer; output: TMixer}
propd{input: xMixer, UMixer, PMixer; output: dMixer}
propH{input: xMixer, UMixer, PMixer; output: HMixer}
propT{input: xReactor, UReactor, PReactor; output: TReactor}
propd{input: xReactor, UReactor, PReactor; output: dReactor}
propH{input: xReactor, UReactor, PReactor; output: HReactor}
propH{input: xVflash, UVflash, PVflash; output: HVflash}
propd{input: xFeed, UFeed, PFeed; output: dFeed}
propH{input: xFeed, UFeed, PFeed; output: HFeed}
```

Nomenclature:
CC1 is parameter of flowrate in C1
CC2 is parameter of flowrate in C2
CC3 is parameter of flowrate in C3
CC4 is parameter of flowrate in C4
CC5 is parameter of flowrate in C5
dFeed is density in Feed
dLflash is density in Lflash
dMixer is density in Mixer
dReactor is density in Reactor
dVflash is density in Vflash
eC1 is transfer energy in C1
eC2 is transfer energy in C2
EC3 is flow parameter in C3
eC3 is transfer energy in C3
eC4 is transfer energy in C4
eC5 is transfer energy in C5
eC6 is transfer energy in C6
fC1 is nc-vector of mass transfer rate in C1
fC2 is nc-vector of mass transfer rate in C2
fC3 is nc-vector of mass transfer rate in C3
fC4 is nc-vector of mass transfer rate in C4
fC5 is nc-vector of mass transfer rate in C5
fC6 is nc-vector of mass transfer rate in C6
HFeed is specific mass enthalpy in Feed
HLflash is specific mass enthalpy in Lflash
HMixer is specific mass enthalpy in Mixer
HReactor is specific mass enthalpy in Reactor

HVflash is specific mass enthalpy in Vflash
K1Re1 is constant of reaction in Re1
K1Re2 is constant of reaction in Re2
LC3_Vflash is a logical variable related to C3_Vflash
M is nc-diagonal matrix of mol weights
mLflash is nc-vector of mass in Lflash
mMixer is nc-vector of mass in Mixer
mReactor is nc-vector of mass in Reactor
mS1 is nc-vector of mass in S1
mVflash is nc-vector of mass in Vflash
PFeed is pressure in Feed
PLflash is pressure in Lflash
PMixer is pressure in Mixer
PProdl is pressure in Prodl
PProdv is pressure in Prodv
PReactor is pressure in Reactor
PS1 is pressure in S1
PVflash is pressure in Vflash
rRe1 is rate of reaction in Re1
rRe2 is rate of reaction in Re2
sLflash is status (V, L, or S) of phase Lflash
sVflash is status (V, L, or S) of phase Vflash
TLflash is temperature in Lflash
TMixer is temperature in Mixer
TReactor is temperature in Reactor
TS1 is temperature in S1
TVflash is temperature in Vflash
UFeed is internal energy in Feed
ULflash is internal energy in Lflash
UMixer is internal energy in Mixer
UReactor is internal energy in Reactor
US1 is internal energy in S1
UVflash is internal energy in Vflash
VLflash is volume in Lflash
VMixer is volume in Mixer
VReactor is volume in Reactor
VS1 is volume in S1
VVflash is volume in Vflash
wC6 is total rate of flow in C6
xFeed is nc-vector of mass fraction in Feed
xLflash is nc-vector of mass fraction in Lflash
xMixer is nc-vector of mass fraction in Mixer
xReactor is nc-vector of mass fraction in Reactor
xS1 is nc-vector of mass fraction in S1
xVflash is nc-vector of mass fraction in Vflash

The System has:
chemical species: CO2 H2 CH3OH CO H2O A B

149 equations
180 variables
31 degrees of freedom and
requires 24 initial conditions

A suggested set of free variables is:

CC1	CC2	VReactor
CC3	CC4	LC3_Vflash
CC5	dfeed	VMixer
PProdv	PFeed	M **
PProdl	wC6	VS1
xFeed *	EC3	
K1Re1	K1Re2	

*The variable is a nc-vector
** The variable is a diagonal matrix