Integrated Process and Control System Design

D. Murray. Laing

Thesis presented for Degree of Doctor of Philosophy
University of Edinburgh
1995

Declaration

I declare that this thesis was composed by myself and that it describes my own work except where specifically stated in the text. The work was carried out between October 1987 and October 1993 in the Department of Chemical Engineering at the University of Edinburgh under the supervision of Prof. J.W.Ponton.

D.Murray.Laing

Acknowledgements

I would like to thank all the people of the *ECOSSE* research group for their help and support. Special thanks are due to Douglas Hutton who was a valuable sounding board for ideas in the early years of this work, Neil Skilling for the computing support he has provided and Eric Fraga and Rama Lakshmanan for proof reading this thesis. I must also thank Prof. Ponton for his patience and tolerance in supporting this work. Finally I wish to thank my wife, Teresa, whose support while I was finishing this thesis was invaluable.

Abstract

The subject of this thesis is the integration of process and control system design. A review is provided of the methods that have been developed to assess the operability of a process design which have been the principal focus for process and control system integration. Such methods are only part of the solution to design integration. Concurrent design of the process and its control system is proposed as the mechanism for more complete integration.

To support concurrent design a framework for hierarchical design of a *process operating system* is developed. A process operating system is defined as the complete collection of control schemes, alarms and operating procedures used for managing the process through all phases of operation. The design of an integrated operating system is approached by decomposing the problem into a hierarchy of operating tasks. Three classes of operating task are identified: regulatory tasks for optimising operation at a steady state, transition tasks for transferring the process from one regulatory state to another and executive tasks which manage the response to discrete events such as alarms and failures.

Operating tasks define the requirements for optimisation and failure management. The implementation of an operating task is achieved by the design of a *control scheme* for which a generic structure has been developed. The structure emphasises the use of explicit models with *parameter estimation* and *control distribution* blocks providing the interface between the abstract model used for optimisation and the reality of the underlying system.

A knowledge based representation has been developed to support operating system design. Particular attention has been given to the problem of supporting concurrent design of the process and operating system. A representation has been developed that links process design alternatives with operating system design alternatives by their association with a common operating task.

A case study that considers the design of a hierarchical operating system for a hydrofluoric acid plant is included in this thesis. The study demonstrates how the operating system may be developed in step with the process design. The hierarchical development of the process is used to help formulate the operating tasks for the operating system design. Through design of the operating system it is possible to provide focussed feedback on the process operability requirements. The final operating system structure demonstrates how failure management and optimisation are integrated together.

Concurrent design makes it easier to formulate focussed operability studies during the preliminary design of the process which are valuable in avoiding potential operability problems. By timely identification of operability requirements more appropriate process designs can be developed. Integrated design of a process and its operating system is thus a significant aid to designing operable plants.

Table of Contents

List of Figures

Chapter 1

Introduction

The subject of this thesis is the integration of process and control system design. A primary motivation for integration is the production of process designs that take appropriate account of how a process will be operated. The next chapter will review the current research on methods for evaluating the operability of a process design. Following the review a framework for the hierarchical design of a 'process operating system' is presented. A case study on the design of a hydrofluoric acid plant illustrates how the framework is used to develop a strategy for process operation in step with the hierarchical design of the process.

1.1 The divide between process and control system design

The conventional perspective on the development of a process is that the design of the control system does not occur until the process itself is well developed. By this stage most of the significant process design decisions have already been committed. If there is a control problem the main recourse is to use more sophisticated control schemes. Such schemes however are more costly to design, implement, and maintain. There are times when modification of the process can provide a better solution. An example is provided by Ryskamp [1] who compared implicit and explicit decoupling of distillation control systems. One of the main observations of the work was that the implicit solution (*ie.* designing to avoid the interaction) in general provided a more robust means of achieving decoupling.

There are two areas of activity directed at bridging the divide between process design and control system design. The first is the development of methods to evaluate the inherent operability of a process design. The second is the development of new approaches to design that integrate the development of the process and control system.

1.2 Defining Process Operability

Operability is an umbrella term used to encompass all the properties that determine whether a process can meet the real time demands that will be made of it. A 'real time demand' may arise from a variety of sources. They can be due to intentional and predictable operations such as switching product grades. Alternatively they can be due to unavoidable uncertainty in the final operating conditions *eg.* variability in catalyst activity. Douglas and Fisher [2] refer to these as the environmental connections of the process and divide them into six categories:

1. Product Quality
2. Production Rate
3. Flows, composition, pressure & temperature of raw materials
4. Flows, composition, pressure & temperature of utilities
    (a) Fuel
    (b) Steam
    (c) Cooling Water
5. Process constraints
6. Internal Connections:
    (a) Catalyst Deactivation

(b) Exchanger Fouling

(c) Equipment Wear

Each environmental connection represents a factor over which the designer has little control. To obtain a base case design it is necessary to make assumptions about each of these connections. The real time demands arise when operation deviates from these assumptions.

The most common approach used to avoid operability problems is to add safety margins or overdesign factors to the design. While the use of design margins is a relatively simple approach to apply it has distinct limitations:

- Overdesign factors are usually quite arbitrary and it is not easy to select the proper amount of over design to match the desired degree of operability, nor where best to apply the overdesign.

- Overdesign tends to consider units in isolation whereas operability is a property of the whole plant and so is better addressed in the scope of the whole plant. This is particularly the case in highly integrated plants where interactions between units are more numerous.

Validating that a process has sufficient operability has been dependent on simulation, either steady state or dynamic. Simulation, however, can also be an arbitrary process. One of the main problems faced is the choice of suitable disturbances to test operability, *ie.* what is the worst possible condition with which to test the plant. Intuitive judgement is relied upon to identify the worst possible case but, as Grossmann and Morari [3] illustrate, proper analysis can sometimes defy intuition. To provide confidence in the operability of a plant several simulation scenarios would have to be run which implies a large consumption of the designer's and computer's time.

There are significant benefits to be gained by a systematic approach to process operability. The research in this field has divided into two general categories:

- **Flexibility**: a flexible plant is capable of maintaining feasible operation for the defined range of disturbances. The evaluation of flexibility is based on steady state process models.

- **Dynamic Resilience**: also referred to as controllability. Dynamic resilience measures are based on the analysis of the plant dynamics and how these affect the ease of control.

The methods developed for analysis and design in both these areas are reviewed in chapter 2. The tools that have developed in these areas provide measures of the inherent operability limitations of a process design. What is often not considered is how they will be utilised during the process of design.

1.3 Integrated Design of a Process and its Operating System

Flexibility and controllability analysis can help the process engineer develop designs that are easier to operate but the burden is on the process engineer to apply these tools and interpret the results. Integrated process and control system design is a way of giving the control engineers a more active role in the development of the process.

The philosophy underlying integrated design is to involve all disciplines in the development of the complete design (figure 1.1). All facets of chemical plant design (the process, the control system, safety procedures, *etc.*) are developed concurrently. Integrated design can ease the burden on the process designer. For example, the control system designers are more likely to have the skills suited to applying and interpreting many of the controllability analysis tools. In addition concurrent design has the potential to make the development of the control system

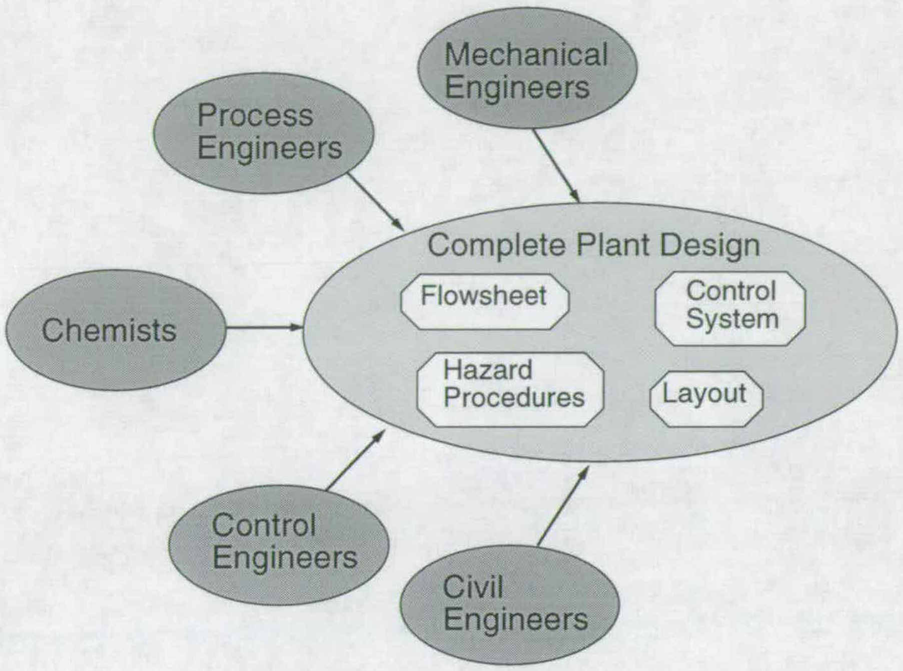


Figure 1.1: Concurrent Integrated Design

easier. If the control engineer's first sight of the process is only when the process flow diagram is fully developed the amount of detail can be overwhelming. By being involved in the evolution of the process it is easier to develop an understanding of the design intent and develop a control strategy to match.

In this thesis concurrent design is considered on a broader scope than just the regulatory control system. The concept of a *process operating system* is introduced. A process operating system is the complete collection of control schemes, alarms and procedures used in managing process operation. The goal is to develop an integrated approach for complete operations management. A framework for concurrent process and operating system design is developed in chapter 3.

As a demonstration of this framework a case study on the design of a hydrofluoric acid plant is presented in chapter 4. The result is an operating policy that addresses both failure management and process optimisation. In addition the case study illustrates how a more complete picture of the operating requirements for a process may be developed from the initial stages of design.

Chapter 2

A Review of Process Operability Analysis

In this chapter we will provide an overview of the methods that are available to assess and improve process operability. The review has been divided into three principal sections:

- **Flexibility**: steady state analysis of operation feasibility for variations from nominal design conditions.

- **Dynamic Resilience**: the impact of process design on the regulatory control of a process.

- **Operating Procedures**: the methods for planning major changes in operation such as startup or shutdown.

2.1 Review of Flexibility Analysis Techniques

Process flexibility is treated as a steady state problem. The goal is to ensure that process operation is feasible for the full range of expected deviations from the norm. Flexibility is limited by the ability of a plant to meet the constraints that arise from safety, environmental, and equipment restrictions. The treatment of these constraints is an important aspect of flexibility analysis. The constraints that a designer would apply at the nominal design conditions do not necessarily correspond to those that would be applied during departures from the nominal conditions. Grossmann and Morari [3] suggest a division of constraints into two categories:

- Hard Constraints: constraints which should never be violated (*eg.* product specifications, safety)

- Soft Constraints: constraints which are more heuristic guidelines rather than rigid rules (*eg.* minimum temperature approach).

At the nominal design point both sets of constraints would be applied, beyond this point only the hard constraints are applied.

Before considering the possible ways of tackling flexibility it is first necessary to express the design problem in a suitable form. A general mathematical representation expresses the design problem as a set of non-linear equalities, and a set of inequalities, *viz*

$$h(d,u,x,p) = 0 \tag{2.1}$$

$$g(d,u,x,p) \leq 0 \tag{2.2}$$

where $h \equiv$ vector of equalities, *eg.* heat & mass balances.

$$\begin{aligned} g &\equiv \text{vector of inequalities, } eg. \text{ product specifications} \\ d &\equiv \text{vector of design variables} \\ u &\equiv \text{vector of control variables} \\ x &\equiv \text{vector of state variables} \\ p &\equiv \text{vector of uncertain parameters} \end{aligned}$$

For the nominal conditions for p the optimal design is given by,

$$\min_{d,u} C(d,u,x,p) \tag{2.3}$$
$$\text{where } C \equiv \text{Cost Function}$$

The distinction between design variables and control variables is important, since only the control variables may be manipulated after a plant is built to cope with variations in the uncertain parameters. The control variables selected do not have to be those that will ultimately be used for control, but must be of sufficient number to eliminate any degrees of freedom left once d and p are specified.

There are three levels at which flexibility is typically addressed:

1. Assessment of the degree of flexibility: Determine how much variation in the uncertain parameters a design can cope with.

2. Optimal design for specified flexibility: Determine the design which can cope with specified parameter variations in the most cost effective way.

3. Design for optimal degree of flexibility: Balance the operating benefits of increased flexibility against increased capital cost.

Each will be considered in turn in sections 2.1.1 to 2.1.5.

2.1.1 Assessing The Degree of Flexibility in a Design

Although this problem would seem to be basic to any form of flexibility analysis, it has only been tackled by a few workers. Most of the studies have been addressed specifically to the problem of designing flexible heat exchanger networks, a topic that will be discussed later. A general method for calculating a 'flexibility index' is developed by Grossmann and Swaney [4]. Their work provides some fundamental insights on the analysis of flexibility.

The first step is to obtain a description of the feasible region of operation for a fixed plant design. The set of equalities, h, may be rearranged to express the state variables, x, as a function of d, u and p, *viz*

$$h(d,u,x,p) = 0 \quad \Rightarrow \quad x = x(d,u,p) \tag{2.4}$$

Substituting for x into the set of inequalities, g, gives a new set of inequalities, f, dependent on d, u & p only, *viz.*

$$\begin{aligned} g(d,u,x,p) \leq 0 \quad &\Rightarrow \quad g(d,u,x(d,u,p),p) \leq 0 & (2.5) \\ &\Rightarrow \quad f(d,u,p) \leq 0 & (2.6) \end{aligned}$$

Thus, when the design variables, d, are known the feasibility of a design for a given value of the parameters, p, is determined by the existence of a set of controls, u, such that $f(d,u,p)$ is less than or equal to zero. The feasible region, R, for the parameters, p, is defined by,

$$\begin{aligned} R &= \{p \mid [\, \exists \; u \mid f(d,u,p) \leq 0]\} & (2.7) \\ \Rightarrow R &= \{p \mid \Psi(d,p) \leq 0\} & (2.8) \end{aligned}$$

$$\begin{aligned} \text{where, } \Psi(d,p) &= \min_{u} \max_{j \in J} f_j(d,u,p) \\ J &= \text{Index set of function vector } f \end{aligned}$$

The function $\Psi(d,p)$ forms a basic measure of feasibility (≤ 0) or infeasibility (≥ 0). The region R provides the basic information on flexibility. Visualising this region once the dimension of p exceeds 3 is difficult. An approach proposed by Arkun and Etzkorn [5], is to generate plots of the feasible region for selected pairs of parameters.

Instead of trying to describe the shape of the feasible region Grossmann and Swaney [4] attempt to identify the maximum amount by which each uncertain parameter may vary independently. The approach is analogous to trying to find the largest hyper-rectangle which may fit within the feasible region. The proportions of the hyper-rectangle are determined by the upper and lower bounds on each parameter and a size index δ such that the set of enclosed parameters P is given by

$$P(\delta) = \left\{ p \mid (p^N - \delta\Delta p^-) \leq p \leq (p^N + \delta\Delta p^+) \right\} \tag{2.9}$$

$$\begin{aligned} \text{where, } \Delta p^- &= p^N - p^L \\ \Delta p^+ &= p^U - p^N \\ p^N &= \text{Nominal Parameter Value} \\ p^U &= \text{Expected Upper Bound} \\ p^L &= \text{Expected Lower Bound} \end{aligned}$$

For the simple two variable case the hyper-rectangle is illustrated in figure 2.1. It can be seen that $P(\delta)$ expands out from the nominal point, as δ increases. An approximation of the feasible region is given by the largest hyper-rectangle that fits within it. The size index δ is treated as a measure of the potential flexibility of the chosen design relative to the nominal disturbance region. The flexibility index F is defined by,

$$F = \max \delta \;\; \text{s.t.} \;\; P(\delta) \subseteq R \tag{2.10}$$

Designs for which $F \geq 1$, have sufficient flexibility to cope with parameter deviations equal to, or in excess of, the specified bounds. If $0 \leq F \leq 1$ the design can only cope with a maximum fraction F of the expected deviations. The analysis requires only minimal information about the parameter uncertainty, namely a nominal point and upper and lower bounds.

The efficient computation of F is not a trivial problem as becomes clear by considering how the requirement that $P(\delta)$ be enclosed by R may be tested math-

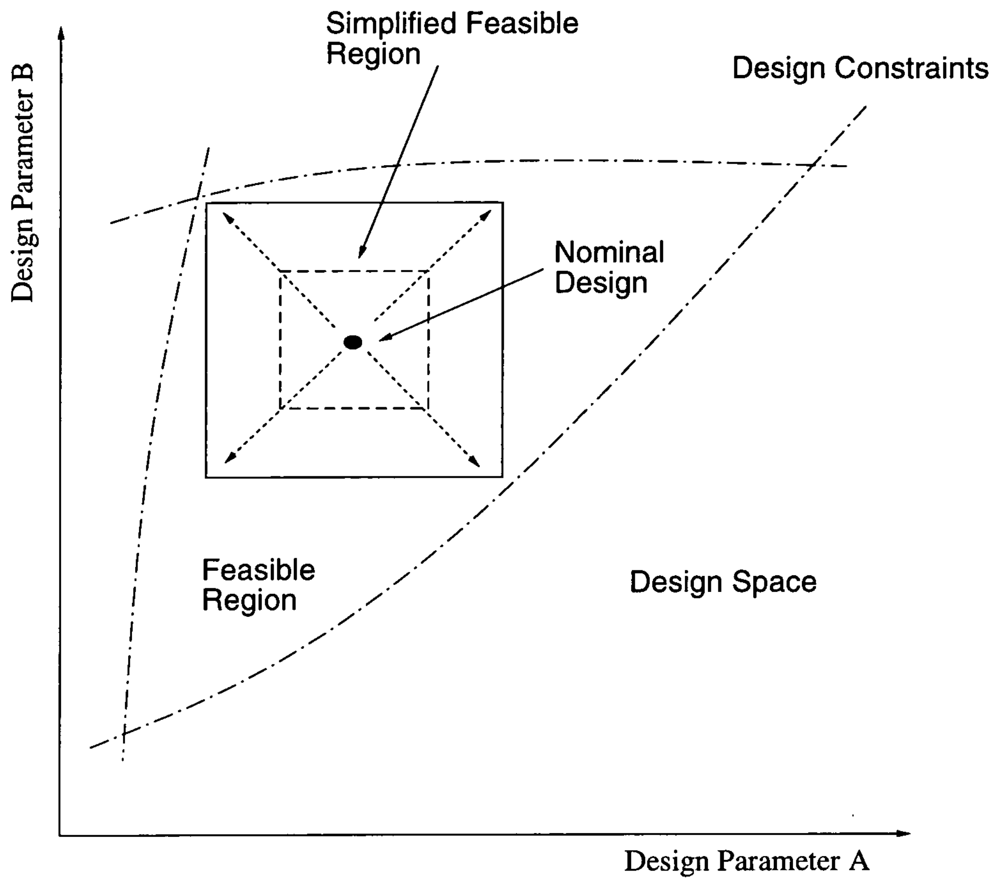


Figure 2.1: Expanding hyper-rectangles for two variable case

ematically,

$$P(\delta) \subseteq R$$

$$\Rightarrow \quad \forall p \in P(\delta),\ \Psi(d,p) \leq 0 \tag{2.11}$$

$$\Rightarrow \quad \max_{p\in P(\delta)} \min_{u} \max_{j\in J} f_j(d,u,p) \leq 0 \tag{2.12}$$

The *max-min-max* formulation generally results in non-differentiable goals leading to a difficult non-linear programming problem. The additional goal of maximising δ further adds to the complexity of the problem.

Some simplification is possible by considering what Grossman refers to as the critical points of the disturbance region. These are the points on the hyper-rectangle which become infeasible first, and may be regarded as the points at which the operating conditions are worst. The formulation above involves an explicit search for these critical points. If they can be predicted beforehand one layer of optimisation can be simplified.

An intuitive assignment of critical points is to select the extreme values of p which correspond to the corners of hyper-rectangle. The assumption is valid provided the shape of the feasible region is convex. It has been demonstrated, in particular for heat exchanger networks, that such conditions do not always hold and intuition has not been able to identify the critical points. Unfortunately unless the problem is of special mathematical structure it is not possible to verify whether critical points will occur at the vertices. However, if this assumption is valid the evaluation of F can be simplified to,

$$F = \max_{u,\delta_k} \delta_k \quad s.t. \, \forall k \in V, \; f(d, u, p(\delta_k)) \leq 0 \tag{2.13}$$

$$\begin{aligned} \text{where, } p(\delta_k) &= p^N + \delta_k \Delta p^k \\ \Delta p^k &= \text{direction of vertex } k \\ V &= \text{Set indexing all vertices} \end{aligned}$$

Even in this form the problem can be computationally intensive. The number of vertices to be considered will increase exponentially with problem size. Also there is no easy way to guarantee that the critical points will correspond to the vertices. These limitations makes the formulation unsatisfactory. Grossmann and Floudas [6] present a reformulation of the flexibility index as a mixed integer programming problem which, it is suggested, is capable of dealing with the problems of non-convex feasible regions, and for convex regions is expected to use fewer search points than a vertex search.

Considered on the basis of probabilities the use of a hyper-rectangle to approximate the feasible region is a conservative approach. The probability of a set of parameters deviating to their maximum or minimum simultaneously is small. It is more likely to get peaks in single parameters. A measure of flexibility that considers only single peaks has been proposed by Morari *et al.* [7]. Their 'resilience index' is a measure of flexibility for heat exchanger networks. A polytope is used rather than a hyper-rectangle to describe the parameter space. The vertices of the polytope expand out parallel to the parameter axes and correspond to extremes of individual parameters rather than extremes of combinations of parameters.

2.1.2 Optimal Design for a Fixed Amount of Flexibility

Having established a means of measuring flexibility, a natural extension to the problem is to consider the most effective way of achieving a desired level of flexibility. The problem here is to optimise the economics of the design while ensuring that the plant will still be capable of operating over the full range of expected parameter values. The optimal design problem has received significantly more attention than the previous question of measuring how much flexibility a process has. Many of the early formulations, which are reviewed by Grossmann *et al.* [8], failed to properly express the problem. For example, no distinction would be made between the control and design variables which would lead to an optimisation of the form,

$$\begin{aligned} & \min_{d,u} E_p \left\{ C(d,u,x,p) \right\} \qquad (2.14) \\ \text{s.t.} \quad & h(d,u,x,p) = 0 \\ & g(d,u,x,p) \leq 0 \\ \text{where,} \ E_p \quad & = \text{Expected value function based on range of } p \end{aligned}$$

An optimisation of this form results in over conservative designs because no allowance is made for the fact that the control variables may be adjusted to reduce the effect of parameter variations. A more appropriate formulation is to include the control variables in an inner optimisation of the cost function, *viz*

$$\begin{aligned} & \min_{d} E_p \left\{ \min_{u} C(d,u,x,p) \right\} \qquad (2.15) \\ \text{s.t.} \quad & h(d,u,x,p) = 0 \\ & g(d,u,x,p) \leq 0 \end{aligned}$$

The two-stage optimisation is sometimes referred to as the 'Here and Now' problem. While this formulation will give the least conservative design, a rigorous solution for any problem of realistic size is generally infeasible. In order to cope with problems of significant size a few workers have searched for suitable decomposition strategies. The approaches divide into two groups,

a) Specify Control Strategy : *ie.* Johns and Lakshmanan [9] specify a control objective of maintaining constant flows between process sections thus reducing the interaction between sections so that each section may be optimised separately.

b) Reduce to a Multi-period Design Problem : *ie.* consider only a discrete set of points within the parameter space and optimise design for this set of operating points. To select a representative set of operating points Malik and Hughes [10] use a stochastic sampling technique. Grossmann and Morari [3] however leave the selection to the designer.

Treatment of the constraints in these formulations varies significantly. In the work of Malik and Hughes [10] feasibility is only checked for the randomly selected set of parameter points. In the work of Grossmann and Morari [3] explicit consideration is given to ensuring feasibility for all possible parameter realisations. All vertices of the parameter space are included in the set of operating points as estimates of the critical points of the feasible region. However, the approach will only be valid if the feasible region has a convex shape, otherwise an alternative technique to find the critical points would be required which would further increase the computational effort required for this problem.

In the work of Johns and Lakshmanan [9] a quite distinct approach to uncertainty is taken. The paper builds upon earlier work of Johns *et al.*[11, 12]. The approach taken is to realistically cost every possible outcome of the uncertain parameters and optimise the process design hierarchically taking into account the full range of possible outcomes and their probabilities. If a particular outcome would result in the process failing to meet its requirements that would be reflected in the expected value. Thus while the approach does not aim to ensure feasibility for all possible parameter realisations the potential impact of infeasibility can be taken into account in the optimisation. The objective function considered in Johns and Lakshmanan [9] incorporates a constraint which specifies a lower limit on the probability of feasible operation for the process design. For example the designer may require that there be a 90% chance of process operation being feasible.

A variation on the problem of optimal design for fixed flexibility is the optimal redesign of a chemical process to increase its flexibility which is addressed by Pistikopoulos and Grossmann [13]. They show that for a linear model there exist analytical properties for flexibility that make it possible to formulate an efficient MILP problem. The MILP reformulation avoids problems of embedded optimisations.

Even assuming that a feasibility test could be done efficiently the above routines tend to isolate the designer from the decisions concerning the design. Instead it may be preferable to develop a more interactive approach, with the computer providing guidance only. If satisfactory techniques can be developed for optimal design with fixed flexibility, the next step would be to consider how much flexibility should actually be incorporated in a process design.

2.1.3 Design with Optimal Degree of Flexibility

The progression from optimal design with fixed flexibility is to determine the optimal degree of flexibility. From the difficulties encountered with the previous problem it is apparent that this is far from a trivial problem. While it is possible to assess the cost of flexibility in terms of extra capital costs, evaluating the benefits of extra flexibility is more difficult. An approach based on stochastic analysis is presented by Pistikopoulos and Grossmann [14]. The approach utilises analytical properties derived for the flexibility of linear process models to simplify the construction of a trade off curve of retrofit costs against flexibility. Stochastic analysis is then used to determine the expected revenue for particular points on this curve.

2.1.4 Stochastic Evaluation of Flexibility

Instead of working simply with bounds on parameters a more meaningful measure of flexibility might be gained by stochastic analysis. Pistikopoulos and Mazzuchi [15] have developed a stochastic flexibility index which measures the probability of feasible operation. Asbjornsen [16] developed a systems based analysis of operability which decomposed the the evaluation of operability into three probability measures:

- **Reliability**: the probability of feasible operation. Reliability may be measured, for example, in terms of the mean time between failures. It will be dependent not only on the feasibility of control but also the reliability of the process equipment.

- **Availability**: the proportion of time that the plant is available for production. Availability is closely related to reliability. It is also affected by the time it takes to bring the process from a state of failure back to normal operation.

- **Performance**: the probability of producing to the standards required.

However Asbjornsen's approach to process operability has received little further development.

2.1.5 Flexibility Analysis For Heat Exchanger Networks

So far the discussion has focussed on techniques for flexibility analysis that are of general application. A result of their generality is that their solution becomes complex. An alternative approach is to develop tools focussed on particular sources of inflexibility. One area which has received such a treatment is the design of flexible heat exchanger networks (HEN).

Some of the early work on flexibility in HEN was performed by Morari and coworkers. In Morari *et al.* [17] a synthesis procedure was proposed based on merging the HEN designs obtained from considering certain special cases. Each case was designed to "subject a different part of the process to a severe test". For example, one test is to require maximum cooling from the network. The test is analogous to a limited inspection of the corner points of the parameter range which implies the same restriction to convex feasible regions. A *corner point theorem* was developed by Morari and Saboo [18] which laid out sufficient conditions to guarantee that the feasible region would be convex (*eg.* constant heat capacity flowrate). If a HEN problem satisfied these conditions it was safe to test its flexibility using only the corner points of the parameter space. The corner point theorem was used as part of a synthesis procedure which built upon the ideas of design merging developed in the first paper [17]. Morari *et al.* [19] reformulated the procedure as a mixed integer program for the automatic synthesis of flexible heat exchanger networks.

The problem of optimal design for flexibility is addressed by Linhoff and Kotjabasakis [20]. The approach taken considers redesign by incorporation of contingency heat exchange area, but not changes in network structure. Alternative strategies for absorbing disturbances into the network are identified by the use of sensitivity tables. The best strategy is then chosen by considering the economic trade-offs. The optimisation is nested within another trade-off study that determines the optimum amount of flexibility. One of the main limitations of the procedures is its inability to deal effectively with non-convexities in the feasible region. The conditions necessary to satisfy Morari's 'corner point theorem' restrict significantly the types of problem that can be considered. To overcome this limitation Calandranis and Stephanopoulos [21] tried to identify the root causes of non-convexity in HEN problems. They observed that there were two basic mechanisms which led to non-convexity:

a) Intrinsic Mechanism: a feasibility constraint of a critical exchanger is non-linear. Critical exchangers are defined as the ones being closest to infeasibility.

b) Pinch Associated Mechanism: a result of disturbances being of sufficient magnitude to shift the location of the pinch to a new position in the network. The effect is equivalent to a discontinuity in the feasibility constraints.

Based on these observations it was possible to predict whether non-convexity would be a problem for particular disturbance cases, and to develop design strategies to work around these problems. In this way by giving specific consideration to the non-convexity mechanisms it is possible to provide the designer with a greater understanding of the limitations in a design.

An argument against treating the flexibility of a HEN in isolation is that the interaction between the network and the rest of the process can not be ignored. Calandranis and Stephanopoulos [21] have considered part of this problem. The disturbances entering a network are often inter-related as a result of the interconnections within the whole process. Therefore the disturbances that should be considered are not those entering the HEN but the root disturbances entering the process as a whole. Calandranis and Stephanopoulos address this by grouping streams into clusters such that streams in different clusters are independent of each other. The number of independent disturbances that can occur within a network is then equal to the number of clusters.

2.2 Dynamic Resilience

Flexibility analysis establishes an outer bound on the feasibility of process operation. It is focussed on determining whether a set of feasible steady states exists. Dynamic resilience is concerned with how well that steady state can be maintained in a dynamic environment. There are three primary criteria to be addressed when considering the dynamic behaviour.

- Stability: a primary consideration for any plant.
- Accuracy: fast, smooth response to the elimination of errors.
- Robustness: insensitivity of stability and accuracy to uncertainties such as model errors.

Limits are placed on these by the process design, the control structure, and the control algorithms. Dynamic resilience analysis is concerned with the limitations on dynamic performance inherent in the process design. Three approaches to this problem will be considered:

1. Use of the Internal Model Control (IMC) Structure: The IMC structure is used to provide a framework for analysis.
2. Extensions of Standard Controllability Definitions: Standard definitions of controllability from control theory are refined to provide useful analysis tools.
3. Application of Steady State Measures: Simple ratios such as the relative gain, based on steady state information, are considered for fast screening of options.

2.2.1 Use of the Internal Model Control Structure

The block diagram for the internal model control (IMC) structure is shown in figure 2.2. In this diagram the variables are as follows,

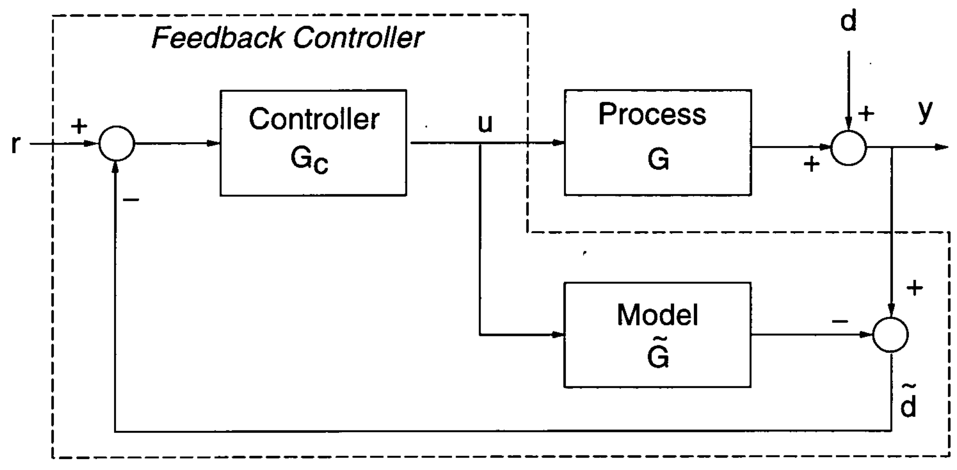


Figure 2.2: The internal model control structure

y	$\equiv$	process output vector
r	$\equiv$	setpoint input vector
d	$\equiv$	disturbance vector
G_c	$\equiv$	control system transfer matrix
G	$\equiv$	actual process transfer matrix
$\tilde{G}$	$\equiv$	process model transfer matrix
$\tilde{d}$	$\equiv$	estimate of disturbance vector

The rationale behind this structure can be seen by considering the relationship between u, d, and $\tilde{d}$, *viz*

$$\tilde{d} = (G - \tilde{G})u + d \tag{2.16}$$

If the error between process behaviour and modelled behaviour is nil (*ie.* $(G - \tilde{G}) = 0$) then the model comparison can derive the disturbances directly from the process outputs. The IMC structure was used in Morari [25] to highlight two fundamental observations concerning feedback control:

- Any feedback controller contains an approximate inverse of the plant transfer matrix.

- Closed loop control quality is limited by system invertibility.

The first observation can be confirmed simply through rearrangement of the block diagram. The second can be derived by considering the input/output relationship of this system in the Laplace domain, *viz.*

$$y = GG_c(I + (G - \tilde{G}))^{-1}(r - d) + d \tag{2.17}$$

If it is assumed that a perfect model is used (*ie.* $G = \tilde{G}$) then this simplifies to,

$$y = GG_c(r - d) + d \tag{2.18}$$

So to achieve perfect control we require,

$$G_c = G^{-1} \tag{2.19}$$

which is equivalent to the second statement above. Put in literal terms to achieve perfect control it is necessary that the behaviour of the process can be exactly predicted in proper time. The dynamic resilience therefore may be expressed as the ability to implement a feedback controller which is equivalent to the plant inverse.

A feedback controller is limited by the requirement that it be stable, and physically realisable. Dynamic resilience therefore is limited by the stability and realisability of the plant model inverse. Factoring G into its invertible and non-invertible parts, *viz.*

$$\begin{aligned} G &= G_+G_- && (2.20)\\ \text{where, } G_- &\equiv \text{Invertible part} && \\ G_+ &\equiv \text{Noninvertible part} && (2.21)\end{aligned}$$

Then the best possible control design is given by, $G_c = G_-$ and the best achievable control by

$$y = G_+(r - d) + d \tag{2.22}$$

Therefore the closer G_+ can be made to identity the better the achievable control. There is no unique solution to this factorisation. The best factorisation will depend on relative importance of outputs, and the sources of non-invertibility. The factors which prevent the full implementation of the plant inverse as a controller may be divided into two classes:

- Non Minimum Phase Elements: either time delays which to be inverted require predictive control, or right half plane (RHP) zeroes (often associated with inverse response) which if inverted create unstable control elements.

- Physical Constraints on the Manipulated Variables: Preventing the full control action to be applied.

Treatment of these two classes of non-invertibility follows different lines, and so will be discussed separately here.

2.2.2 Non-minimum Phase Elements

The term non-minimum phase (NMP) derives from frequency response analysis of systems. If a system exhibits non-minimum phase behaviour then there exists another system that can produce the same amplification but with a smaller phase lag. For example, a system which can be modelled simply by a constant gain involves no phase lag. When the system has a time delay added to it the gain remains the same but a phase lag is added to the frequency response. Thus a system with a time delay exhibits non-minimum phase behaviour.

As well as time delays a general source of non-minimum phase behaviour are systems that possess right half plane (RHP) zeroes. The zeroes of a plant are defined as the roots of the numerator polynomial of the plant transfer function. If this transfer function is inverted for use as the control algorithm these zeroes become poles (the roots of the denominator polynomial) in the control system.

The location of poles in the complex plane is one of the common forms of dynamic analysis used in control theory. As illustrated in figure 2.3 a pole which

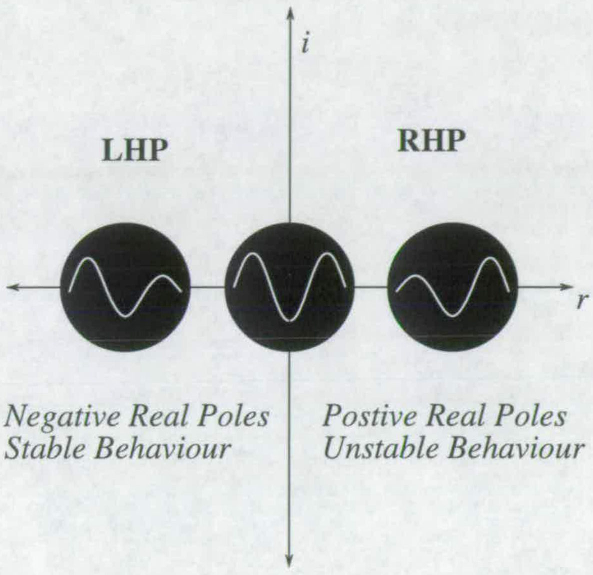


Figure 2.3: Correspondence between pole locations and dynamic stability

has a positive real part (*ie.* they lie in the right half plane of the graph) indicates unstable behaviour. To avoid this instability the RHP zeroes of the plant must not be transformed into poles of the controller. Thus right half plane zeroes in the system model restrict the use of a plant inverse as part of the control scheme.

A characteristic indicator of RHP zeroes is the presence of an 'inverse response' in the process behaviour. An inverse response is a response whose final steady state offset is in the opposite direction to its initial response. The cause of this is normally competing dynamic effects (see figure 2.4) one of small magnitude but high frequency, the other of larger magnitude but lower frequency. An example of inverse response is the behaviour of liquid level in a distillation column reboiler when the heat input is increased. Initially the increase in vapour from the reboiler

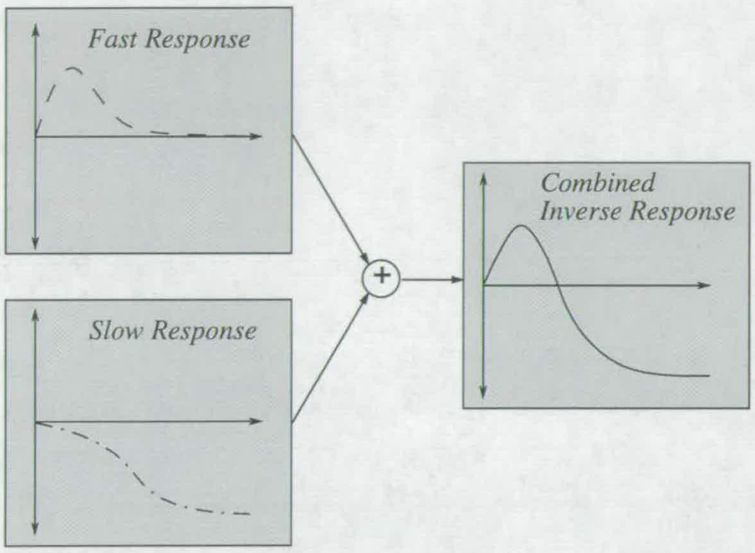


Figure 2.4: Combination of slow and fast dynamics forming an inverse response

can cause liquid on the lower trays of the column to spill over into the reboiler, increasing the liquid level. Eventually the increased holdup in the reboiler will be boiled off by the increased heat input, the final effect being a lowering in the liquid level. A controller using the heat input to regulate the level would, in this situation, continuously increase heating in response to the the initial inverse response of the level. In this way the inverse response can drive the controller to saturation very quickly.

The definition that has been given for RHP zeroes applies to the single input single output case. In the multi-variable case the process is represented by a plant transform matrix $G(s)$. The zeros of $G(s)$ are defined as the values of s for which the rank of $G(s)$ drops below its nominal rank. An implication of this definition is that the presence of RHP zeroes in a non-square system is rare. Such systems have an excess of control variables and it is unlikely for them to lose rank.

The treatment of non-minimum phase elements is tackled by considering the possible factorisations of G into its invertible and non-invertible parts. Arkun [26] proposes a systematic factorisation procedure to generate a set of feasible controllers, G_I, defined by

$$\begin{aligned} G_I &= \left\{ G_-^{-1}\, F \text{ , s.t. } F \text{ is proper \& stable} \right\} && (2.23) \\ \text{where } F &= \text{The Feedback Filter} \end{aligned}$$

this leads to the following input output relationship,

$$\begin{aligned} y &= G_+ Fr + (I + G_+ F)d && (2.24) \\ &= Hr + Dd && (2.25) \\ \text{where, } H &\equiv \text{setpoint sensitivity} \\ D &\equiv \text{disturbance sensitivity} \end{aligned}$$

Since NMP elements cannot be removed from the control system then they will appear in certain combinations of G_+F. So studying H and D will give an indication of whether the desired performance can be achieved. However, since there is no unique factorisation of G there is a degree of freedom in the selection of G_+ and so of F. It is suggested by Arkun [26] that this be "judiciously used" by the designer to generate control systems giving the required nominal performances.

An alternative approach is used by Morari *et al.*[27, 28]. Their aim is to analyse the nature of the limitations imposed by NMP elements, and how the position and magnitude of these influence the selection of optimal factorisations. The derivation of an optimal factorisation requires the definition of a performance measure. The measure depends on the weightings placed on each of the outputs. Since it is not feasible to consider every possible combination of weightings a completely general study is impractical. Instead Morari and coworkers consider special cases such as totally decoupled control, or perfect control on one output. Both time delays [27] and RHP zeroes [28] have been considered by this method, with the following results:

- Time Delays : Routines to find the maximum necessary delay and the minimum possible delay for each output have been derived. Also general insights are given to guide the designer in reducing the effects of delays.

- RHP Zeroes : The concept of a 'zero-direction' is developed that indicates the outputs with which a zero is predominantly aligned. To shift the effect of a zero from these outputs will result in significant interactions.

The limitation to the work of Morari and coworkers work is that it is only capable of treating time delays and RHP zeroes separately. In comparison the factorisation proposed by Arkun addresses both time delays and RHP zeroes simultaneously but does not try to really analyse the problems. Both approaches assume that there are no limitations to the range of control action that may be applied a problem which will be considered next.

2.2.3 Treatment of Constraints on Control Action

An important physical constraint on control is the range of control action possible. A first step in evaluating the impact of such constraints is to determine what range of control is required for perfect control. For the IMC control structure the control action u is given by

$$u = G_c(r - d) \tag{2.26}$$

Taking the moduli gives

$$|u| \leq |G_c|\,|(r - d)| \tag{2.27}$$

A Euclidean norm may be used for the vectors, but for G_c a compatible matrix norm is required. Morari [25] uses a spectral norm which is defined as,

$$|G| = \max_i \lambda_i^{1/2}(G^*G) \tag{2.28}$$

$$\begin{aligned} \text{where, } G^* &\equiv \text{complex conjugate transpose of } G \\ \lambda(G) &\equiv \text{Eigenvalue of } G \end{aligned}$$

The spectral norm corresponds to largest singular value of G (sometimes called the principal gain). It may be shown that,

$$\sigma_m(G)|u| \leq |Gu| \leq \sigma_M(G)u \tag{2.29}$$

$$\begin{aligned} \text{where, } \sigma_m(G) &\equiv \lambda_{\min}^{1/2}(G^*G) \\ \sigma_M(G) &\equiv \lambda_{\max}^{1/2}(G^*G) \end{aligned}$$

So substituting G_-^{-1} for G_c will give,

$$\begin{aligned} |G_c| &= |G_{-1}| && (2.30) \\ &= |G_+ G^{-1}| && (2.31) \\ &\leq |G_+||G^{-1}| && (2.32) \end{aligned}$$

If it is further required that the factorisation should give $|G_+| = 1$ then,

$$|G_c| \leq \frac{1}{\sigma_m(G)} \tag{2.33}$$

So, using singular values for the matrix norms, an upper bound on the control action required may be approximated by,

$$|u| \leq \frac{1}{\sigma_m(G)}|r - d| \tag{2.34}$$

If u is physically constrained, such that $|u| \leq |u|_{\max}$, and the system is scaled such that $|u|_{\max} = 1$ then to guarantee no control saturation the disturbance range is restricted by,

$$|r - d| \leq \sigma_m(G) \tag{2.35}$$

which may then be expressed as a frequency response curve, *viz*

$$|(r - d)| \leq \sigma_m(G(i\omega)) \tag{2.36}$$

where $\omega \equiv$ frequency

Such a plot gives a bound on the disturbance amplitude above which it is likely that control will be saturated. The analysis is analogous to the common SISO control criteria:

"Choose systems where the manipulated variable has a large effect on the controlled output."

A frequency response curve of minimum singular value provides a guide to the effects of physical constraints on dynamic resilience. The limitation of singular values is that they are scale dependent. The method is therefore most suitable to comparing similar systems. What this method cannot provide is an indication of what loss of performance would arise from occasional control saturation. In this respect singular values are a conservative measure. The interpretation of a singular value analysis is therefore not always straightforward.

2.2.4 Use of Extended Controllability Definitions

In state-space control theory, there are standard definitions for controllability. The most common is state controllability which is defined as follows:

State Controllability:

"A system represented by the state space model,

$$\dot{x} = Ax + Bu$$

is pointwise state controllable if, given any two states x_0 and x_1, there exists a time $t_1 > 0$ and an input u defined on the interval $[0, t_1]$ such that x is carried from x_0 at $t = 0$ to x_1 at $t = t_1$."

The implication for a system which is state controllable is that from any initial condition the system can be driven to any final condition using the set of controls specified by u. The definition has certain limitations for the analysis of dynamic resilience, *viz*

- The path from x_0 to x_1 is completely arbitrary.

- No account is taken of any bounds on variables.

- There is no information on regulation, *ie.* disturbance rejection.
- It is assumed that all states are to be controlled, generally an impractical requirement.
- The test does not give quantitative information on how controllable a system is.

Thus state controllability is in some respects an insufficient test and in other respects an over rigorous test.

An alternative to state controllability is functional controllability, which may be defined as follows for systems involving time delays:

Functional Controllability:

> "A system represented by a state space model and augmented with a set of output equations, *viz.*
>
> $$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$
>
> is functionally controllable if given an output trajectory, $\hat{y}$, which is zero for $y < T_{min}$ and satisfies certain smoothness conditions, there exists an input trajectory, $\hat{u}$, such that $\hat{u}$ generates $\hat{y}$ starting from an initial state at the origin."

The presence of the output equation makes it possible to relax some of the conditions of state controllability. Specifically it is not necessary for all the state variables to be controllable, only those which have an effect on the output vector y. A necessary condition for functional controllability can be found by considering the Laplace transform of the above state-space description

$$\begin{aligned} \bar{y} &= [C(sI - A)^{-1}B + D]\bar{u} && (2.37) \\ &= G\bar{u} && (2.38) \end{aligned}$$

$$\begin{aligned} \text{where, } G &\equiv \text{process transfer matrix} \\ &\equiv C(sI - A)^{-1}B + D \end{aligned}$$

So a sufficient condition for there to be a realisable input trajectory that will generate the desired output trajectory is that the process transfer matrix be invertible, as was demonstrated by Morari *et al.*using the IMC structure.

The requirement that $y = 0$ for $t < T_{min}$ is included because for any system involving delays a minimum delay must be allowed before independent trajectories may be specified for the outputs. It was proposed in Perkins and Wong [29] that this be used as a measure of the effects of the delays in a system on its dynamic resilience, and an algorithm was developed to determine T_{min}. The results of this analysis are closely related to those obtained from analysis of the IMC framework by Morari and Holt [27].

So far the results derived from controllability analysis have not differed significantly from those obtained from the use of the IMC framework, which seems to provide a more flexible analysis framework. Where controllability analysis comes into its own is in the application of structural controllability for which a general definition would be as follows,

Structural Controllability:

> "A system, X_0 , is structurally controllable if there exists a controllable system, X_1, which is structurally equivalent to X_0"

Two system matrices are structurally equivalent if there is an exact correspondence between the locations of the fixed-zero and arbitrary entries in each matrix. Structural controllability is not concerned with details of how state and control variables are related just whether a relation exists. The structural analysis is a simple form of cause and effect analysis. An *occurrence matrix* is used to represent the cause and effect relationships of the dynamic system. The occurrence matrix has columns which correspond to the variables and rows which correspond to the equations relating them. The presence of a state variable in an equation is marked by an entry at the appropriate position in the matrix.

The necessary conditions for a system to be structurally controlled are well established. Also the reasons for a system to fail the structural controllability test have been shown by Johnston and Barton [30] to be related to one of three problems:

- A contraction exists in the cause and effect relationships (*ie.* it is not possible to independently specify all of the outputs).
- Not all outputs are accessible from manipulated variables.
- Access to one output is via another output state (*ie.* controlling one output is only possible via another output so they cannot be controlled independently).

An enhanced version of structural controllability is functional-structural controllability which augments the occurrence matrix with rows corresponding to the output equations of a system and columns corresponding to the additional variables. An augmented matrix is used by Perkins and Russell [31] as an analysis tool. The representation was found to provide a better insight into the structural limitations of a system. It was observed that the conditions for structural controllability define physically meaningful cause and effect paths.

Structural controllability in this form is only able to confirm whether a suitable set of cause and effect paths exist for control. On its own this does not represent a very powerful analysis tool. In order to extend the usefulness of structural controllability an adaptation of the augmented occurrence matrix is developed. Each non-zero element of the occurrence matrix is replaced by a value representing the delay associated with that cause and effect relation. The effect of the location of time delays on dynamic resilience are studied with this matrix.

Using this matrix it was possible to determine the same minimum delay that Perkins and Wong [29] identified using functional controllability. In addition, based on the principles of structural controllability, cause and effect paths could be traced out in the delay matrix to identify which were the limiting delays within the system. For analysis of delays the augmented matrix is a useful tool, having a simple physical interpretation which makes it easier to pin point appropriate process modifications.

Another time-domain measure has been proposed by Carvallo *et al.* [32]. In this case controllability is quantified as the minimum time necessary to overcome the worst expected disturbance or setpoint change. Methods have been developed to calculate this measure for linear deterministic systems subject to process constraints and delays. The development of a method for stochastic models is also mentioned.

2.2.5 Application of Steady-state Interaction Measures

The problem with most techniques for dynamic analysis is that they are complex and require detailed information which is only available at a late stage in design. It would be useful to have simpler techniques which could give an indication of possible dynamic problems at the earlier stages of design. A tool with such potential is the relative gain array (RGA) which provides a measure of the possible interaction between control loops. The advantage of the RGA is that it is based on steady state information alone. The use of the RGA has been subject to controversy for two principal reasons:

a) By using only steady state information, the results can lead to false conclusions due to dynamic interactions.

b) It only considers setpoint perturbations, whereas in processes it is more common for other external sources to be the principal source of disturbances.

The first limitation is commonly tackled by using one of the dynamic interaction measures, but these are of varying value and defeat the aim of using steady state measures. The second limitation however has received less attention. An interesting variation on the *RGA* is the Relative Disturbance Gain (*RDG*) discussed in McAvoy and Marlin [33] . The *RDG* may be viewed as a more general form of relative gain in which the source of perturbation is not restricted to setpoint changes. For the simple case of two interacting control loops, $u_1 \rightarrow y_1$ and $u_2 \rightarrow y_2$, under a disturbance d the *RDG* for u_1 is calculated as,

$$RDG_{u_1} = \frac{gain\ of\ u_1\ wrt\ d,\ with\ both\ control\ loops\ closed}{gain\ of\ u_1\ wrt\ d,\ with\ only\ u_1 \rightarrow y_1\ loop\ closed} \tag{2.39}$$

$$= \frac{\partial u_1/\partial d\,|_{y_1,y_2}}{\partial u_1/\partial d\,|_{y_1,u_2}} \tag{2.40}$$

Using the *RDG* it is possible to identify where the greatest interactions exist, but it does not give a quantitative measure of the loss or gain in performance. McAvoy and Marlin [33] have investigated the relationship between the *RDG* and the integral of the error (IE). Two integral errors are considered for each loop, the IE_{sv} which is the expected IE if interaction did not exist, and the IE_{mv} which is the IE with the effects of interaction included. Based on simple process models, *ie.* only process gains, it was demonstrated that with P&I feedback control the ratio of IE_{sv} to IE_{mv}, referred to as the integral error ratio (IER), could be related to the *RDG*, *viz.*

$$IER = IE_{mv}/IE_{sv} = (RDG) * tuningfactor \tag{2.41}$$

$$\text{where, } tuningfactor = (P/I)_{sv}/(P/I)_{mv}$$

$$(P/I)_{sv} \equiv \text{ratio of proportional to integral action when tuned ignoring interaction.}$$

$$(P/I)_{mv} \equiv \text{ratio of proportional to integral action when tuned to account for interaction.}$$

The tuning factor is a measure of how much the controller had to be de-tuned due to interaction. To obtain an accurate measure of this would require the multi-variable system to be optimally tuned which would impair the general convenience of the procedure. To avoid this an approximate method for which the objective is simply good dynamic performance is used to estimate the tuning factor.

The integral error ratio provides a convenient and easily understood measure of the effect of interaction. A value greater than one indicates that the interaction is detrimental to performance, and a value less than one that the interaction is beneficial. Note that the IER of a decoupled system will be close to one, since its aim is to eliminate the interactions, which provides a simple indication as to whether decoupling is necessary.

The objective of developing a steady state measure has been to provide a convenient and simple method of identifying and quantifying the significance of process interactions. To achieve this it has been necessary to make several simplifications. The integral of error IE is a poorer measure of performance than the integral of absolute error (IAE) or integral of the square of error (ISE). Also some of the conclusions are based on the use of a specific control system which may not be the most appropriate for the problem. These assumptions though necessary to maintain simplicity can lead to deceptive results, but there is no easy way of judging when a fuller dynamic study should be used instead.

A frequency domain measure that is related to the RDG is discussed in Skogestad and Wolf [35]. The indicators developed are aimed at providing a controller independent measure of the sensitivity of a system to disturbances. The measures do not account for RHP zeroes or time delays but are seen as a complement to methods that have been developed to address those specific aspects.

2.2.6 Dynamic Robustness

In the techniques discussed up to this point there has been the implicit assumption that an accurate dynamic model of the process is being used and that plant behaviour will not deviate from that predicted by the model. The reality is generally short of the ideal for several reasons, key amongst these being:

- Most analysis techniques are based on linear models, while real processes are non-linear. Linearisation reduces the non-linear models to a form which can be studied but results in uncertainty over the linear coefficients.

- Even for systems well approximated by linear models a change in operating conditions will result in a change in the model parameters.

- Few processes are completely understood, particularly their high frequency dynamics, so there is always a certain amount of "genuine" uncertainty.

In view of these factors it is important to ensure that a process is stable and performs satisfactorily not only for the nominal dynamic model but also for all realisable models. Research so far has only investigated the requirements for robust stability. Robust stability is a problem that is more relevant when there are only occasional plant perturbations, and it is only important that the plant remain stable through these disturbances. Robust stability is also an important precursor to the study of robust performance, a significantly more complicated problem that as yet has no convenient solution.

A key step in assessing robustness is the selection of a suitable model for the uncertainty. In Morari and Skogestad [36] it is observed that there is a distinct trade-off between the rigour of a robustness study and the value of the results. For example, simple robustness bounds could be obtained from crude uncertainty descriptions, but such bounds tend to be misleading, and often difficult to attribute physical significance to. Using more rigorous descriptions of uncertainty these limitations can be overcome but the study of such uncertainty models involves significantly greater effort. Thus the choice of uncertainty model does to a large extent determine the value of the results obtained, and a certain amount of experience is required to select an uncertainty description of suitable detail.

The simplest uncertainty model that is commonly used is a lumped or 'unstructured' uncertainty which groups all the uncertainty together into one uncertainty matrix, L. Unstructured uncertainty may take several forms, *eg.*

$$\begin{aligned} \text{additive}: \quad G &= \tilde{G} + L_A && (2.42)\\ \text{multiplicative input}: \quad G &= \tilde{G}(I + L_I) && (2.43)\\ \text{multiplicative output}: \quad G &= (I + L_O)\tilde{G} && (2.44) \end{aligned}$$

The degree of uncertainty is normally specified in terms of a bound, l, on the magnitude of the perturbation matrix, L, given here in terms of the frequency response,

$$|L(i\omega)| \leq l(\omega)\ \forall\omega \qquad (2.45)$$

These models describe a region around $\tilde{G}$ that will include the set of realisable plants and also other plants which may not be realisable, thus its description as 'unstructured'. The analysis of robustness must therefore assume that the perturbation will occur in the worst direction for the plant. The conditions for robust stability based on these types of uncertainty description generally involve the condition number, ε, of the process transfer matrix,

$$\varepsilon(\omega) = |G^{-1}(i\omega)|\,|G(i\omega)| \qquad (2.46)$$

For example, in Morari [25] (where singular values were used for the matrix norm) it was shown that for input multiplicative uncertainty a sufficient condition for robust stability is,

$$\begin{aligned} \varepsilon(\tilde{G}) &< 1/\{\sigma_M(F)l_I(w)\} && (2.47)\\ \text{where, } \varepsilon(G) &\equiv \sigma_M(G)/\sigma_m(G) \\ F &\equiv \text{Filter Function} \end{aligned}$$

Good performance would require that $F = I$, but to guarantee stability when the condition number is small, or the magnitude of uncertainty is large, would require that $F \longrightarrow 0$ (*ie.* no feedback). The condition number is therefore an indicator of the robust stability of a process. The main difficulty is the dependence of the matrix norms on the scaling of the matrix. The problem of selecting an appropriate scaling is discussed by Perkins and Wong [29] who also propose the use of a condition number as a guide to robustness. In their work a different matrix norm is used for which the optimal scaling problem has been solved.

The weakness of this unstructured approach is, as mentioned earlier, that it tends to be over conservative. It is necessary to assume that the uncertainty will occur in the worst way, irrespective of whether this is physically realistic. Also it is difficult to specify $l(\omega)$ from the physical uncertainty bounds. A more structured approach is used in Arkun *et al.* [37] to derive the following condition for robust stability,

$$(1-\alpha)\frac{1}{\varepsilon_1(\tilde{G})\varepsilon_2(\tilde{G})} + \alpha\frac{1}{\varepsilon_1(\tilde{G})^2} > (l(\omega)\sigma_M(H))^2 \tag{2.48}$$

where,

$$\begin{aligned}
\alpha &\equiv \text{error projection}\\
\varepsilon_i &\equiv \sigma_{M,i}(\tilde{G})/\sigma_m(\tilde{G})\\
\sigma_{M,i} &\equiv \text{i'th largest singular value}\\
l(\omega) &\equiv \max_{p\in P}\sigma(L)\\
P &\equiv \text{region of uncertainty in parameters}
\end{aligned}$$

There are two improvements in this over the totally unstructured approach. The most obvious improvement is in the calculation of $l(\omega)$. Its value is derived from the maximum magnitude L can attain within the uncertainty region, P. The second improvement is less obvious and is related to α which is a measure of the projection of the model error onto the most sensitive direction of the closed loop system. If, by good design of a filter, this projection can be made smaller then a greater amount on model uncertainty can be handled by the system.

An interesting element of this work was the use of a symbolic equation manipulation package called 'MACSYMA' to carry the key design variables through to the transfer functions and so facilitate a sensitivity study to help in directing process modifications. Two robustness indices are incorporated into a multi-objective optimisation in Arkun and Palazoglu [38], trading off robust stability against economics. The value of this procedure is unclear since robustness is only one aspect of dynamic resilience, and the procedure ignores the effect of robust stability requirements on nominal performance.

While the work of Arkun does take more account of the structure of the uncertainty, it still groups all sources of uncertainty into a single matrix. A more complete model would consider each source of uncertainty as a separate source of perturbations. The analysis of such models has been considered by Morari and Skogestad [36] based on treating each perturbation source as an additional feedback path. To specify the tightest possible bounds on the robust stability conditions the structured singular value (SSV) is used. The problem with this approach is that it involves significantly more effort to develop the complete uncertainty model and to evaluate the SSV.

2.3 Operating Procedure Synthesis

Both flexibility and dynamic resilience focus on the operability of the plant during its production phase of operation. The analysis methods determine limits on the demands the process can be subject to and still maintain regulatory control. While it is undesirable to exceed these limits, there are certain activities in which this is unavoidable such as startup and shutdown. For these activities explicit operating procedures are required so the operations are performed safely and efficiently.

To support these operating procedures it is not uncommon for modifications to the process design to be required. As with the design of the control system if the operating procedures are not considered until the later stages of design then options for redesign of the process will be restricted. Integrating the design of the process and its operating procedures has received relatively little attention. The principles source of insight on this topic come from research on operating procedure synthesis. A systematic procedure for planning special operations can help to evaluate the feasibility of a process design for the required operations.

2.3.1 A Definition of Operating Procedure Synthesis

Operating procedure synthesis is defined as determining the sequence of actions which will drive a process from an initial state to a goal state (*eg.* from off line to normal operation). The transition is subject to both physical constraints (*eg.* a valve must be open before there can be flow through it) and safety constraints (*eg.* at no time should an explosive mixture be formed).

If the two states are close then this could be treated as a normal setpoint tracking control problem. In the operations of interest, however, there is a wide separation between the initial state and the final state. The transition will generally involve several discontinuous modes of behaviour making it impractical to manage by means of a single control scheme. An operating procedure decomposes the transition into small steps that can be more easily managed.

Synthesis of operating procedure is a combinatorial problem since there are many possible orderings of the operations that would achieve the same overall transition. Searching the space of possible sequences for ones that are physically feasible, safe and efficient is a non-trivial problem.

2.3.2 Synthesis Methods

Some of the earliest studies of operating procedure synthesis is provided by Rivas *et al.* [41] and Rivas and Rudd [42]. The particular emphasis of their work was the determination of safe sequences of valve operations in safety interlock systems. The method developed represented the process as a flow network and used symbolic logic to model the effect of valve operations. Synthesis of a valve sequence was achieved by constructing a hierarchy of goals which identified the key operations and the order in which they should be performed. The valve operations synthesised from this goal hierarchy were tested using the qualitative models. The limitation of this work is its use of a 'generate and test' algorithm, an approach which is not suitable for combinatorial problems of any significant size.

Fusillo and Powers [43, 44] developed a more general methodology for operating procedure. The problem of searching a large space of alternative sequences is decomposed in two ways:

- Divide the process into isolated sub-systems
- Break down the transition path into simpler transitions between intermediate stationary states.

Symbolic models were also used in their procedure but their form allowed for more general specifications of constraints. The sequencing of the operations is achieved by means ends analysis, comparing the current state with the goal state and selecting the operation which can reduce the most important difference. The emphasis of the work was on the sequencing algorithm. Selecting appropriate system decompositions is not considered. The work was later extended to support planning of purge operations [45] which involved the incorporation of extra knowledge on possible purge operations and purgatives.

Lakshmanan and Stephanopoulos [46] provide a more rigorous study of the modelling requirements for operating procedure synthesis. A hierarchically structured modelling framework is developed and used in the development of a non-linear planning methodology [47]. Their work was followed by a specific study of mixing constraints [48] focussing on how such general constraints can be transformed into specific temporal constraints that are used by the planner.

Aelion and Powers [49] have developed an approach for automating the retrofit design of a process in order to make operating procedures feasible. The approach decomposes the problem into two parts. Established synthesis methods are first used to determine whether the procedure is possible with the unmodified design. If not then a *structural planner* is used to modify the design and improve the feasibility of the operation. The structural planner works by redesigning the process so that additional intermediate states are possible. The intermediate states are intended to provide a feasible path for the operating procedure.

2.4 Summary of Operability Analysis

The range of techniques for operability analysis is broad and complex. The goal for most methods is to identify fundamental bounds on process operation that are imposed by the process design. To assess the value of these methods for application during process design the following issues should be considered:

- Value in comparison of designs: Often the primary issue to the designer when considering operability is which of two otherwise equivalent designs is better. An understanding of the significance of an operability measure is necessary to determine what constitutes a significant difference.

- Degree of insight: Methods which can direct the designer to ways of improving operability are naturally more desirable.

- Ease of application: If the formulation and computation of an operability measure is too complex then it is less likely to be employed. Also if the results are difficult to interpret then the value of the method is diminished.

2.4.1 Flexibility

Flexibility measures are the most straight forward for a process designer to understand. The flexibility index developed by Grossman is a measure which can be used not only for comparison of designs but also to identify when overdesign exists. The approach relies on being able to identify the critical points of the parameter space which is non-trivial for feasibility regions which are not convex. The complexity of the resulting mathematical problems suggests that they are most appropriately applied at the final stages of design.

A criticism of automated design methods is that they do not provide the designer with a lot of insight. The work of Calandranis and Stephanopoulos [21] is an example of a more focussed study. Their work provides useful insights into the operability problems of heat exchanger networks. Unfortunately similar analysis for other aspects of process design is not available.

2.4.2 Dynamic Resilience

The methods developed for studying dynamic resilience are more difficult for a process engineer to employ. The insights derived from Morari's analysis of the IMC structure are a valuable basis for evaluation of the inherent limitations that a process design imposes on control. Their main drawback is their foundation on multi-variable control theory and frequency domain analysis, both unfamiliar domains to most process engineers. Frequency domain plots of singular values and condition numbers are a difficult basis for the process engineer to discriminate between process designs. Robustness has received a lot of academic attention, however an inspection of the case studies that have been published [50, 51, 52] shows that most applications are on the synthesis of control structures. There are few examples of its application as a dynamic resilience measure to improve process design.

Simpler analyses have been proposed which are derived from the state-space concept of controllability. Also the RDG is possibly one of the simplest measures for a process engineer to appreciate. In both cases the methods are focussed on particular controllability problems.

There is a general difficulty of relating measures of dynamic resilience to economic performance. Some guidance in this respect comes from case studies by Marlin *et al.* [53] and Barton *et al.* [54]. These studies apply cost analysis to determine where efforts should be focussed in retrofit control system design. The effect of poor dynamic resilience can affect economic performance in different ways. Where optimal operation is limited by constraints the effect of poor control is to force operation further from the optimal constraints. In other situations the effect of poor control may be in the cost of the control mechanism employed. Thus at present putting resilience measures on a cost basis relies on insight into the optimal operation of the process.

2.4.3 Operating Procedures

The brief review gives a taste of the challenges faced simply in finding suitable operating procedures for a fixed plant design. Particular gaps are systematic methods to decompose and re-integrate plants. The focus of synthesis methods is on operation planning for final designs. The use of these synthesis methods during process development to identify in advance the requirements for special operations has not been considered.

2.4.4 Aiding Operability Analysis by Integrating Design

There are now many tools available to the process designer for evaluating the operability of a process design. Putting these tools together to derive a balanced conclusion is a non-trivial task for the designer. In particular the tools for studying dynamic resilience produce results that are difficult for a process engineer to interpret. Instead of requiring the process designer to predict the needs of the control system an alternative approach is to involve the control system designers in the evolution of the process. The control engineers will be better placed to provide feedback on the controllability of a design. Further if the control system is developed in step with the process it is possible that more focussed operability studies can be made. The next chapter will discuss how concurrent design of the process and control system may be achieved.

Chapter 3

Concurrent Hierarchical Process and Operating System Design

3.1 Introduction

The ability to evaluate the operability of a process design is only the first step towards improved integration. Full integration is best supported by the concurrent design of the process and control system. The potential benefits of this are:

- Improved understanding of the process by the control engineer.
  Developing the control system in step with the process gives the control engineer the chance to absorb the detail of the process design incrementally rather than being faced with the information overload of a fully detailed flowsheet.

- Easier Analysis of Operating Requirements.
  A knowledge of the intended control strategy helps to focus operability analysis and clarifies the process operating requirements.

- An improved forum for informed negotiation between process and control preferences involving both process and control system designers.

Concurrent design of the process and control system is a significant departure from conventional design practice. The hierarchical model of design is proposed as the basis of concurrent development. A framework for the hierarchical design of a *process operating system* is presented which provides a basis for integrating the management of process operations including the regulatory control, startup and shut down of the process.

3.2 Definition of a Process Operating System

The *process operating system* is the complete collection of control schemes, alarms and procedures used in managing process operation. By broadening the focus from solely regulatory control to complete operations management an integrated strategy for process operation can be developed. An additional goal in taking this broader perspective is to identify operating requirements as early as possible in the design process.

There are two basic functional requirements for an operating system failure management and process optimisation.

Failure Management: Failure management does not only refer to equipment failure but also encompasses such failures as exceeding safety limits or quality constraints. There are two aspects to failure management, failure prevention and failure recovery. The goal of failure prevention is to minimise the occurrence of failures. The goal of failure recovery is to minimise the effect of a failure. Where potential failures can be predicted accurately the emphasis can be placed on failure prevention. In many cases the prediction of a failure event will involve too much uncertainty in which case consideration must be given to failure recovery as well as prevention.

Process Optimisation: The second function of the operating system is to optimise the operation of the process. The objective of the optimisation will normally be an economic one. There will also be phases of operation when a different objective will be dominant, for example emission minimisation.

In addition to these basic functional requirements there are other criteria that an operating system design must satisfy:

- Robustness: Most control algorithms either utilise an explicit model of the process or require to be 'tuned' to one. Since the models are rarely precise it is important that the performance of the operating system is insensitive to model inaccuracies. Robustness is the ability of the system to maintain safety and performance despite model uncertainties.

- Implementation: The design of an operating system must take into account what it is practical to measure and control. Also the capabilities of the control hardware must be considered. The solution of complex optimisation may not be possible in real time. If the control hardware cannot run the algorithm at sufficient speed then less complex algorithms or models need to be adopted.

- Clarity: Systems in which the functions of components and their interrelations are easier to perceive are also easier to maintain and update. One way of improving clarity is to employ explicit models where possible. For example, where a derived property is required rather than implicitly encoding this derivation into the algorithm formulation, utilise an intermediate state variable and separate the property derivation from its use in the control algorithm.

Conflict amongst these criteria is expected. They are listed in their approximate order of importance. The order is not an absolute one and in practice an operating system design must achieve a balance. Hierarchical problem solving can help by providing a structured approach to balancing complexity against performance. The details of hierarchical problem solving and its application in design are considered next.

3.3 Hierarchical Problem Solving & Design

Hierarchical problem solving is a method of progressing from the abstract or 'high level' solution to a final detailed, *ground* solution in an incremental fashion. The approach makes it easier to manage the complexity in a design. There are two key aspects to the approach:

- *Abstraction:* The use of reduced models which help focus on the key decisions to be made.
- *Decomposition:* Dividing a design into parts that may be independently tackled.

A useful general perspective on the use of abstraction for problem solving is provided by Sacerdoti [56]. The focus of his work was on supporting domain specific knowledge in a general purpose planner. It was observed that a common aspect of human problem solving was the grading of decisions according to their relative importance. The grading leads to a hierarchy of abstraction spaces and decisions. The most ground problem contains all the variables of the problem. Each increment of abstraction corresponds to a reduction in the number of decision variables. Problem solving procedure works from most abstract to most ground in a recursive process:

1. Solve problem for the current level of abstraction,
2. Fix the decisions made at this level,

3. Formulate the problem for the next more ground level of abstraction subject to these fixed decisions,

4. Repeat 1 to 3 until all decisions are fixed.

The solution of a simple problem using purely abstraction is illustrated in figure 3.1.

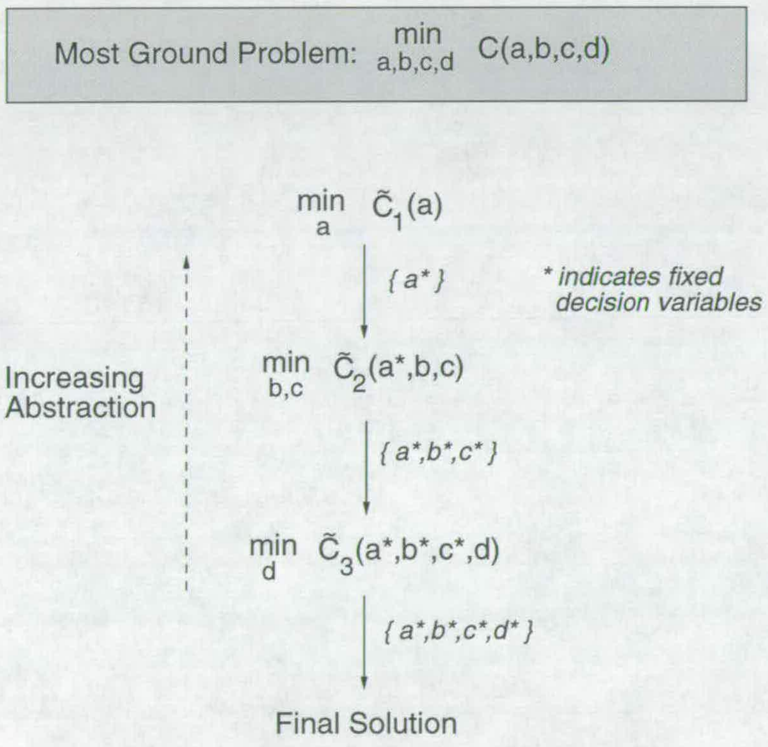

Most Ground Problem: $\min_{a,b,c,d} C(a,b,c,d)$
$\min_{a} \tilde{C}_1(a)$
$\{a^*\}$
* indicates fixed decision variables
Increasing Abstraction
$\min_{b,c} \tilde{C}_2(a^*,b,c)$
$\{a^*,b^*,c^*\}$
$\min_{d} \tilde{C}_3(a^*,b^*,c^*,d)$
$\{a^*,b^*,c^*,d^*\}$
Final Solution

Figure 3.1: Problem solving using abstraction only

If the progression from abstract problem to ground problem can be made in small steps a solution will be reached without decomposition. Limitations on how a problem may be formulated restrict the step size. For example, in process modelling it is difficult to define an intermediate step between using a mass balance only and using a mass and heat balance. If it is not possible to reduce complexity sufficiently by abstraction then decomposition of the problem is employed.

Decomposition is the partitioning of a problem into separate, hopefully independent, parts. The resulting set of reduced problems are solved concurrently. The process of hierarchical refinement and decomposition is applied recursively to each problem. The simple linear decision hierarchy of figure 3.1 now takes on the structure of a decision tree as in figure 3.2.

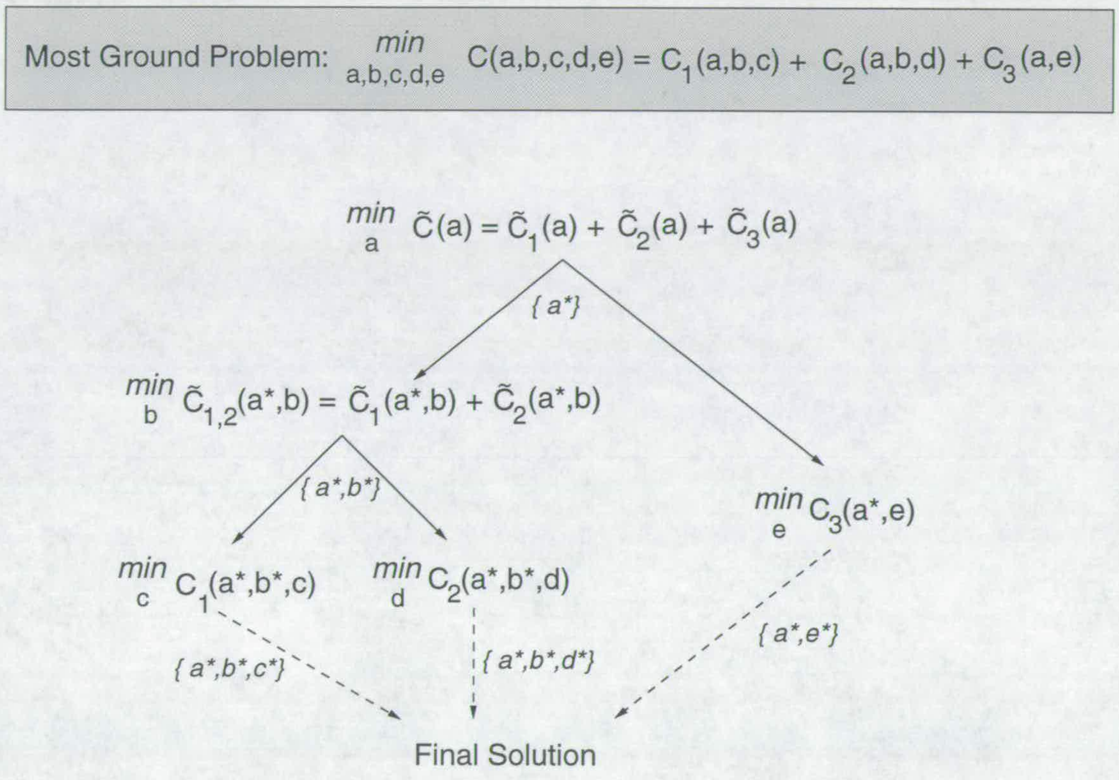

Most Ground Problem: $\min_{a,b,c,d,e} C(a,b,c,d,e) = C_1(a,b,c) + C_2(a,b,d) + C_3(a,e)$
$\min_a \tilde{C}(a) = \tilde{C}_1(a) + \tilde{C}_2(a) + \tilde{C}_3(a)$
{ a*}
$\min_b \tilde{C}_{1,2}(a^*,b) = \tilde{C}_1(a^*,b) + \tilde{C}_2(a^*,b)$
{ a*,b*}
$\min_e C_3(a^*,e)$
$\min_c C_1(a^*,b^*,c)$
$\min_d C_2(a^*,b^*,d)$
{ a*,e*}
{ a*,b*,c*}
{ a*,b*,d*}
Final Solution

Figure 3.2: Problem solving using abstraction and decomposition

The trade off in using a hierarchical procedure is between ease of solution and accuracy of solution. The primary issue is the significance of unmodelled interactions on the optimum solution. For example, in the simple problem represented in figure 3.1 the ideal model for $\tilde{C}_1(a)$ is one equivalent to

$$\tilde{C}_1(a) = \min_{b,c,d} C(a,b,c,d)$$

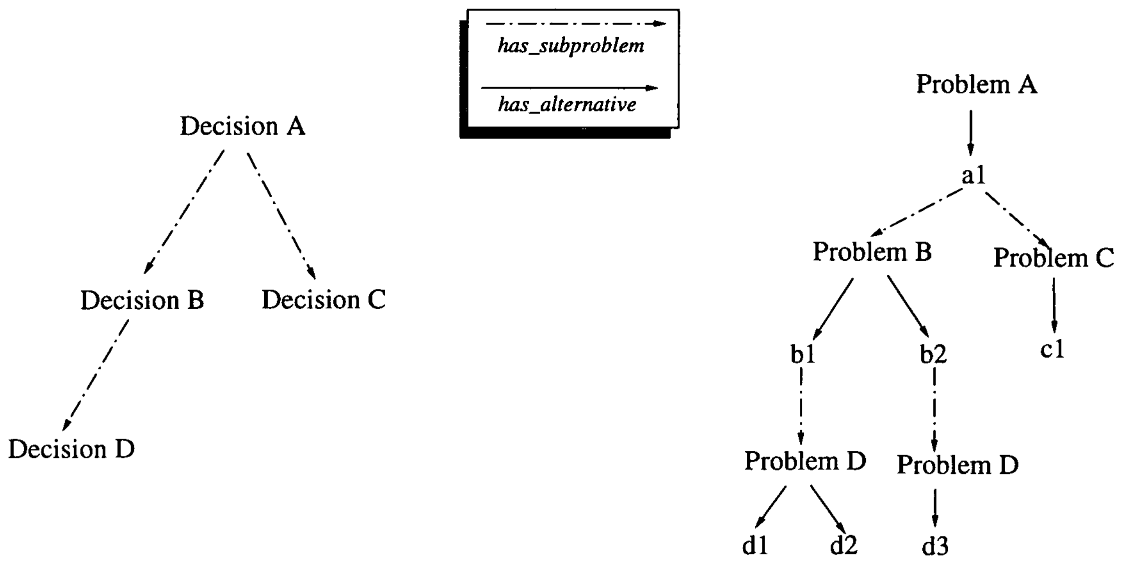
Achieving a balance between accuracy of abstraction and ease of solution is a key part of the hierarchical approach. Compensation for modelling inaccuracies can be made by treating established values for decision variables only as initial guesses. Refining approximate values as the degree of abstraction reduces can produce a final solution that is closer to the local optimum. Revising decision variables that have been made prior to a decomposition must be done with care as this will potentially introduce inconsistencies between the decomposed parts. To avoid this problem it is important that before decomposition the key interactions between the sub-problems have already been resolved.

3.3.1 Applying Hierarchical Problem Solving to Design

The assumption so far has been that a fully detailed model of the problem has been available from which abstractions are derived. For design this is not the case: the specification for a design evolves as the design evolves. For hierarchical design abstractions are derived from an understanding of prior designs. The decision hierarchy is elaborated as the design is elicited.

In using abstract models for a design significant uncertainty is introduced. At a preliminary level, therefore, it is not always possible to discriminate between a set of alternative solutions. The hierarchical structure is a useful aid for systematically exploring the design alternatives. The organisation of design alternatives is built upon the decision tree that is created during hierarchical design (see figure 3.3). Each decision node is now connected to a set of alternatives and each alternative then has a separate decision tree. The alternatives associated with nodes at the top of the hierarchy will lead to more distinct design solutions than those further down.

The need for decomposition is common in hierarchical design. However, with the limited advance knowledge of design structure it is difficult to account for all interactions before decomposition. In such situations maintaining consistency between the branches of the decomposition is particularly important. A simple approach to this is to recombine design branches when interdependencies have



(a) Original Decisions Hierarchy (b) With alternatives interposed

Figure 3.3: Representing solution alternatives as part of the decision hierarchy

to be resolved. In recombination specific alternatives from each branch of the decomposition of a decision node are selected and combined. The resulting detailed design is recorded as a new design alternative for that node. For example, figure 3.4 shows how the design alternatives from figure 3.3 are reorganised when

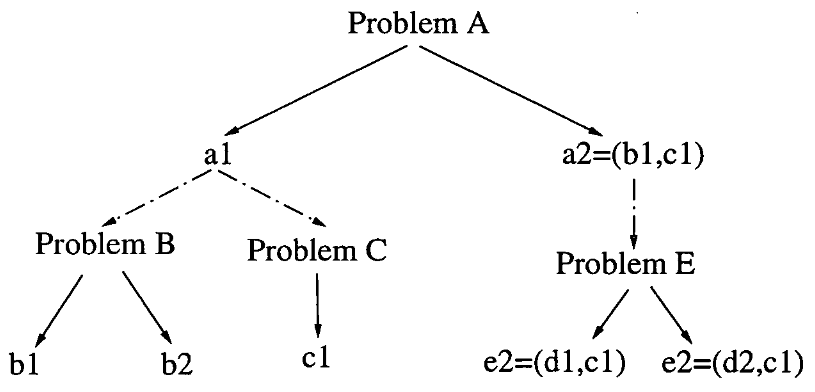


Figure 3.4: Recombining design alternatives.

the refinement of a design from level B to level D is dependent on design details associated with problem C. The extra detail is provided by recombining alternatives from the separate decomposition branches of problem B and C. Alternative *a2* is an example which is the result of combining design alternatives *b*1 and *c*1. The refinement of the recombined design (problem *E*) incorporates refinement of *b*1 from level *B* to level *D* with the additional detail required being taken from *c*1. Recombination also provides an opportunity to review the assumptions and models that led to the current design hierarchy. Validating the final design hierarchy is important to ensuring that the optimal design is found.

3.3.2 Hierarchical Process Design

Process design is an example of a domain where the methodology for hierarchical design is well developed. Douglas [57] provides a thorough treatment of hierarchical preliminary design. The levels of design defined in this approach are:

- Batch versus Continuous.
- Input/Output Structure: A single block representation of the process focussing on the primary material flows into and out of the process. An economic evaluation is restricted to material costs.
- Recycle Structure: Decisions are made on the reactor systems to be used and what materials are to be recycled or purged. The recycle structure decisions fix the product distribution entering the separation system.
- Separation System: The design of the separation system is itself treated in three parts, general structure, vapour recovery system and liquid separation system.
- Heat Integration.

The methodology can be used to rapidly generate a set of design alternatives. The division of the process into sections with specific functions provides a suitable basis for decomposition. Once the recycle structure is defined the reactor and separation systems can be developed concurrently. Likewise as the separation sub-sections are defined their design can be decomposed and developed concurrently.

Douglas [57] also provides shortcut models and heuristics to aid in the evaluation of design alternatives. Having suitable models to evaluate the process at many levels of abstraction is important to the effectiveness of hierarchical design. In developing a framework for concurrent design of the control system it is desirable to take advantage of the hierarchical structure present in process design.

3.3.3 Hierarchical Control System Design

The application of hierarchical design to develop the control system concurrently with the process design has received limited attention. Most studies of control system synthesis start from an assumption that the process is fully defined. An exception to this is the work of Ponton and Laing [58]. They demonstrate how a control system structure can be evolved in step with the evolution of the process. For each step in the Douglas [59] methodology a corresponding control step is proposed:

Process Design	**Control System Design**
Input/Output structure	Feed and product rate control
Recycle structure	Recycle rates & purges
Separation sequencing	Composition controls
Energy integration	Temperature and energy balance control
—	Inventory control

The procedure has been taught to chemical engineering students as part of their control course and applied in their design projects. Using a hierarchical approach was found to simplify the placement of the control loops. The relative importance of different control loops was also more readily understood. A criticism of the approach was that the final control system designs developed by the students required an excessive number of composition meters. What is typically missed is an extra level of refinement where required states are mapped to available measurements, if necessary by using inference algorithms.

While the procedure uses only structural analysis and is focussed on regulatory control it demonstrates the potential advantage of a hierarchical approach. The following sections will present a more general framework for hierarchical design of the complete process operating system.

3.4 A Framework for Hierarchical Operating System Design

Organising the operating system as a hierarchy is a good way to approach the balance between complexity and performance. From the work of Ponton and Laing [58], it is also seen that the hierarchical approach to process design can be utilised in the design of the control system. Operating system design will be based on developing a hierarchy of *operating tasks*. High level operating tasks perform the global optimisation and failure management for the process and plan on a long time scale. Intermediate tasks focus management of a small set of units and plan on a short time scale. The lowest level tasks will correspond to simple controllers which have little planning capability but have a fast response.

3.5 Structuring the Operating Task Specification

To provide a framework for defining the requirement of an operating task the formulation of a dynamic optimisation problem is considered,

$$\min_{u(t)} C[u(t)] = \int_{t_I}^{t_F} C_T(u,d,y,t) + C_F(u_{t_F},d_{t_F},y_{t_F},t_F) \tag{3.1}$$

subject to,	$\dot{y} = y'(u, d, y, t)$	*Physical Relations*
	$\dot{d} = d'(t)$	*Demands*
	$0 \leq g(u, d, y, t)$	*Operating Limits*

where,	u =	The set of control variables.
	y =	The measure of the process state.
	d =	The time dependent demands.
	$\{t_I, t_F\}$ =	Planning window of optimisation
	C_T =	State transition cost function
	C_F =	Final state cost function

Using this formulation as a guideline the specification of an operating task is structured as follows:

Objective: An objective function is necessary to relate control actions to performance. Included with the objective is the time scale or temporal scope of the task (*ie.* $\{t_I, t_F\}$). For tasks concerned with basic production optimisation this will take the form of a cost function. For the low level tasks which are essentially standard controllers the objective simplifies to minimising error. The primary function of some tasks will be failure management and for them the objective is to minimise the number of failures.

Constraints: Constraints on the operation of a process are divided into three classes,

- Physical laws: *eg.* heat and mass conservation and thermodynamic relationships. The physical laws restrict which variables may be manipulated independently.

- Physical Limits: limits which physically cannot be exceeded. For example, it is not possible for a composition to be greater than one or less than zero.

Similarly there are limits to flow that may derived from restricted valve ranges and pump capacities.

- Failure Conditions: for example safe temperature and pressure limits for a unit. The limit of a failure condition can be physically exceeded but the result would be a failure in operation.

While all three categories of constraint must be taken into account during design the failure conditions require particular attention. These must be actively managed otherwise there is no guarantee that they will not be contravened.

Demands: The demands are time varying events not under the direct control of the operating task which affect either the objective function or the integrity of the constraints. The design of the optimisation strategy is directed towards response to these demands. Demands arise from two sources,

- Supervisory Demands: targets or setpoints for operating variables that are given to the operating task from a higher level task. Most high level tasks do not change process operation directly but use supervisory demands to direct lower level tasks. This allows the low level tasks to perform local optimisation for the demands of shorter time scale.

- External Demands: the uncontrolled inputs to the system managed by the operating task which includes the knock-on effect of external demands not fully controlled by peer tasks. There is a one to many relationship between external demands and disturbance variables. Where possible separate external demands are used to differentiate between different causes of variation of a disturbance variable.

Understanding the time scale of each demand is important in deciding the level at which it will be addressed in the operating system.

3.6 Defining the Fundamental Operating Tasks

The starting point of operating system design is the root operating task. The specifications on this task that can be made at the start of design can only be vague:

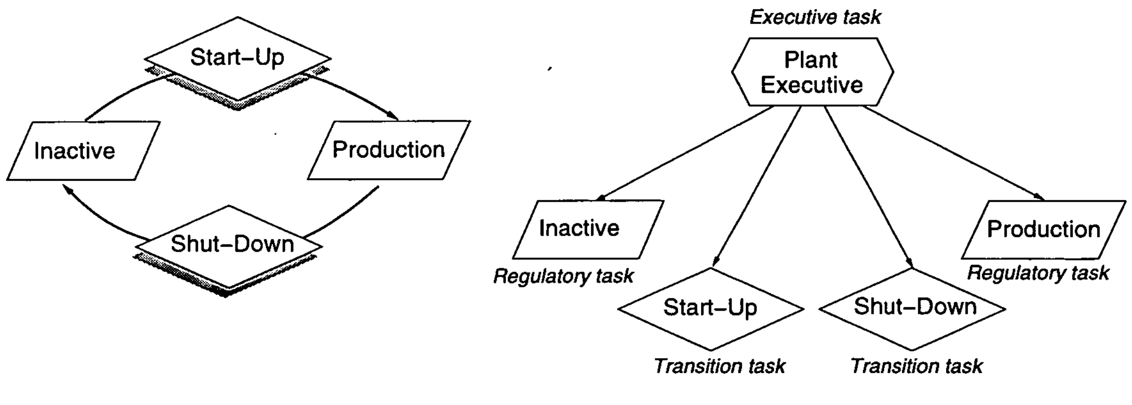
- Objective: fundamentally the objective for the operating system is to maximise profit from startup to shutdown.
- Constraints: As with the external demands it is difficult to determine the physical relations or limits without some knowledge of the process design. With respect to failure protection a general failure condition of hazardous operation is identified. It is not possible at this stage to specify the constraints that this implies.
- Demands: Without knowledge of the process structure it is difficult to pinpoint external demands and at this level supervisory demands do not exist.

The objective given here encompasses a broad range of operation that is rarely considered as a continuum. There are distinct phases of operation, which we define as the *operating modes* of the process. The difference between operating modes is related to either a discontinuity in process behaviour or a change in operating objective. A case for the latter would be when the set of active constraints changes. A case for the former would be a shift from operating to maximise profit to operating to minimise hazardous emissions. Different operating modes are likely to require different operating strategies and so different operating task structures. Identifying the fundamental operating modes for the process provides a valuable start on structuring the operating system.

For most plants there will be at least four modes of operation. The most considered phase of operation is the production mode when the process is running at its nominal design conditions. Multiple production modes may exist, for example when the process produces multiple grades of product. In order to reach a production mode the process must go through startup which is represented as a separate mode of operation. Likewise the shutdown of the process is treated as a separate mode of operation. Finally, an important, though not so obvious, mode to identify is the process when shutdown or in its inactive state. While the 'inactive' mode is not like to require any control it serves to define the start and finish conditions for process operation. The modes that have just been defined fall into two categories:

- Regulatory modes: The production and inactive modes are examples of regulatory modes. During these phases of operation the focus is on maintaining an optimal steady state.
- Transition modes: Startup and shutdown are transition modes. The emphasis during these modes of operation is more on achieving a fast transfer of the process from one state to another.

The mode decomposition can be viewed as a directed graph, as in figure 3.5(a),



(a) Graph of mode decomposition (b) Operating task hierarchy

Figure 3.5: A simple operations system structure

where the regulatory modes are represented as nodes and transition modes by arcs.

The basic operating modes of the process are used to define the fundamental operating task for the operating system. This provides the first decomposition of the root operating task. The resulting hierarchy of tasks is illustrated in figure 3.5(b). The operating tasks are divided into three classes:

- Regulatory Tasks: Regulatory tasks are associated with regulatory modes. The supervisory demands for a regulatory task normally take the form of conditions that should be maintained as long as that task is active.

- Transition Tasks: A transition task is similarly connected to a transition mode. The supervisory demands for a transition task specify changes of conditions which must be effected over a given period.

- Executive Tasks: The function of executive tasks are to control which mode of operation the process should be in and consequently which operating tasks should be active.

The basic decomposition that has been derived considers only the fundamental process requirements. The decomposition can be extended by consideration of the strategic operating requirements. For example in figure 3.6 an additional regulatory mode labelled the 'secure state' is defined. This mode is intended to provide a second option to full shut down in the case of minor failure events. Startup and shutdown are both split into two phases utilising the intermediate secure state. As a contingency, in the case of severe failures, a second mode for shutdown is added for which the emphasis in on achieving a complete shutdown quickly.

Another example of mode decomposition could arise for processes that have a wide range of throughput. Such processes may be more straightforward to design

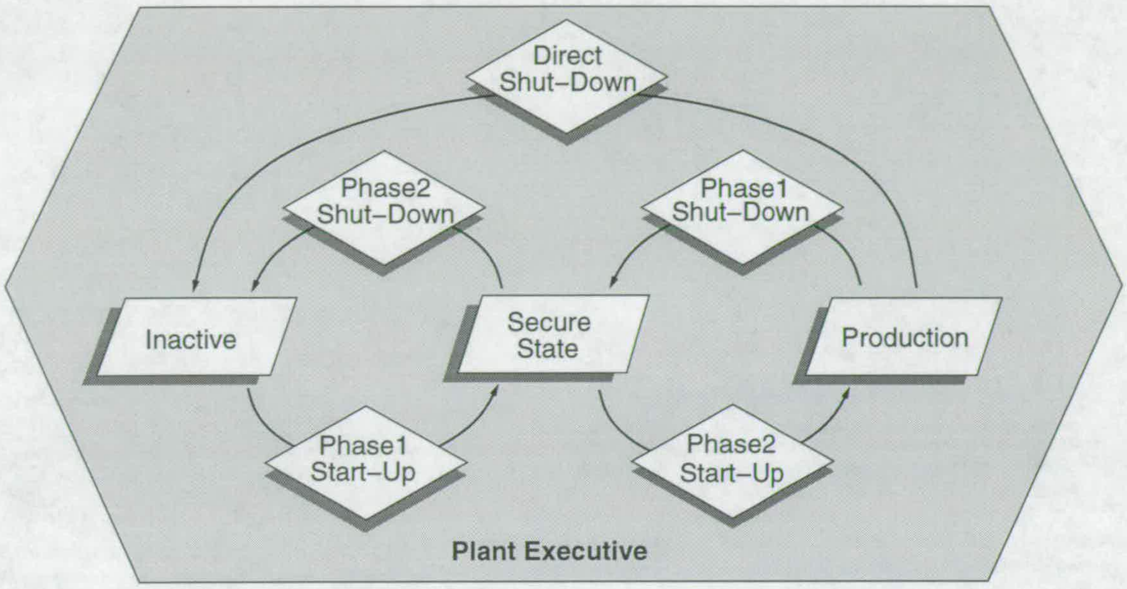


Figure 3.6: A more sophisticated preliminary operating system design

as parallel sets of units. The main set of units support a base level of throughput. The secondary set of units are brought on line only when throughput reaches the limits of the main set of units and are taken off line when production is turned down. The strategy can be represented as four operating modes, two regulatory modes for high and low production rate and two transition modes for startup and shutdown of the secondary set of units.

Defining the basic operating task structure is a valuable precursor to the initial process design. The range of operating activities that the process must support is more clearly identified.

3.7 The Functional Components of an Operating Task

The analysis discussed above establishes an outline strategy for process operation based on identifying the fundamental modes of operation and defining a corresponding operating task hierarchy. Each operating task is a separate control or optimisation problem. The set of controls, measurements, and algorithms that are employed to solve an operating task are defined as the *control scheme* for the task. To provide a framework for the design of control schemes an analysis of the necessary functional elements is considered.

The core function for a control scheme is optimisation. Design of this part of the control scheme requires the selection of a set of control variables, u, and an algorithm for determining their optimal value, $\dot{u} = u'()$. To assist design clarity optimisation models are formulated in terms of the supervisory and external demands, d_s & d_e, and a convenient set of state variables, x (*eg.* $\dot{u} = u'(u, d_s, d_e, x, t)$). The core structure of a control scheme is illustrated in figure 3.7

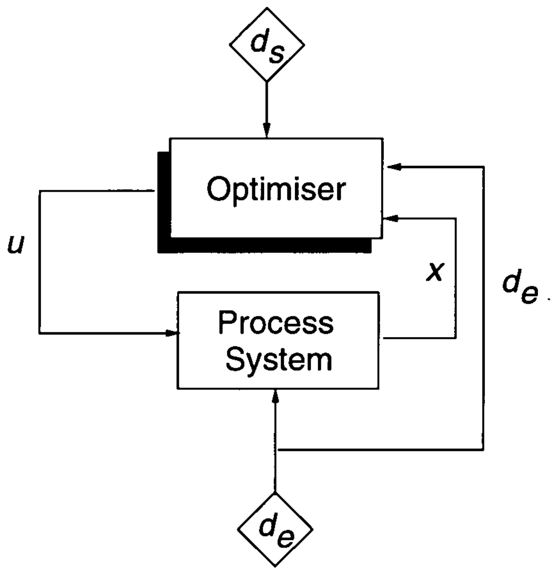


Figure 3.7: Core structure of a control scheme

Not all external demands will be available by direct measurement, and the state variables chosen as convenient for the optimiser may not all be readily measurable states of the process. Incorporating the relations between the desired properties and available measurements into the optimisation formulation compromises clarity. Instead a *property estimation* block is introduced. The function of this is to bridge the gap between what is desired as inputs by the optimisation and what is available from the underlying system. The revised structure is shown in figure 3.8. This

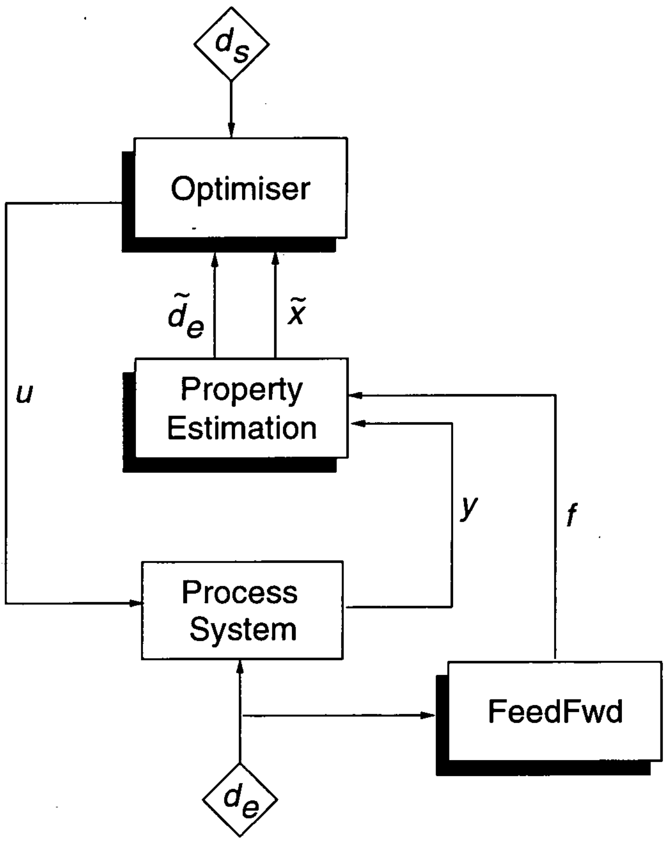


Figure 3.8: Control scheme structure with *property estimation*

structure also makes the distinction between disturbances that may be detected before they impact on the process and those which can only be detected by their effect on the outputs of the process.

Just as the state variables have been chosen as a convenient set for the optimisation so to some extent are the control variables. For example, it may be simpler to formulate an optimisation algorithm using temperature as the control variable even though in practice it is a flow of heating utility that will be manipulated. Once again to help keep the formulation of the optimisation algorithm simple this is dealt with by introducing an extra functional block. This is referred to as the *control distribution* block. The revised structure is shown in figure 3.9.

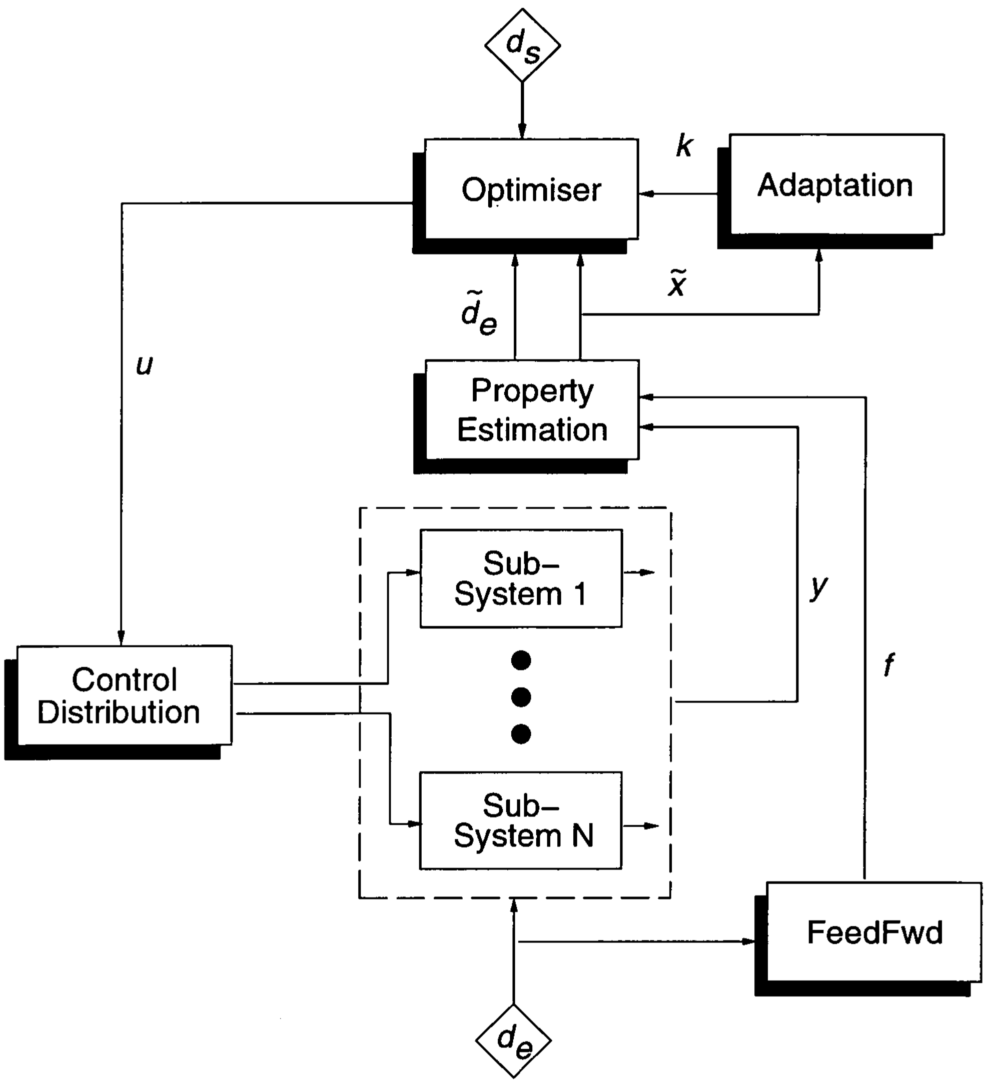


Figure 3.9: Complete Functional Structure of a Control Scheme

The control distribution block bridges the gap between the control variables of the optimisation and the available control mechanisms, it may take two forms. If the relation between the control variables and control mechanisms is simple then a straightforward mathematical transformation can be used. Alternatively if the relation is more complex it would be transformed into a supervisory demand for a slave operating task. In some cases implementing the control variable employed by the optimiser may require supervisory demands to be distributed to a set of slave operating tasks. For example, an optimisation algorithm may use production rate as a control variable which according to the abstract model employed fixes flowrates across the whole process. To bridge the difference between the abstraction and the reality of the underlying system the control distribution block converts the specification of production rate into supervisory demands for flowrates across the whole process.

The ability to preserve the abstract model used in the optimisation module is a convenient feature in the hierarchical development of the operating system. Initial control scheme designs focus primarily on the optimisation module using the abstract model available at that stage of design. When the process is refined the extra detail is accounted for in the slave operating tasks and the optimisation module does not have to be reworked.

Also shown in figure 3.9 is an *adaptation* block. Adaptation is used as part of the design strategy for addressing robustness problems. Where the design of a control scheme is too sensitive to variations from modelled behaviour there are three basic strategies for the redesign of that control scheme:

- Detuning: If it is not possible, or not desirable to track the model variation then it is necessary to 'detune' the optimisation to allow for the uncertainty that exists. Detuning usually implies a loss of performance.

- Mode Decomposition: If the variation in behaviour is large then it may be worthwhile dividing the range of behaviour into separate operating modes and tasks. A focussed strategy is then designed for each mode of operation. In doing this the original task is replaced by an executive task responsible for selecting the operating strategy best suited to the current process conditions.

- Adaptation: In many cases mode decomposition is an extreme solution to a robustness problem. The adaptation block therefore is introduced to adjust the strategic parameters of the optimisation and keep it in tune with the process state. Strategic parameters are values such as planning horizon or control gain.

The generic structure for a control scheme developed is common to all classes of operating task. The sequence in which this structure has been developed reflects the manner in which it is expected that a control scheme will be developed for an operating task.

3.8 A Preliminary Design Case Study

In this section the development of the operating system following preliminary design of the process is considered. The process that will be considered is an intentionally simplistic example. In chapter 4 a case study is presented for a real process (the manufacture of hydrofluoric acid).

3.8.1 Overview

During the preliminary design phase process designers are interested in generating a small set of candidate block flowsheets. This involves the enumeration of possible structural alternatives followed by shortcut economic evaluation to eliminate the least promising alternatives.

As candidate block flowsheets are identified they can be passed on to the operating system designers. For these flowsheets preliminary operating systems such as those identified earlier can be extended utilising the additional information now available.

Combining the knowledge of operating strategy with the techniques of operability analysis discussed earlier the operations designer can then provide a more complete evaluation of the feasibility of each candidate design. This can help further reduce the range of design alternatives to be considered to a manageable size.

At the same time through this development operational requirements for the process will be clarified (*ie.* the process characteristics desired for ease of operation). Being able to do this is important in establishing a forum for negotiation between the preferences of the process designers and those of the operating systems designers.

3.8.2 Outline of Steps in Operating System Development

The following is an indication of the basic steps followed in developing an operating system:

- Expand the Flowsheet: In particular make sure all inputs and outputs to the process have their sources or sinks identified. Also identify any key capacities that should be considered.

- Refine the Operating Task Specification: with the additional detail available about the process the optimisation specifications of the various operating tasks may be refined. This includes adding detail to the physical relations and also expanding on the demands and failure events that need to be addressed.

- Identify control mechanisms available to deal with each demand.

- Based on the interactions between control mechanisms determine what demands will be handled by the global control scheme and which will be distributed to subsidiary operating tasks.

- From the control scheme designs determine operability requirements for the process.

3.8.3 Process Outline

To help illustrate how the specific classes of operating task will be developed we will use a hypothetical process. The objective of the process is to produce a product X via a simple reaction path $A + B \rightarrow X + Y$. Y is a byproduct of negligible value which must be passed to a site waste treatment facility. Of the two raw materials B is the most expensive. It is therefore planned to run the reaction with an excess of A to ensure as complete conversion of B as possible. The initial process block diagram for this process is shown in figure 3.10

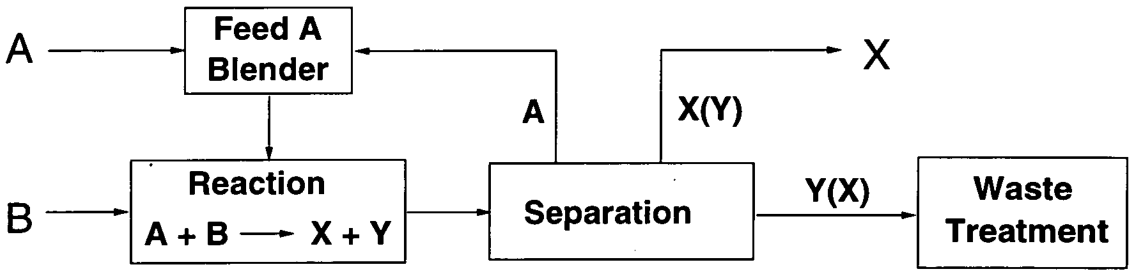


Figure 3.10: A Preliminary Block Flowsheet

For clarity when considering the operating system design some simple extensions to this flowsheet are introduced. First the basic storage facilities that are planned are added. The available storage facilities affects the stock control policy which is an important part of the top levels of an operating system design. Second to help with identifying sources of external demands all streams should have some form of both source and sink. To this end the raw material streams for A and B are drawn as coming from *market sources* and the product stream for X is shown going to a *market sink*. The byproduct Y is going to a waste treatment facility which is outside the scope of this design project and is not traced any further than that block. The extended flowsheet then looks like that of figure 3.11

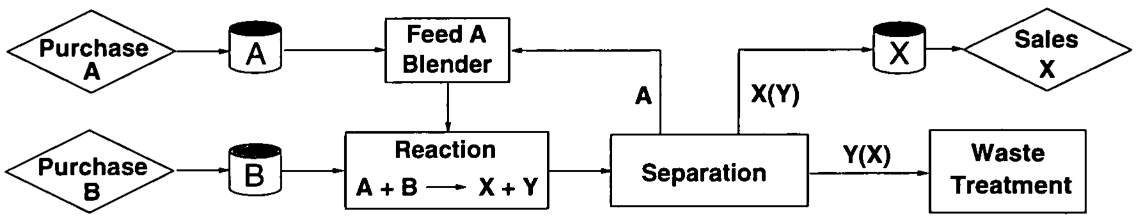


Figure 3.11: Enhanced Preliminary Block Flowsheet

3.8.4 Refinement of Regulatory Tasks

To illustrate the steps in development of a regulatory task the production task will be considered as an example. The first step is to derive a model for the operating task for which the categorisation developed in section 3.5 is followed.

3.8.5 Physical Relations

At this stage of design only general details are available. The emphasis therefore is on identifying the key variables that will be employed in operations planning at this level in the operating system hierarchy. If the process designers have made a basic evaluation of the process then the models they employed can probably be directly used here. The formulation of a model starts by relating the objective (maximising profit) to the state variables of the process.

$$\begin{aligned}\frac{d}{dt}.Profit &= Sales\ Rate\ X \times \\ &\quad (Added\ Value\ In\ X - Unit\ Operating\ Costs) \quad (3.2)\\ Added\ Value\ In\ X &= Market\ Value\ X - Market\ Cost\ A \\ &\quad -Market\ Cost\ B \quad (3.3)\\ Unit\ Operating\ Costs &= Blender\ Costs + Reactor\ Costs + Separation\ Costs \\ &\quad +Waste\ Treatment\ Charges \quad (3.4)\end{aligned}$$

For the individual section costs only qualitative relations are possible,

- Feed A Blender: The blending system at this stage of design is predicted to have negligible operating costs, *ie.*

$$Blender\ Costs = 0 \tag{3.5}$$

- Reactor: The two major factors affecting the reactors unit operating costs are its throughput and the amount of excess A fed to the reactor (full conversion of B is assumed). Nominally the reactor throughput is defined in terms of the feed rate of B. Therefore the reactor cost function has the form,

$$Reactor\ Costs = C_R(Throughput\ B, \%Excess\ A) \tag{3.6}$$

- Separation System: As with the reactor system the cost of the separation system can be expected to be dependent on its throughput. In addition increasing either the purity or recovery requirements for X will increase the operating cost of the separation system. Finally variations in separation feed composition can be expected to have significant effects on operating cost. The cost function for the separation system is therefore expected to take the form,

$$\begin{aligned} Separation\ Costs \ = \ & C_S(Separator\ Feed\ Rate, \%Recovery\ X, \\ & \%X\ in\ Product, \%X\ in\ Separator\ Feed, \\ & \%A\ in\ SeparatorFeed) \end{aligned} \tag{3.7}$$

- Waste Treatment: The waste treatment system charges for its service on the basis of volume treated. The unit operating cost for this system depends therefore on its throughput and composition.

$$Waste\ Treatment\ Charge = C_W(Waste\ Feed\ Rate, \%X\ in\ Waste) \tag{3.8}$$

To determine the interrelations between the variables that have been enumerated above and to derive the degrees of freedom available the fundamental physical laws must be taken into account. At this level of detail only mass conservation is of concern. Assuming complete conversion of B, and including the mass conservation relations the profit function can be reformulated as,

$$\begin{aligned}\frac{d}{dt}Profit &= Sales\ Rate\ X \times (Added\ Value\ In\ X \\ &\quad -C_R(Throughput\ B, \%Excess\ A) \\ &\quad -C_S(Throughput\ B, \%Excess\ A, \%X\ in\ Product, \%\ Recovery\ X) \\ &\quad -C_W(Throughput\ B, \%X\ in\ Product, \%\ Recovery\ X))\end{aligned}$$

The mass conservation also introduces other dependencies from the dynamic mass balance applied to the storage tanks, *viz*

$$\begin{aligned}\frac{d}{dt}(A\ Stored) &= Purchase\ RateA - Throughput\ B &(3.9)\\ \frac{d}{dt}(B\ Stored) &= Purchase\ RateB - Throughput\ B &(3.10)\\ \frac{d}{dt}(X\ Stored) &= Throughput\ B \times \frac{\%Recovery\ X}{1 - \%X\ in\ Product} \\ &\quad -Sales\ Rate\ X &(3.11)\end{aligned}$$

No further physical relations are needed at this stage of design.

3.8.6 Physical Limits

Most of the physical limits are straightforward (*eg.* fractions must be in the range of zero to one) and will not be enumerated here. An important set of limits that are worth mentioning are the market limits. For this example, a case will be considered where there is a limit to the amount of X that may be sold and to the amount of B that may be purchased. The resulting constraints are,

$$0 < Purchase\ Rate\ B < Availability\ B \qquad (3.12)$$

$$0 < Sales\ Rate\ X < Demand\ X \qquad (3.13)$$

3.8.7 Failure Events

In the preliminary design only one general failure event was identified of hazardous operation. With the additional detail about the process it is possible to refine this. First there are the potential overflow hazards for the storage tanks. For the production operating task this defines the following constraints,

$$\begin{aligned} A\ Stored &< Max\ A\ Storage \qquad (3.14)\\ B\ Stored &< Max\ B\ Storage \qquad (3.15)\\ X\ Stored &< Max\ X\ Storage \qquad (3.16) \end{aligned}$$

In addition there are the failure conditions of hazardous operation for each of the process sections. However since at this stage it is not possible to provide a better definition for these it is not possible to give them further attention in the production task.

For the production task there is an additional failure condition derived from product quality requirements which introduces a further constraint,

$$98 < \%X\ in\ Product \qquad (3.17)$$

3.8.8 External Demands

For the model that has been developed the principal external demands are associated with the market nodes and affect *Added Value in X*, *Availability B*, and *Demand X*. It is important to determine the magnitude and frequency of the external demands that affect these variables. With market related factors one form of demand that can be expected is seasonal variation. Seasonal variations operate over a long time scale and are typical of the external demands that would be addressed in the highest levels of the operating system hierarchy.

3.8.9 Operating Strategy

If the operating task truly had as simple a specification as the one developed here there would be little problem designing a single level control scheme to perform the task. However the operating task is expected to become more complex as the process develops. The aim at this stage of design is to identify a hierarchy of operating tasks suitable for distributing management of the production mode. The foundation for deriving this hierarchy is an analysis of the demands and how they may be controlled. For each demand there is a set of potential *control mechanisms*, where a control mechanism is defined as the set of free variables that will be controlled and the physical relations that connect them to the demand. For each control mechanism the following issues need to be assessed:

1. Control Power: Most control variables have a limited range of variation. At a minimum the intended control mechanism must have sufficient 'control power' to compensate for the expected range of the demand.

2. Speed of Response: it is important that the speed of response of a control mechanism can match reasonably the time characteristics of the demand. Speed of response will be limited in two ways
   - Speed of Action: There are limits on how fast a change in a variable can be implemented. Limits can be due to the physical inertia of a system, *ie.* to effect a change in temperature requires the thermal hold-up of the heat exchange equipment to be changed first. It can also be limited by operational requirements, for example while it may be physically possible to change the throughput of a process rapidly this would have destabilising effects on other parts of the operating system. Such a restriction most often applies to controls employed at the top levels of the hierarchy.
   - Speed of Detection: As well as limits on how fast a control action can operate there are limits on how fast the demand can be detected. Speed

of detection depends on where the demand can be detected and what means of measurement are available.

3. Scope of effect: this is defined as the set of process units whose operation will be significantly affected by a demand or the control response to that demand. The scope of effect is used to define the set of units that must be included in the process scope of an operating task.

4. Control Cost: The cost of a control mechanism should be compared against the cost of the best achievable control.

At the initial stages of design it will not be possible to perform a rigorous evaluation of all these factors. As with process flowsheet design to work round this it is necessary to employ design experience and where possible develop shortcut methods to evaluate the categories set out above.

Seasonal Variations of '*Added Value in X*': There is nothing that can be done within the scope of the production operating task to respond to any variations in this variable. The demand is of more relevance to the superior executive task where a change in operating mode may be considered if value of the product reduces sufficiently to make operation unprofitable.

Seasonal Variations in '*Availability B*': Seasonal variations affect the constraint on *Purchase Rate B* and through this the overflow constraint on *B Stored*. If operation is favoured by maximising the throughput of *B* leading to *Purchase Rate B* being matched to *Availability B*. In this case there are two control mechanisms available:

1. The inventory of *B Stored* can be allowed to vary so that *Throughput B* can be maintained. This is a relatively fast mechanism with only a small scope of effect. It is however limited in control action and for this reason

unsuitable as a mechanism to compensate for the large or long term upsets such as seasonal variations.

2. Adjust *Throughput B* to match the seasonal variations in *Availability B* and leave *B Stored* as a free control mechanism for demands that have not yet been detailed. Adjusting the *Throughput B* however has a significant impact on all parts of the process and so should be addressed as part of the global scope of the root production operating task.

Seasonal Variations in *Demand X*: These has a direct impact on the failure protection for the capacity constraint on *X Stored.* The demand is most relevant when operation is working to match *Sales Rate X* to *Demand X*. To avoid changing production throughput *XStored* could be allowed to float, but as before this is unlikely to provide sufficient control power to be able to isolate *Throughput B* from the seasonal variations in *Demand X*. Another possibility in this case is to use *Recovery X* to affect the feed rate to the storage tank for *X*. The control power possible from using *Recovery X* is even less likely to be sufficient and will certainly be a more costly mechanism. Therefore to manage this disturbance source again it seems best to use *Throughput B*.

For this simple example there is no problem using the same control mechanism to deal with two disturbance sources because at any one time only one of them is likely to be constraining operation. If this was not the case it would be necessary to ensure that there was sufficient flexibility in *Throughput B* to cope with simultaneous disturbances from both sources.

At a minimum the top level operating task must manage the seasonal variations in *Availability B* and *Demand* and will use *Throughput B* as its primary control variable. The other free variables are still to be allocated to operating tasks. While the demands that will be controlled by these free variables are not known their scope of effect can be approximated and an appropriate task hierarchy determined from that.

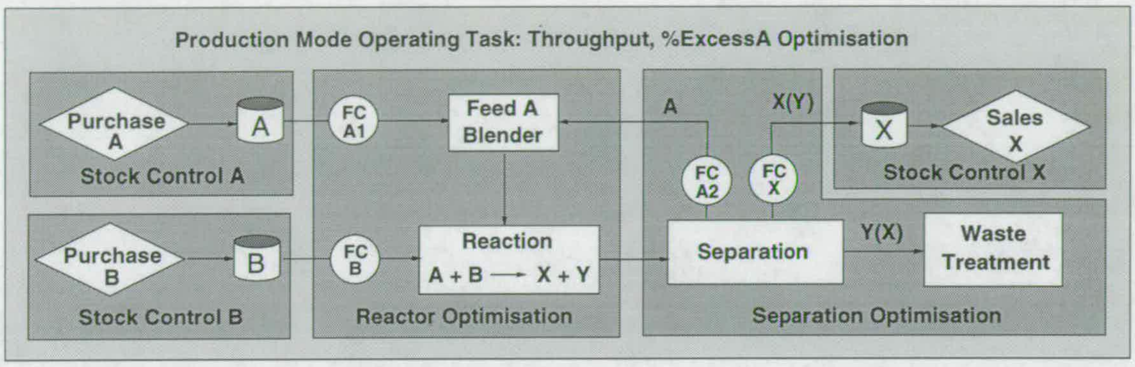


Figure 3.12: Block flowsheet with production task decomposition overlayed

Figure 3.12 overlays on the process flowsheet a possible decomposition for the production operating task. Note that where there is a possible conflict over which operating task controls a flow, a flow control block has been added and associated with a specific operating task. The elements of the hierarchy are:

- The Root Production Operating Task: The root production operating task has a process scope that encompasses all parts of the process flowsheet. It has already been identified that within this task the variable $Throughput\ B$ is to be used to coordinate process operation with market conditions. It is also decided that $\%ExcessA$ should be addressed at this level since it affects most elements of the profit function. If it is assumed that operating conditions favour maximising $Throughput\ B$ then the optimisation algorithm for this operating task will include a rule of the form,

$$Throughput\ B = \min(Availability\ B, Demand\ X) \tag{3.18}$$

- Stock Control Tasks: Each storage facility and corresponding market point have been assigned separate operating tasks. These will be responsible for adjusting the appropriate sales or purchase rate to control inventories.

- Blending and Reactor Optimisation Task: These two systems are grouped together because it is expected that fine control of $\%ExcessA$ will be implemented over the scope of these units. It might be expected to group the stock control of A under this task since any fine control of $\%ExcessA$ would affect the control of $AStored$. The presumption of the decomposition chosen is that the magnitude of any knock on effect to stock control will be sufficiently small that it can be managed as an extra disturbance source to the stock control task.

- Separation Optimisation Task: $\%RecoveryX$ and $\%XinProduct$ have been left free for this operating task to use in minimising its operating cost. Similarly to the blending and reactor optimisation task the scope of this operating task could have been extended to include the stock control of X. Once again the presumption is that any variations in operation in the separation system can be managed by the stock control task as a new disturbance source.

Control Multiplexing and Supervisory Demands: At this stage it is useful to consider how the control actions derived by the optimiser are applied to the process. It has been indicated that $Throughput\ B$ is determined at the top level of the task hierarchy with the intention of fixing flows across the whole flowsheet. This may appear to be eliminating important degrees of freedom at a very early stage in design. However control of $Throughput\ B$ is only being considered over a long time scale. What is determined is the desired long term average for throughput across the process. To express this within the framework of the operating task hierarchy *supervisory demands* are used. The supervisory demands define target flows appropriate to the process model used by the subsidiary task that correspond to the flow target derived by the optimiser. The interpretation of these supervisory demands can take two forms.

- A Directive: employed when an operating task actually has control over the variable specified by the supervisory demand. A directive can have varying degrees of sophistication. For example the directive can define upper and lower bounds on the property restricting the amount of free action left to the subsidiary task. Alternatively it may take the form of a nominal value and a cost penalty indicating how expensive a deviation from the nominal value will be. The subsidiary operating task can then include in its optimisation a measure of the cost of deviations from supervisory targets.

- A Notification: Some subsystems are included in the scope of an operating task not because they provide control but because they are affected by the control. For these subsystems the supervisory demand provides notification of the change in operating conditions providing advance information of a demand event.

For example in the hierarchy proposed above, the operating task responsible for the reactor also manages the flow controller FC$-B$. Based on the model employed by the optimiser this flow is directly determined by *Throughput B*. A supervisory directive is sent to this operating task to implement changes in throughput. The operating task managing stock control of B is also strongly affected by changes in throughput and receives a supervisory notification when a change in flow is requested. The use of the control distribution block to manage supervisory demands in this way is a useful mechanism for coordination of the hierarchy of operating tasks.

3.8.10 Operability Analysis

Having refined the design of the operating task hierarchy and associated control schemes the operating system designer is in a position to provide some useful feedback for the process designers. The understanding gained from developing the operating system can help focus operability analysis and provide the process designer with a better specification of the capabilities required for operation. For the simple example considered limited analysis is possible. A useful analysis that is possible is to use information about the market conditions to determine the flexibility in throughput rate required. The case study in chapter 4 gives more consideration to this.

Simply having an understanding of the operating strategy can help as it provides a stronger basis for negotiation between the preferences of the process designers and operating system designers. By having some idea of how the process will be operated it is easier to determine the advantages and disadvantages of opting for one process or operating system design over another. Creating this forum for negotiation is an important part of achieving integrated concurrent design.

3.9 Developing Transition Tasks

For small transitions the procedure for development of the transition task design follows the same steps as described above for regulatory tasks. That is first to use knowledge of the process design to refine the description of the operating mode, construct an operating strategy appropriate to this model and define the subsidiary operating tasks. For large transitions such as startup or shutdown the procedure is not as simply defined. Such transitions often involve several possible phases of behaviour and it is not practical to try and build models for every potential operating mode in a transition when only a few are likely to be relevant. Developing the definition of the operating mode and developing the transition task design are thus much more closely interlinked. This is similar to the situation faced at the initial stage of design where developing the mode decomposition and the operating task decomposition are closely interlinked. Procedures such as those developed by Fusillo and Powers [43] provide a useful framework for developing transition tasks:

1. Plant Decomposition:
   (a) Divide Process into subsystems that can be physically isolated from each other.
   (b) Specify the local initial and goal states.
2. Goal Reduction & Sequencing for each isolated system
   (a) Identify possible actions that will reduce the difference between the current state and the goal state.
   (b) Select a feasible sequence of actions using the constraints as a guide to reduce the search space.
3. Re-integrate the subsystems:
   Find a suitable procedure to recombine the subsystems and drive the whole process to its final goal state.

Determining the best process decomposition for an operation is a difficult balance. Smaller sets will be easier to drive to the goal states. However, achieving such isolation can be costly in terms of capital equipment and re-integrating the system can be more complex.

Based on such a procedure a transition task such as startup would develop into a decomposition such as that in figure 3.13. Each intermediate stationary state derived by the planning algorithm represents a new regulatory mode and correspondingly a new regulatory task. To connect the sequences of stationary states transition modes are included, each defining an additional transition task.

In addition to the operating tasks defined directly from the mode decomposition we have new executive tasks. First the original startup transition task is now redefined as an executive task. Its function is to coordinate the isolation and re-integration of the process sections. Second, for each section an executive task is assigned to coordinate their respective start up sequences.

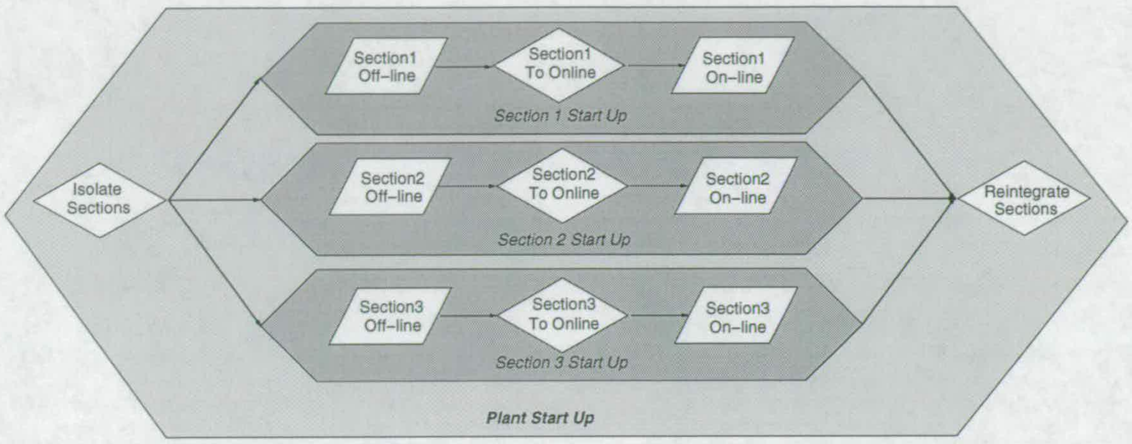


Figure 3.13: Decomposition of Transition Mode

As understanding of the operating sequence develops this structure may be simplified. For example some of the intermediate regulatory tasks may be dropped with the intention of moving from one transition mode to another. In general however the regulatory modes immediately following section isolation and preceding section re-integration will always be required while the separate startup sequences synchronise.

3.10 Development of the Executive Tasks

Regulatory tasks and transition tasks are intended to deal with single operating modes. In contrast the executive tasks are responsible for overseeing the switch over from one mode to another. One of the most important aspects of this is initiating failure recovery. For example, in the design of the regulatory task while attention was paid to the conditions that could lead to failure events it was not part of the design of the regulatory task to determine what action should be taken if such events took place. Managing alarms and determining the appropriate response to failure events is a function for the executive tasks. From the perspective of the generic task structure developed earlier the objective of the optimisation is focussed on determining the best mode of operation to be in. To design the executive tasks there are two basic pieces of information that are required about the supervised operating tasks.

- **Preconditions:** The preconditions for an operating task define the necessary state conditions for the process before that operating task can be brought into action.

- **Termination Conditions:** The termination conditions define states beyond which the operating task is not designed to cope. These will in the main correspond to the failure events of the corresponding operating mode. With transition tasks it would also include a definition of the end point that the transition is designed to reach.

A simple executive task design would orient around monitoring for termination conditions of the active task. If any arose it would search for a match between the process conditions and operating task preconditions to determine the appropriate operating task to initiate. More sophisticated designs might take a set of operating tasks that could be active and continuously review which would give the best performance.

3.11 Knowledge Representation for Operating System Design

A knowledge representation to support operating system design has been developed. The representation was developed in parallel with the framework that has been presented in this chapter and in part guided the development of the framework. A full discussion of the knowledge representation is provided in appendix C.

The principal problem that is considered is how a knowledge based system can assist in coordinating the development of the process and operating system. Two approaches are consider to address design coordination. The first approach was to represent the process and operating system as a combined design class, called a *production system.* A single design hierarchy is supported with refinement and decomposition applied to the process and operating system as a whole. Using a combined hierarchy was simple to represent but in the context of concurrent design was too restrictive.

The second approach to design coordination used independent design hierarchies for the process and operating system. Separate design hierarchies removed restrictions on the concurrent development of process and operating system designs. However with independent hierarchies more careful attention to the link between hierarchies is required. The approach that was adopted used a representation of operating tasks and their associated process model as the basis for providing the link. The operating tasks defines the requirements for operation and include a model of the system to be controlled. If the behaviour of a process is reasonably described by the operating task's system model then any operating system alternatives for that task can be used with the process.

The weakness of the second approach to design coordination is the complex system of object relations that were required. Where further development was seen as particularly necessary was in supporting the relationships between models, process designs, and operating tasks.

3.12 Summary

In this chapter the general principles of hierarchical design have been presented. These principles have been used in developing a framework for hierarchical operating system design. Using a hierarchical approach is convenient for integrating the concurrent design of the process and operating system. The framework also provides a basis for developing an integrated operations management policy.

Designing an operating system is approached by defining a hierarchy of operating tasks and then developing a control scheme for each task. The hierarchy employs both process based and behaviour based decomposition to balance performance with model complexity. A generic control scheme structure is proposed for operating tasks which emphasises the use of explicit models in optimisers. Property estimation and control distribution modules are used to interface between the model employed for optimisation and the reality of the underlying system.

The process example that has been used in this chapter is a simple and artificial problem. In chapter 4 a more complete case study is presented which addresses the design of an operating system for a hydrofluoric acid plant.

Chapter 4

Operating System Design for a Hydrofluoric Acid Plant

In chapter 3 a small example of process operating system design was presented for a simple process, a more complete case study will now be developed. The subject of the case study is the design of a process and operating system for the production of hydrofluoric acid. Background on the process is available in the literature [60, 61] and from the BUSS patent [62]. The process will be developed through the hierarchical steps proposed by Douglas [57]. At each stage of development consideration is given to the appropriate operating system design activities. It should be noted that while only one decomposition is developed in the case study more choices exist in the design of the operating system. Appendix B contains the derivations of the models used in the case study.

4.1 Input-Output Analysis

The process input/output structure is shown in figure 4.1. The function of the plant is to produce two grades of hydrofluoric acid, a technical grade (HF_T, $>$ 90% HF), and an anhydrous grade (HF_A, $>$ 99.5% HF). The hydrofluoric acid is produced by the reaction of CaF_2 with concentrated sulphuric acid,

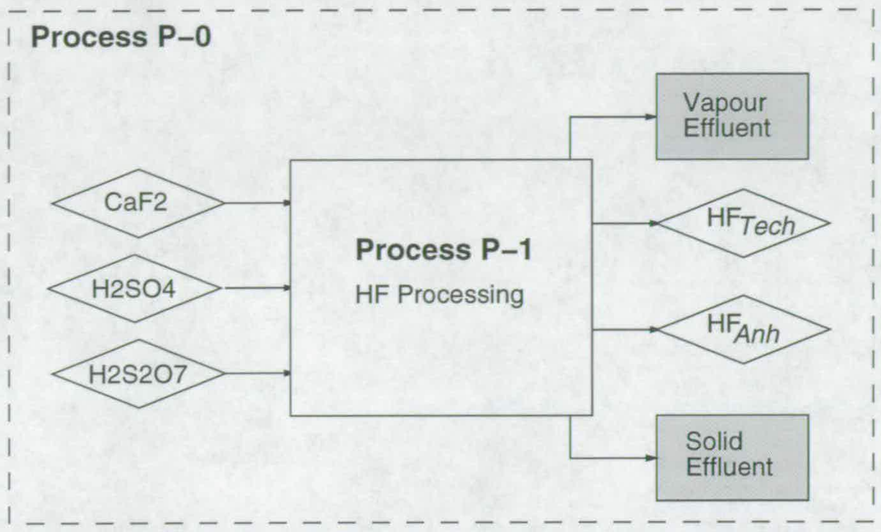


Figure 4.1: Process input/output structure.

$$CaF_2 + H_2SO_4 \rightarrow 2\ HF + CaSO_4 \qquad (R1)$$

The feed sulphuric acid is provided as a combination of concentrated sulphuric acid and oleum (which is used to eliminate any excess water that may be present). The source of the $CaF_2$is fluorspar ore. The primary impurities in the fluorspar are water, $CaCO_3$ and SiO_2. The water increases the amount of oleum required while the $CaCO_3$ and SiO_2 lead to the following side reactions:

$$\begin{aligned} CaCO_3 + H_2SO_4 &\rightarrow CaSO_4 + CO_2 \qquad (R2) \\ SiO_2 + 4\ HF &\rightarrow SiF_4 + 2\ H_2O \qquad (R3) \end{aligned}$$

Two effluent streams are produced a solid effluent ($CaSO_4$) and a vapour effluent (SiF_4, CO_2, H_2O). The effluent treatment blocks will be treated as utility systems and are grayed out in figure 4.1.

4.1.1 Production Task Modelling

It is assumed that the operating system design has already been decomposed into offline, startup, production, shutdown and primary executive tasks as was illustrated in figure 3.5. Development of the production task will be considered first.

The initial step is to build a model of the process that identifies the key demands and failure conditions relevant to the production task. At the input/output design level modelling focusses on the the market nodes which are divided into two classes: purchase nodes and sales nodes.

Purchase Nodes: are the source of feed materials. The key elements for modelling a purchase node are

- **Quality:** variations in feed compositions are modelled as external demands.
- **Availability:** A limit on supply rate of a feed is modelled as a physical constraint. External demands are used to model any variability in the limit.
- **Dynamics:** how quickly can supply be adjusted to match consumption.

Models for external demands focus on putting bounds on the magnitude of variation and if possible providing some indication of the time scale of the dynamics of the demand.

For the purposes of the case study the sulphuric acid and oleum are treated as reliable and unlimited resources. For the fluorspar the following conditions are assumed:

- Quality: As a natural ore the composition of fluorspar cannot be expected to be constant. The uncertainty in fluorspar composition is modelled by an external demand: [1]

  ED-1: { description: "Variation in CaF_2 content of fluorspar ore"
  disturbance variable: "Fraction of CaF_2 in feed fluorspar"
  range: 0.95-0.99
  dynamics: continuous, frequency 10 minutes per cycle. }

- Feed Availability: there is no limit to the availability of ore.

[1]The format adopted for describing demands and failure conditions is for clarity and does not represent active objects.

- Dynamics: The delivery of ore is considered to be continuous but changes in supply rate must be placed seven days in advance. The requirement is modelled as a simple time delay:

$$Supply\_Fluorspar(t) \quad = \quad Purchase\_Fluorspar(t - 7days)$$

Sales Nodes: these are the final destinations for products. The key elements required for modelling a sales node are:

- **Quality:** what are the product composition requirements? These will be modelled as failure conditions.
- **Rate of sale:** what variation in sales rate is expected? Any variability is modelled by an external demand.

For the technical grade hydrofluoric acid,

- Quality: technical grade acid requires a minimum of 95% HF. The quality constraint is modelled by a failure condition *viz*

  FC-1: { description: "Off spec technical grade product"
  limit variable: "Composition of HF in technical grade product"
  failure condition: < 95% }

- Rate of sale: Technical grade product has a reasonably steady market, changes are only expected to occur on a time scale of a month. The variability is sales rate is modelled by the demand:

  ED-2: { description: "Market demand variations for technical grade"
  disturbance variable: "Sales rate of Technical Grade"
  range: 45,000-50,000kg/day
  dynamics: sustained deviations for periods of 30 days }

For the anhydrous grade,

- Quality: anhydrous grade requires a minimum of 99.5% HF. The quality constraint is again modelled as a failure condition:

  FC-2: { description: "Off spec anhydrous grade product" limit variable: "Composition of HF in anhydrous grade"

failure condition: $< 99.5\%$ }

- Rate of sale: The market for anhydrous grade acid is smaller and more competitive. Wider variations in sales rate are expected on a more frequent time scale than encountered for the technical grade. The external demand that models the variability in anhydrous grade sales is:

  ED-3: { description: "Market demand variations for technical grade"

  disturbance variable: "Sales rate of Technical Grade"

  range: 5,000-15,000 kg/day

  dynamics: sustained deviations for periods of 4 days

Hazard Analysis: further failure conditions are identified from analysis of the hazardous properties of the materials being used in the process. For example, emission of HF to the atmosphere is a general hazard that is described by the failure condition, *ie.*

FC-3: { description: "Hazardous loss of HF to atmosphere" }

The failure condition is only minimally defined. More specific failure conditions implied by this general failure condition are identified when failure management is addressed. .

4.1.2 Failure Management

In constructing a model for the production task several failure conditions have been identified. The development of the operating system design starts by addressing these failure conditions. There are two design issues to be considered for each failure condition:

- Failure Prevention: what actions can be taken to prevent failure?
- Failure Recovery: what action should be taken if the failure event should occur?

Failure prevention: to ensure the product quality constraints (failure conditions *FC-1* and*FC-2*) are met requires composition control on the appropriate streams. At the current level of process abstraction it is not possible to determine appropriate mechanisms for composition control. Instead the control requirements are marked as incomplete loops on the process block diagram (see figure 4.2).

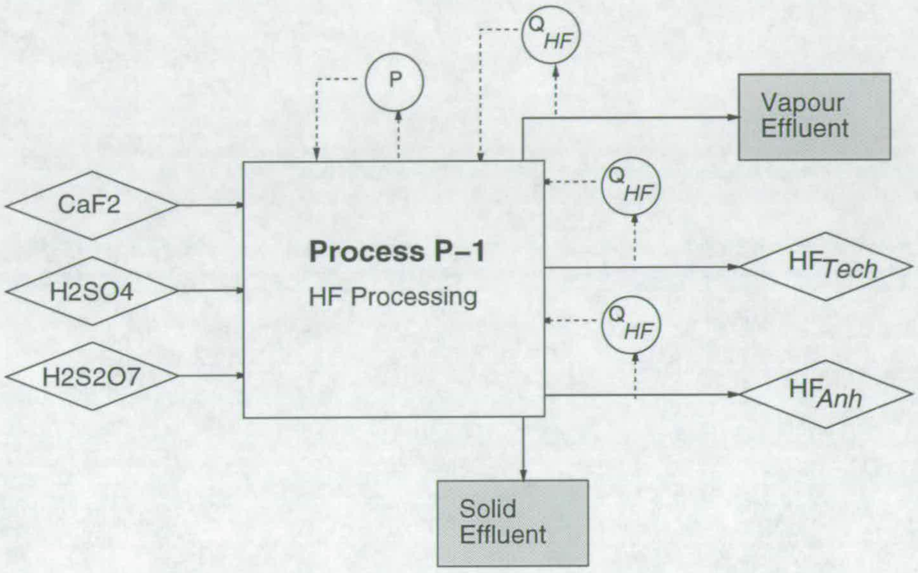


Figure 4.2: Process with failure prevention loops

To consider failure prevention for the safety constraint *FC-3* it is first necessary to consider how failure might occur. At the current level of design two mechanisms can be foreseen by which HF could escape to the atmosphere *ie.* either with the vapour effluent or by direct leakage from process equipment. In the first case the vapour effluent treatment system provides a basic preventive mechanism. However, the capability of the vapour treatment system to remove HF is restricted, therefore to protect failure condition *FC-3* a secondary failure condition is derived,

FC-4: { description: "HF composition limit on effluent"
limit variable: "Composition of HF in vapour effluent"
failure condition: $> 1\%$ }

Prevention for the new failure condition is provided by applying composition control to the effluent feed stream.

In the case of a leak in the process equipment the safest approach is to ensure that the pressure in all units is below atmospheric pressure. Any leaks will then be of air into the process rather than of HF to the atmosphere. A secondary failure event again defines this requirement,

FC-5:{description: "Preventive Condition for FC-4"
limit variable: "Pressure of units in section $P1$"
failure condition: $< P_{Atm}$" }

The additional pressure and composition control requirements are also marked on figure 4.2.

Failure recovery: a strategy for failure recovery is necessary when the prevention strategy cannot guarantee that the failure will not occur. Within the initial operating task structure the basic action available for failure recovery is process shutdown. For hazardous conditions that cannot be immediately controlled shutdown is the necessary action and therefore is required for failure condition *FC-3* and the conditions derived from it (*ie. FC-4* and *FC-5*). Basic alarm response for these failure conditions is assigned to the primary executive task which is responsible for initiating shutdown. For the quality constraints *FC-1* and *FC-2* shutdown is only justified if the quality requirements are persistently off spec. In the primary executive the alarm response to a quality constraint failure is therefore delayed to allow subsidiary tasks to make less extreme corrective action.

4.1.3 Demand Management - Stock Control

Three external demands have been defined for the production task. The first demand (*ED-1*) affects the composition of the fluorspar. At the input/output level of design this demand cannot be appropriately addressed and is deferred.

Demands *ED-2* and *ED-3* affect the product sales rate and are an important consideration in developing a stock control policy. The approach taken to stock control also impacts on the operability requirements for the process. Three stock control strategies will be considered.

Stock Control Strategy 1: The simplest approach for process design is to set a fixed production rate for both product grades. Such an approach requires that a floating inventory of product be maintained to absorb changes in the sales rate for each product. Assuming the simple operating profile shown in figure 4.3

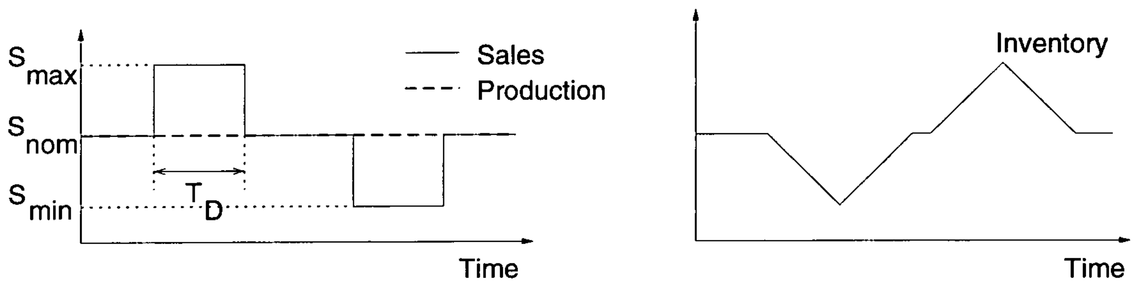


Figure 4.3: Stock control strategy 1

a conservative estimate of the necessary storage capacity of each grade can be derived (see section B.1).

$$V \geq T_D(S_{max} - S_{min}) \tag{4.1}$$

$$\begin{aligned} \text{where } V &\equiv \textit{Storage capacity} \\ T_D &\equiv \textit{Time for which a peak deviation persists} \\ S_{max} &\equiv \textit{Maximum sales rate} \\ S_{min} &\equiv \textit{Minimum sales rate} \end{aligned}$$

A constant production does not require any special buffering for the process feeds as they will be consumed at a constant rate. For disturbances with a long time scale or wide sales ranges the constraint implies a large storage requirement for the product. For the variations in sales rate set out in demands *ED-2* and *ED-3* the storage requirement for technical and anhydrous grades would be 150,000kg and 40,000kg respectively. Storage capacities of that order are both economically unviable and a serious safety hazard.

Stock Control Strategy 2: An alternative control strategy is to adjust production rate to match current sales demand. Some storage capacity will still be required to meet the sales requirements during the time it takes to change the production rate of the process. Again, assuming a simple operating profile (figure 4.4),

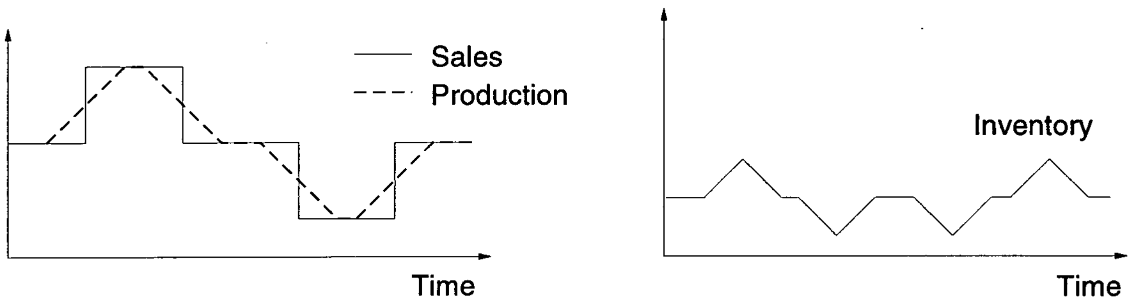


Figure 4.4: Stock control strategy 2

an estimate of the minimum storage required can be derived (see section B.2),

$$V \geq \frac{(S_{max} - S_{min})^2}{4r} \tag{4.2}$$

$$\text{where } r = \textit{Speed of response of production rate}$$

The derivation assumes that changes in sales rate can be predicted in advance by half the time it takes to respond (*eg.* sales orders precede sales delivery by $(S_{max} - S_{min})/4r$). If sufficient advance notice is not available storage requirements will increase.

The second strategy requires more flexibility in the process design. Comparing the limits for the two strategies a necessary condition to justify the extra flexibility is

$$r > \frac{(S_{max} - S_{min})}{4T_D} \tag{4.3}$$

To satisfy the variation in technical grade sales given in *ED-2* would require $r > 42kg/day^2$. Similarly for demand *ED-3* requires that $r > 625kg/day^2$. A response rate of $625kg/day^2$ is easily achievable but the storage capacities required will be 10,000kg and 40,000kg of technical grade and anhydrous grade respectively. To get the storage capacities for anhydrous grade down to the equivalent of 1 hours production requires that the process be able to change from nominal to maximum production rate in 3 hours (*ie.* $r > 40000kg/day^2$).

Stock Control Strategy 3: A refinement of the second strategy is to utilise the storage capacity for technical grade HF as a common buffer for both products. Using a common storage buffer is possible when the technical grade is produced as an intermediate step in the production of the anhydrous product. The process can be split into two sections with the technical grade storage interposed between them (see figure 4.5). The rectifier section ($P3$) upgrades the HF product from

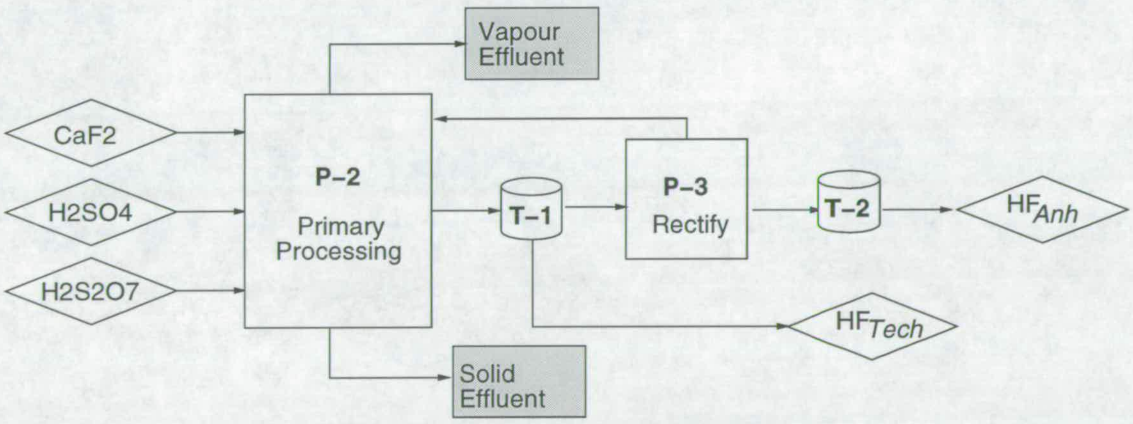


Figure 4.5: Two part process decomposition

technical standard to anhydrous standard and is expected to have a faster response than the primary processing section ($P2$). The storage requirement for anhydrous HF, V_{T2}, is determined by the response rate of the rectifier section alone, r_{P3},

$$V_{T2} \geq \frac{(S_{max,An} - S_{min,An})^2}{4r_{P3}} \tag{4.4}$$

If the response rate of the rectifier section is sufficiently high it may be possible to eliminate the requirement for anhydrous storage. To compensate for the slower response rate of the first section additional technical grade storage capacity is required. The worst case scenario is when the peaks in demand for both products coincide. The storage requirement for technical grade is then,

$$V_{T1} \geq \frac{(S_{max,T} - S_{min,T})^2}{4r_{P2}} + \frac{(S_{max,An} - S_{min,An})^2}{4}\left(\frac{1}{r_{P2}} - \frac{1}{r_{P3}}\right) \quad (4.5)$$

When $r_{P2} = r_{P3}$ this is equivalent to the previous strategy. The total storage $(V_{T1} + V_{T2})$ is determined by r_{P2} which is also the determining factor in the overall response rate of the process (r). The total storage therefore is similar to that for the second strategy, r_{P3} only determines the distribution of storage between V_{T1} and V_{T2}. Therefore for the worst case scenario the gains from the new strategy are restricted to the difference in storing anhydrous versus technical grade HF.

The total storage requirement can be reduced if the peaks in sales of technical and anhydrous grade are not expected to coincide frequently. If peaks are never coincident then anhydrous storage can be reduced to the maximum of the following two constraints,

$$V_{T1} \geq (S_{max,T} - S_{min,T})^2/8r_{P2} \quad (4.6)$$

$$V_{T1} \geq (S_{max,An} - S_{min,An})^2/8(r_{P2} - r_{P3}) \quad (4.7)$$

Three alternative stock control strategies have been considered. From an analysis of the storage and process requirements the third strategy is chosen. Further analysis of the failure conditions and demands at the current stage of design is not practical.

4.1.4 Operating task decomposition

The division of process operations in the third stock control strategy provides a basis for decomposing the production task into a hierarchy as shown in figure 4.6.

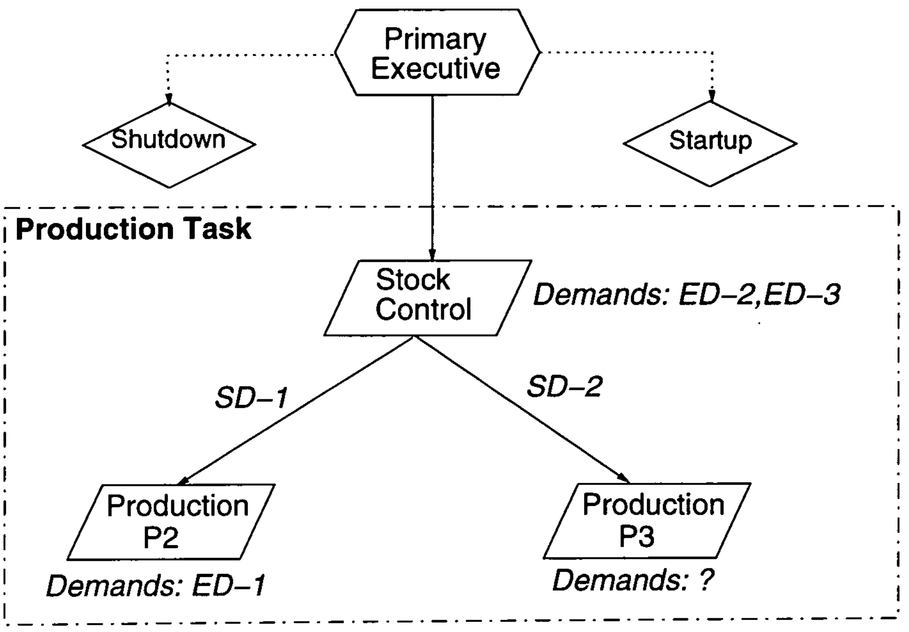


Figure 4.6: Production task decomposition

At the top of the hierarchy is the stock control task which monitors stock levels and market demands and determines the target production rates. Control of the process below this task is decomposed into two parts corresponding to the two sections of the process. The stock control task directs the operation of the subsidiary tasks via supervisory demands. The analysis of the control strategy provides additional information on the expected range and and dynamics of the supervisory demands.

SD-1:{description: "Target production rate for section $P2$"
range: $50,000 - 65,000 kg/day$"
dynamics: discrete, $d/dt > 40,000 kg/day^2$ }

SD-2:{description: "Target production rate for section $P3$"
range: $10,000 - 15,000 kg/day$"
dynamics: discrete, $d/dt > 60,000 kg/day^2$ }

The external demand affecting the fluorspar feed quality (*ED-1*) is delegated to the operating task controlling the primary processing section.

4.1.5 Startup and Shutdown

Some failure conditions have been identified for which shutdown should be initiated. Monitoring the failure conditions and instigating shutdown is the responsibility of the plant executive. The operating system design that can be performed at the current level for the shutdown and startup tasks is very restricted. The division of the process into two sections and the presence of a capacity between them does provide a convenient basis for decomposing startup and shutdown procedures. Also it is noted that section $P3$ is expected to have faster dynamics than section $P2$. It is undesirable to require section $P3$ to be maintained in an idle state while waiting for section $P2$ to start up. A start up procedure is therefore chosen that delays start up of section $P3$ until the latest practical moment. The decomposition of the startup operating task is shown in figure 4.7. In choosing

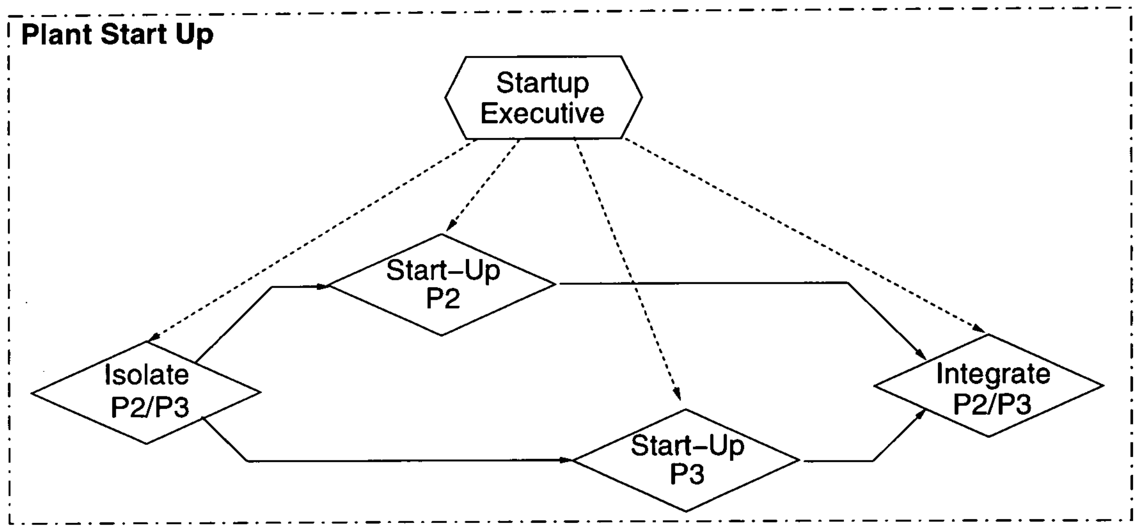


Figure 4.7: Decomposition of Plant Startup

this strategy it is being assumed that heat integration will not couple the rectifier and primary processing section.

4.2 Recycle Analysis

In the Douglas methodology after defining the input/output structure the next step is to decide on the required reactor systems and associated recycles. Separation systems are treated as black boxes at this level. For the HF process the development of a recycle structure is straightforward. The products are produced in a single reaction step and no reactions are reversible. Of the reactants only the H_2SO_4 is feasible to recover and recycle. The recycle structure for the process is shown in figure 4.8.

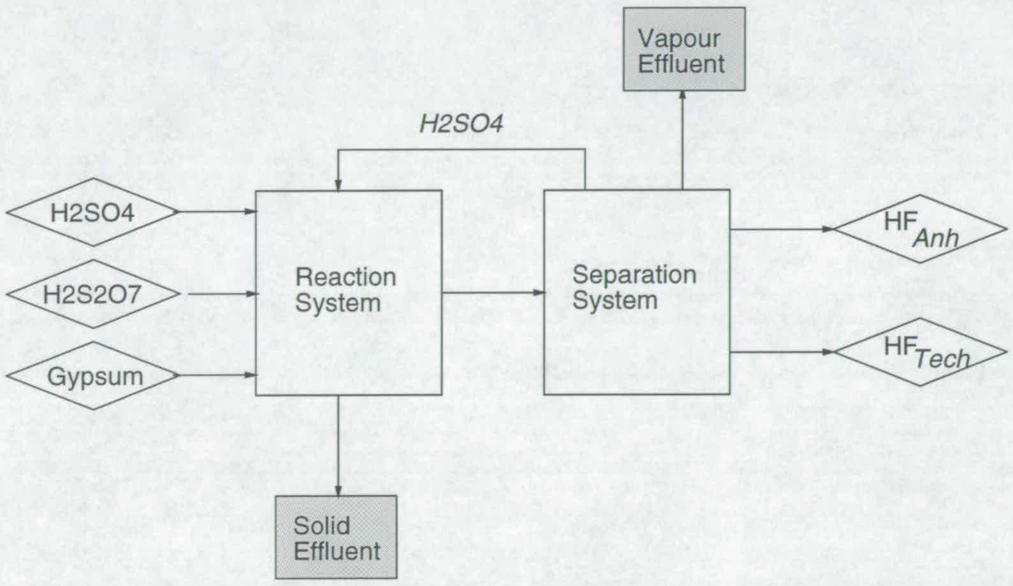


Figure 4.8: Recycle Structure of HF Process

The process decomposition that has been developed for the production task hierarchy (figure 4.5) does not match the recycle structure shown in figure 4.8. As the decompositions are not incompatible it is a simple step to combine the two (figure 4.9). The process design focusses on the reactor and acid recycle systems which are encompassed by the primary processing section ($P2$). The results of the

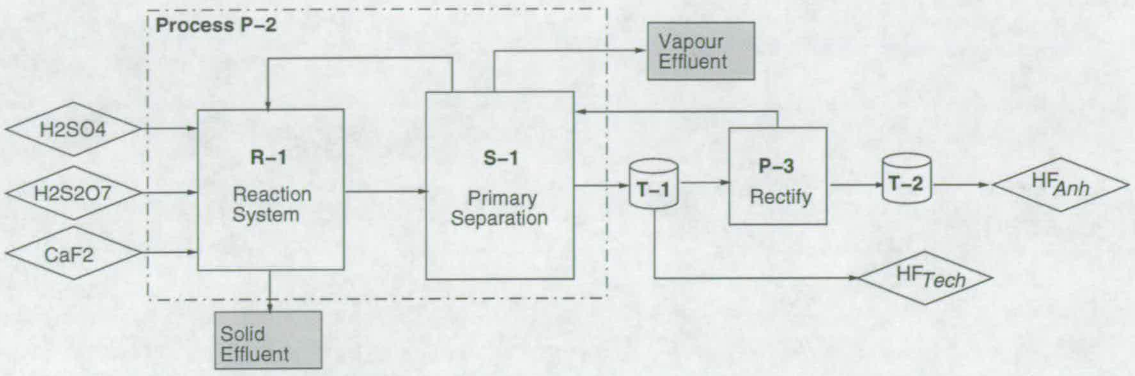


Figure 4.9: Merging of process decompositions

process design are therefore most relevant to the operating task controlling that section.

4.2.1 Preliminary Process Optimisation

The models used for preliminary process optimisation are an important basis for the refinement of the production task hierarchy. A short analysis of the process optimisation is therefore provided here. The key design variables introduced in developing the recycle structure are:

- Reactor size, V_R.
- Reaction temperature T_R.
- Reactor Pressure P_R.
- Molar ratio of H_2SO_4:CaF_2 at reactor inlet, A.
- Fractional recovery of H_2SO_4in the separation section, R_{SO4}.

For optimisation of the recycle structure it is assumed that production rate has been fixed (*eg.* as a result of the stock control strategy). A simple profit function can be derived,

$$Profit = 0.5F_{HF}.\left(C_{HF} - \frac{C_{CaF2}}{X} - \frac{A(1-R_{SO4})}{X+R_{SO4}}.C_{SO4}\right) \quad (4.8)$$

$$-(C_{R,F} + C_{R,O}) - C_S \quad (4.9)$$

where F_{HF} $\equiv$ Annual production rate of HF

$$\begin{aligned}
X &\equiv \text{Fractional conversion of CaF}_2\text{to HF} \\
C_{CaF2} &\equiv \text{Purchase price for CaF}_2 \\
C_{SO4} &\equiv \text{Purchase price for H}_2\text{SO}_4 \\
C_{R,F} &\equiv \text{Annualised fixed costs for the reactor} \\
C_{R,O} &\equiv \text{Annual operating costs for the reactor} \\
C_S &\equiv \text{Annualised separator costs} \\
C_{HF} &\equiv \text{Combined sales price for } HF_T \ \&\ HF_A \\
&\quad\ \text{(a fixed split between the two products is assumed)}
\end{aligned} \tag{4.10}$$

To estimate the reactor fixed costs ($C_{R,F}$) a correlation based on reactor size V is required. The operating costs of the reactor ($C_{R,O}$) are derived from the cost of the utilities that are required to meet the reactor's heat duty. The reactor heat duty will be determined by the reactor conversion and temperature. The separation costs (C_S) are more difficult to correlate at the current level of abstraction. If C_S cannot be correlated with the design variables it is difficult to determine the optimal level of recovery for H_2SO_4. The derivative of the profit function with respect to recovery, R_{SO4}, is,

$$\frac{d(Profit)}{dR_{SO4}} = 0.5F_{HF}(\frac{r}{X} - 1)C_{SO4} - \frac{d(C_S)}{dR_{SO4}} \tag{4.11}$$

Ignoring the contribution from C_S would lead to an optimal recovery of 100% which is not realistic. Instead the recovery is set to 98% which is a heuristic estimate of the optimal recovery. For the remaining design variables the influence of C_S is assumed to be zero.

The derivative of the profit function with respect to reactor temperature is given by,

$$\frac{d(Profit)}{dT_R} = \frac{F_{CaF2}}{2X^2}.(C_{CaF2} + A(1 - R_{SO4})C_{SO_4})\frac{dX}{dT_R} - \frac{d(C_{R,O})}{dT_R} \tag{4.12}$$

In determining the optimal reactor temperature there is a trade off between material costs and utility costs. For the purpose of the case study it is assumed that material costs dominate over utility costs. The resulting implication is that profit increases monotonically with reaction temperature. The reaction temperature is ultimately constrained by the limitations of the construction materials which is modelled by the failure condition:

FC-6:{description: "Maximum safe temperature for reactor"
variable: "Reactor wall temperature"
limit: $< T_{max,Reactor}$" }

The optimal temperature for the reactor is at this constraint.

The reactor pressure (P_R) only has a weak effect on the reaction kinetics. However because operation must run below atmospheric pressure (see failure condition *FC-5*) there is a problem of drawing air into the process. To keep the inflow of air to a minimum reactor pressure is kept as close to the constraint of *FC-5* as possible.

The remaining design variables are acid to fluorspar ratio (A) and the reactor size (V_R). As neither of these are constrained and they are interdependent their values are optimised simultaneously.

4.2.2 On-line Optimisation of Section P2

The break down of the process optimisation that has been derived above for design of the process is also a useful basis for the online optimisation of the process. As the reactor volume will be fixed the number of decision variables is reduced. However in the online optimisation it should be possible to take more accurate account of the cost of separation. Recovery is therefore included in the simultaneous optimisation rather that being treated as a fixed variable. The optimisation can be broken down into three regulatory tasks:

- Constraint control on the reactor temperature.

- Constraint control on the reactor pressure.
- Simultaneous optimisation of recovery and acid:fluorspar ratio.

Determining how the tasks will be organised into a hierarchy is dependent on their scope of effect. Optimising the recovery and acid:fluorspar ratio must take into account both the reactor section and primary separation section and therefore is expected to be placed at a high level in the hierarchy. In contrast the temperature control can be reasonably treated as a task local to the reactor system. The pressure control is less easily placed into the hierarchical structure.

The pressure of the reactor is controlled by regulating the flowrate of the reactor offgas. The pressure driving force for all flows of gas and vapour in the process is provided by an exhaust fan at the end of the process. The flowrate of the reactor offgas may be regulated either by regulating the power to the exhaust fan or by placing a flow control valve on the reactor offgas stream. The latter option would give the most direct control of reactor pressure and keep the scope of the pressure control local to the reactor system. However a valve on any of the gas or vapour streams is undesirable as it will increase pressure drops and therefore pumping costs. The chosen method therefore is to control pressure by regulating power to the exhaust fan.

To control pressure by regulating exhaust fan power implies a scope for the pressure control that is wider than that of the optimisation (it affects both the primary processing section and the rectifying section). On this basis pressure control should be placed above the optimisation. However pressure control is expected to operate on a much faster time scale than the optimiser. Also while pressure control only has a weak influence on the optimisation, the optimisation is expected to have a more significant influence on the pressure control. The flow of information therefore is mainly from the optimiser to the pressure control, providing advance notice of shifts in operating conditions. Pressure control is therefore placed as a subsidiary task to the optimiser.

The final task hierarchy for production optimisation of section $P2$ is shown in figure 4.10. Solid lines are directives while dashed lines identify feedback or

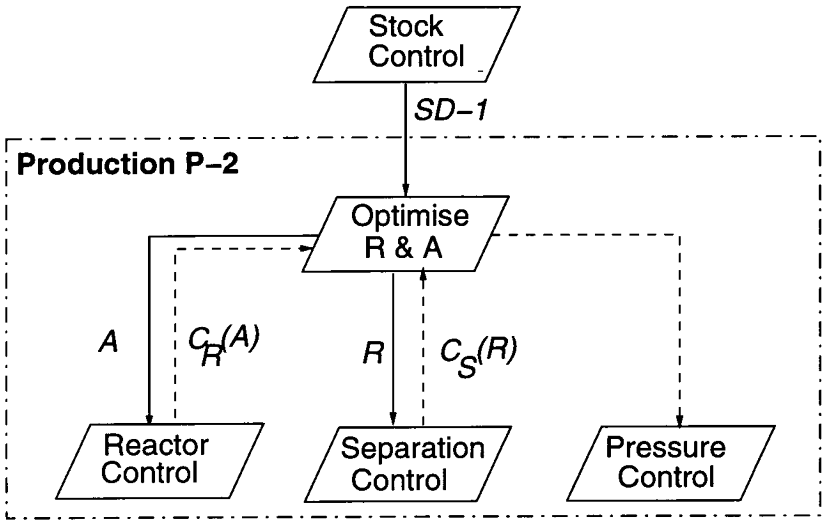


Figure 4.10: Operating task decomposition for production optimisation

feedforward information. The tasks in the decomposition are:

- Optimiser (process scope - "Primary Processing"): to optimise of recovery, R_{SO4}, and acid:fluorspar ratio, A. The task passes supervisory demands for R_{SO4} and A to the separation control reactor control tasks respectively.
- Reactor Control (process scope - "Reactor System"): to implementing supervisory demand setting acid:fluorspar ratio the reactor control task is responsible for reactor temperature control. The task is also required to provide feedback to the optimiser of its operating costs.
- Separation Control (process scope - "Primary Separation"): to implement the recovery requirements from the optimiser and provide feedback on the operating costs of the separation section.
- Pressure Control (process scope - "HF process"): to maintain a safe pressure in the process.

4.2.3 Determining Necessary Safety Margins

An important observation is that two failure conditions (*FC-5 & FC-6*) are active constraints in the process optimisation. For a failure condition that is an active constraint the control objective is to operate as close to the constraint as possible. Failure prevention requires that a there be a safe margin between the operating point and failure constraints. A safety margin has a direct impact on optimal operation. Analysis of the demands and planned control strategies can provide more information on the necessary safety margins.

Temperature Control: The function of the temperature control is to prevent the reactor wall from overheating and the basic control mechanism is to regulate heat input. Throughput, which is under the control of the supervisory demand *SD-1*, is a significant disturbance for this control task. However, the supervisory demand provides advance notice of changes in throughput so it is possible to schedule changes in heat input to match throughput and avoid significant temperature deviations. The speed at which temperature control can respond to the feedforward signal limits the rate at which production rate can be changed.

The variation in CaF_2 concentration identified in external demand *ED-1* will have an effect on conversion and therefore on the heat balance. Also uncertainty in the operation of the reactor will also lead to variations in conversion. An approximate model of the relationship between conversion and the reactor wall temperature can be derived from a simplified heat balance (see appendix B.4),

$$\begin{aligned} M_W C_{p,W} \bar{T}_W \mathrm{s} &= \bar{Q}_R - UA\bar{T}_W \\ &\quad + UA \frac{\bar{X} F_{CaF} \Delta H_R + UA\bar{T}_W}{M_R C_{p,R}\mathrm{s} + \mathrm{FC_{p,R}} + \mathrm{UA}} \end{aligned} \tag{4.13}$$

The relationship between T_W and X is second order and rearranging the equation into the standard form for a second order system (*ie.* $\bar{T}_W = K\bar{X}/(\tau^2\mathrm{s}^2 + 2\tau\zeta\mathrm{s} + 1)$) gives the constants:

$$\text{System Gain}: \; K = \Delta H_R / C_{p,R} \tag{4.14}$$

$$\text{Time Constant :} \quad \tau = \sqrt{\frac{M_W C_{p,W} M_R}{UAF}} \tag{4.15}$$

$$\text{Damping Factor :} \quad \zeta = \frac{1}{2\tau} \times (UAM_W C_{p,W} + FC_{p,R} M_W C_{p,W} + UAC_{p,R} M_R) \tag{4.16}$$

For a second order system the output gain to a sinusoidal input is given by,

$$A_{Tw} = \frac{KA_X}{\sqrt{(1-(\omega\tau)^2)^2 + (2\zeta\tau\omega)^2}} \tag{4.17}$$

$$\begin{aligned} \text{where } A_X &\equiv \text{Amplitude of input disturbance} \\ A_{Tw} &\equiv \text{Amplitude of temperature response} \\ \omega &\equiv \text{Frequencyofinputdisturbance} \end{aligned}$$

The time constant can be split into two parts: the reactor residence time (M_R/F) and the thermal inertia of the reactor wall ($M_W C_{p,W}/UA$). Reactor residence time is known to be approximately 60 minutes and if the thermal inertia is of the same order then the combined time constant (the root of the product of the two terms) will also be approximately 60 minutes. It is safe to expect that the system will not be underdamped (*ie.* $\zeta < 1$) and assuming perfect damping (*ie.* $\zeta = 1$) the normalised amplitude ratio is

$$AR_N = \frac{A_{Tw}}{KA_X} = \frac{1}{\sqrt{(1-3600.\omega^2)^2 + 14400\omega^2}} \tag{4.18}$$

The normalised amplitude ratio provides a measure of the effect of dynamics on damping an uncontrolled disturbance in X. Combined with a steady state analysis to determine the gain for the expected magnitude of input disturbances an estimate of the necessary safety margin can be derived. Dynamic effects alone will damp temperature disturbances by 99% for frequencies faster than 5 minutes per cycle which is close to the time scale of the composition demand *ED-1*. A significant safety margin with respect to *ED-1* is therefore expected to be required. Also the analysis indicates that control for demands of similar frequency is not necessary.

Pressure Control: the pressure control strategy proposed regulates the gas and vapour flowrates through the whole process. Precise regulation of the pressure is therefore important not only to improve safety margins but also to reduce the demands on the separation control system. As for the temperature control a system of Laplace equations can be derived to model the pressure dynamics (see appendix B.3),

$$\bar{P}_R = \frac{RT_R(1 + K_F\bar{P}_S) + \bar{F}_{CaF}\bar{X}}{V_R s + 2RT_RK_F} \tag{4.19}$$

$$\bar{P}_S = \frac{RT_S(1 + K_F(\bar{P}_R + \bar{P}_F))}{V_S s + 2RT_SK_F} \tag{4.20}$$

where $P_S \equiv$ Separation system pressure

$P_F \equiv$ Downstream pressure created by vent fan

$K_F \equiv$ Pipe flow constant

Deriving values for the model parameters from the current level of process definition is difficult. As the author's design experience is not sufficient to estimate the parameters, analysis is restricted to a qualitative evaluation. The time constants for the separate equations are both of the form $V/(RTK_F)$. At one atmosphere V/RT is equal to the molar hold up of a system which is expected to be small. Also if pressure drops are to be kept small K_F is expected to be large. Therefore on a qualitative basis the pressure control system is expected to have a fast response and is likely to be most sensitive to demands of higher frequency than those identified so far.

In an industrial environment a stronger base of process design experience is available. Design experience and a history of previous designs would assist in determining reasonable estimates of the model parameters. A numerical analysis of the pressure control system would then be possible.

4.2.4 Secondary Operating Strategies

Within the operating task structure that has been developed so far if conventional pressure or temperature control fails the only planned action is to shut down the process. It is desirable to avoid such an extreme response if possible. To reduce the need for shutdown secondary operating strategies are considered. A secondary operating strategy uses additional control mechanisms to gain more control over a failure condition. Typically additional control is obtained by overriding supervisory demands given by optimisers.

Overpressure in the reactor can be compensated both by increasing the rate at which gases are removed (the conventional control) and by reducing the reaction extent in the reactor (*eg.* by cutting the feed to the reactor). Within the normal operating task hierarchy the supervisory demand *SD-1*, which sets the production rate, ultimately governs reaction extent. As a backup strategy in the case of overpressure reaction extent is governed by a "Pressure recovery" task. If pressure reaches a given alarm level control is switched to the pressure recovery task which will attempt to compensate for the overpressure by cutting the reaction extent. Either operation will return to normal and operation will switch back to optimisation or pressure will continue to rise until the alarms in the primary executive initiate a plant shutdown. Switching between normal optimisation and pressure recovery requires an executive task which is interposed between the stock control task and the recycle optimisation (figure 4.11). The pressure recovery task will also be decomposed into a hierarchy of operating tasks but many of the tasks will be in common with the optimisation hierarchy (*eg.* for regulation of reactor temperature and composition control).

4.2.5 Start up Procedure

The preliminary start up procedure defined at the input/output level decomposes the operation into start up of the reactor and primary separation followed later

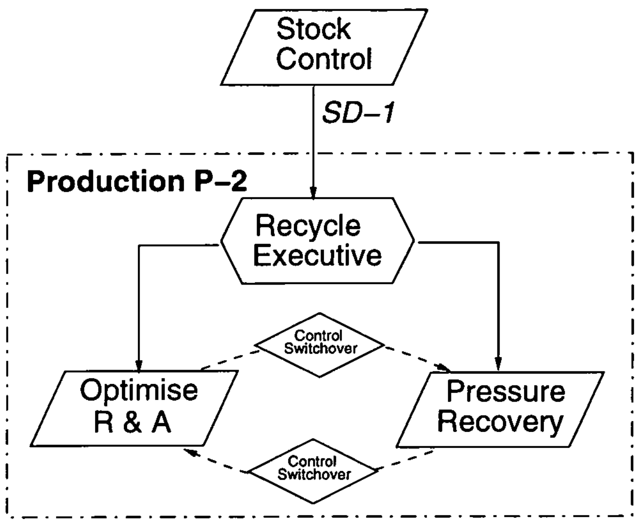


Figure 4.11: Secondary operating tasks for pressure recovery

by startup of the rectifier section. It is now known that the reactor has a long residence time and getting it to its operating temperature is likely to be the slowest part of the start up operation. To ensure that no loss of HF occurs the primary separation must be active before the reactor offgas starts to flow. If the reactor can be preheated without having to charge it with feed then startup is arranged as shown in figure 4.12. Preheating brings the reactor as close as possible to its production temperature. As the reactor approaches its operating temperature the primary separation is started up running at maximum recycle. Only once the primary separation is running is acid and fluorspar fed to the reactor. Finally the reactor and separation are brought up to production grade.

4.3 Primary Separation System

Following the design of the recycle structure the process design may be decomposed into the separate sections of reactor, primary separation and and rectification. Figure 4.13 shows the structure for the primary separation system. Primary separation is divided into three sections:

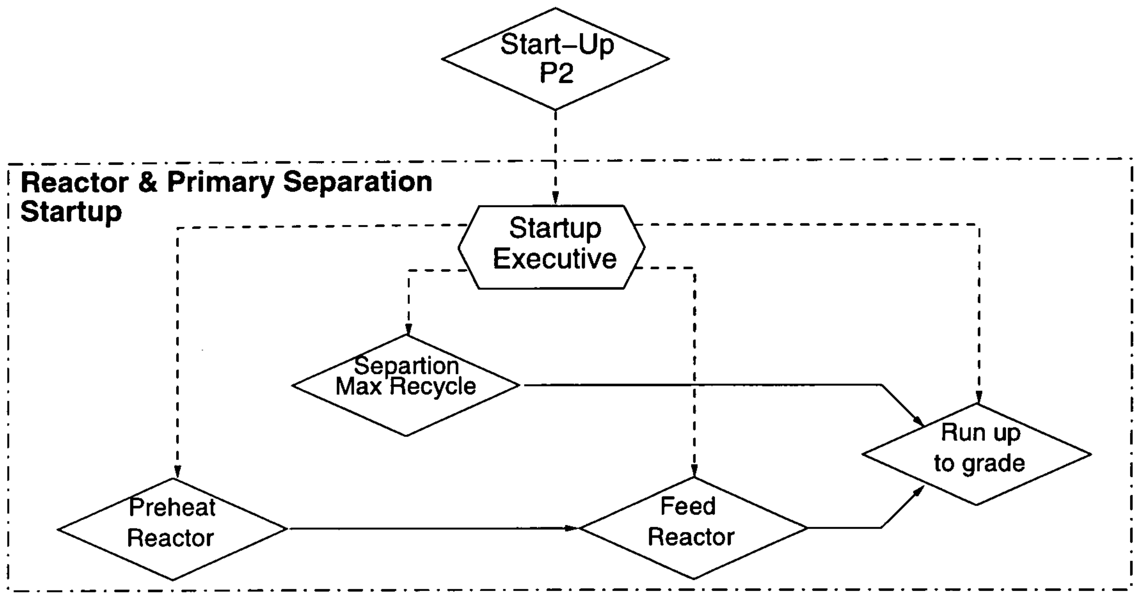


Figure 4.12: Startup of Reactor and primary separation systems

- Acid Quench: to cool the reactor offgas and condense out the sulphuric acid for recycle.
- Condenser: to separate the HF from the low boiling point impurities.
- HF Absorber: to recover HF from vapour effluent. Cooled H_2SO_4 is used to extract the HF from the vapour stream.

The acid used for recovery in the absorber is fed back to the quench system where the HF is boiled off by the hot reactor gases.

When decomposing the process design the context of each section as part of a whole process is maintained (the grayed sections of the flowsheet). Maintaining the process context serves two purposes, first it connects the inputs of the section with external demands entering the process. For example the uncertainty in fluorspar composition (external demand *ED-1*) will affect the reactor offgas composition entering the separation section. A simplified model of the reaction system relates the stream entering the separation section with the external demand. The second purpose is to identify any feedback effects that may exists between the outputs of the section and its inputs, *eg.* build up of impurities in recycles.

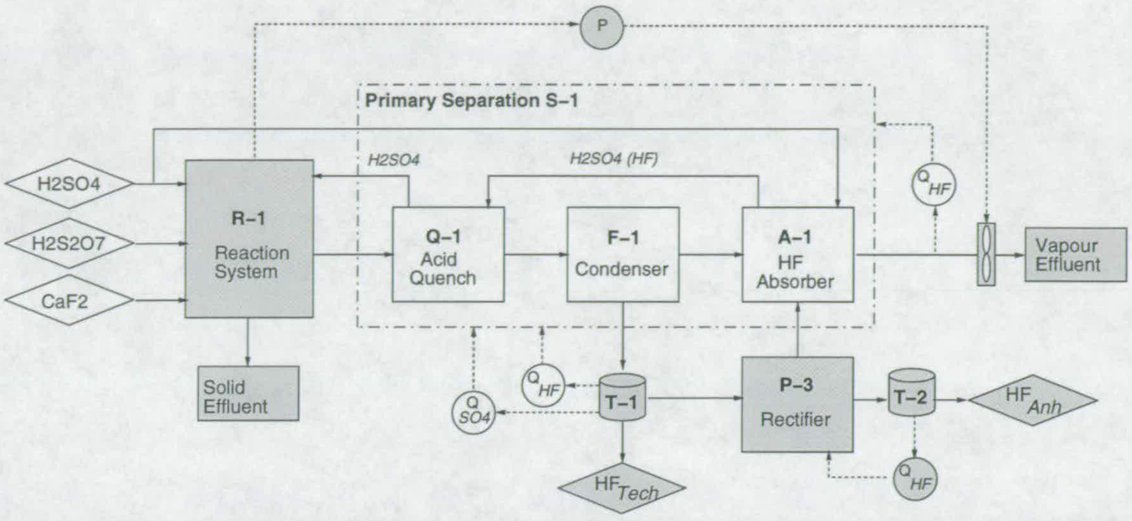


Figure 4.13: Structure of the primary separation system

The decomposition of the production tasks at the recycle design level defined a 'separation control' task to manage the primary separation system. The degrees of freedom available for local optimisation are restricted by various constraints. The failure conditions *FC-1* and *FC-4* set limits on the HF composition in the technical grade product and effluent stream (marked as open control loops in figure 4.13). Also as the rectifier of section *P3* is only designed to remove components lighter than HF there is a limit imposed proportion of H_2SO_4 which can be present in the technical grade product so that failure condition *FC-2* can be satisfied. Finally the recovery of acid from the reactor offgas is supplied as a supervisory demand by the production optimisation task.

A natural assignment of the control objectives to process operations is:

- Quench: H_2SO_4 Recovery & H_2SO_4:HF product ratio.
- Condenser: HF composition in product.
- Absorber: HF Composition in effluent

There is no apparent benefit to including an additional layer of process optimisation so the separation control task is decomposed into three reduced regulatory tasks, one for each operation. An expansion of the flowsheet for each operation with control loops marked is shown in figure 4.14. To ensure maximum recovery of

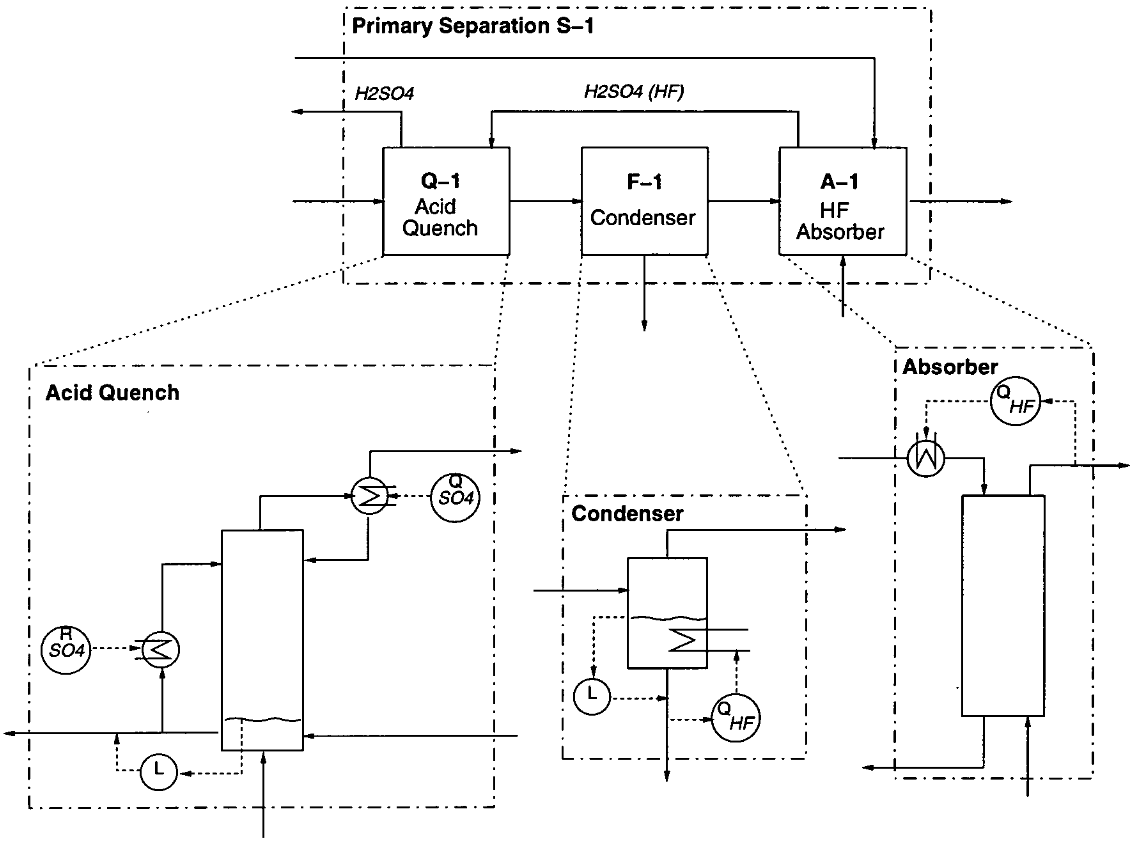


Figure 4.14: Expanded design of separation operations

HF the flowrate of acid to the absorber is set to the maximum that can be achieved without flooding the column. Similarly the liquid recycle rate in the quench is also set to the maximum flow that can be achieved.

4.4 Reactor System

The basic structure of the reactor system is straightforward, the oleum and sulphuric acid are combined in a pre-mixer which is then fed along with the fluorspar to the reactor.

As with the primary separation most of the regulatory tasks for this section have been pre-specified. The stock control task determines the feed rate of fluorspar, the optimal ratio of acid to fluorspar is calculated by a supervisory task. New control tasks to be addressed at this level are the regulation of oleum feed rate and preheating of the feed acid to the reactor.

The oleum is used to keep the water content of acid that is fed to the reactor to a minimum. The flowrate of oleum is therefore set by a control monitoring the fraction of water in the reactor feed acid. The reaction of oleum with the water partially preheats the reactor feed acid. Heating utility is still required to provide additional preheating. The flowsheet published in the BUSS patent [62] has a heat exchanger placed on the fresh feed to the pre-mixer. The placement is not ideal as a means for controlling the temperature of the acid fed to the reactor. Instead it has been chosen to place the heat exchanger after the mixer which provides a more direct means of temperature control.

The final structure for the reactor system with its regulatory control system is shown in figure 4.15.

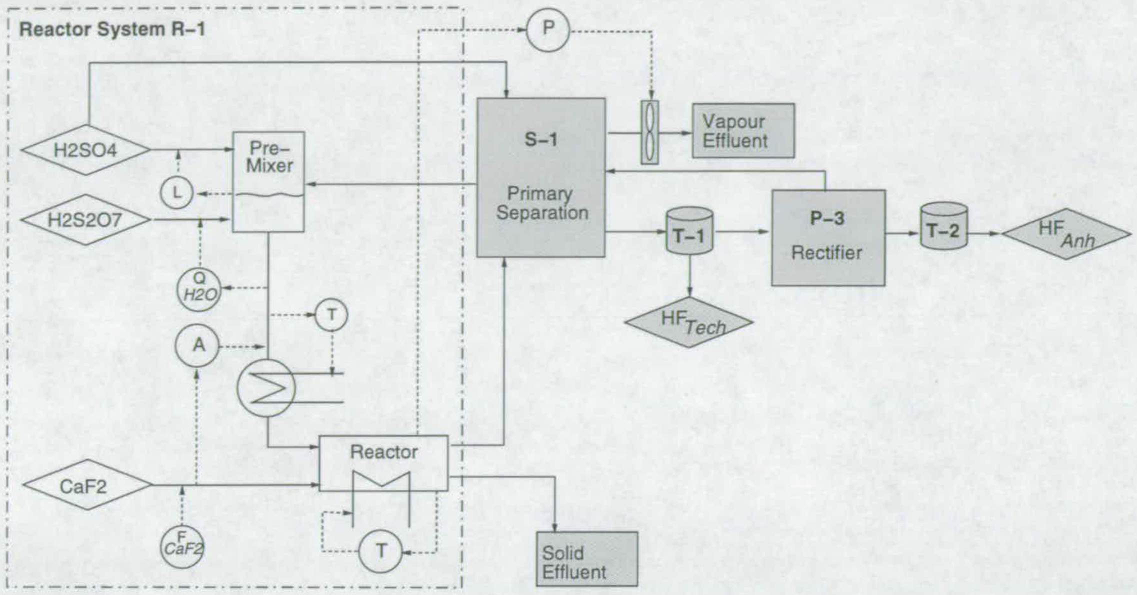


Figure 4.15: Reactor system design

4.5 The Rectification Section

The rectifier section has the standard configuration of a simple distillation system (figure 4.16). The stock control task sets the production rate for this system. As was discussed in the development of the stock control strategy the rectifier section is required to have a fast response to changes in production rate. The primary failure condition pertinent to the rectifier section is *FC-2* which sets a composition limit on the product stream.

A degree of freedom is still left in the operation of the rectifier *ie.* the recovery of HF to the product. Any HF lost in the top product of the rectifier is recycled and must be recovered in the absorber of the primary separation system. To determine the optimal recovery for the rectifier it is therefore necessary to account for the interaction with the absorber. The implication is that the original hierarchical structure is inappropriate. A revised hierarchy for the on-line optimisation is overlayed on the process in figure 4.17.

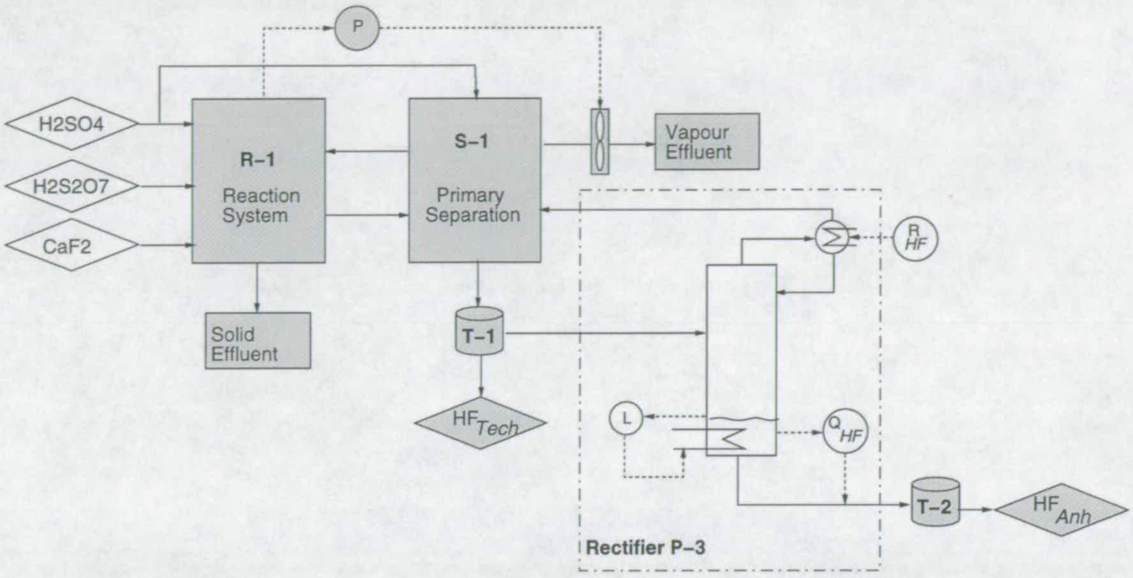


Figure 4.16: Rectifier section design

If operation of the rectifier and absorber are interdependent the implication is that the design of the rectifier and absorber are also interdependent. The new decomposition of the operating system is therefore a more appropriate decomposition for the process design as well. The original hierarchical structure is more appropriate for the startup of the process. Therefore for the same process design two separate decomposition strategies are used in the operating system design. However, the process design will only follow one decomposition path, and concurrent design for an operating tasks which employs a different decomposition will not be straightforward.

4.6 Final Operating System Design

Figure 4.18 shows the complete HF process with the basic regulatory control system marked. The flowsheet is derived from that given in the BUSS patent [62]. The development of the process at this stage focusses on equipment design for each of the unit operations. For the operating system design it is necessary to identify

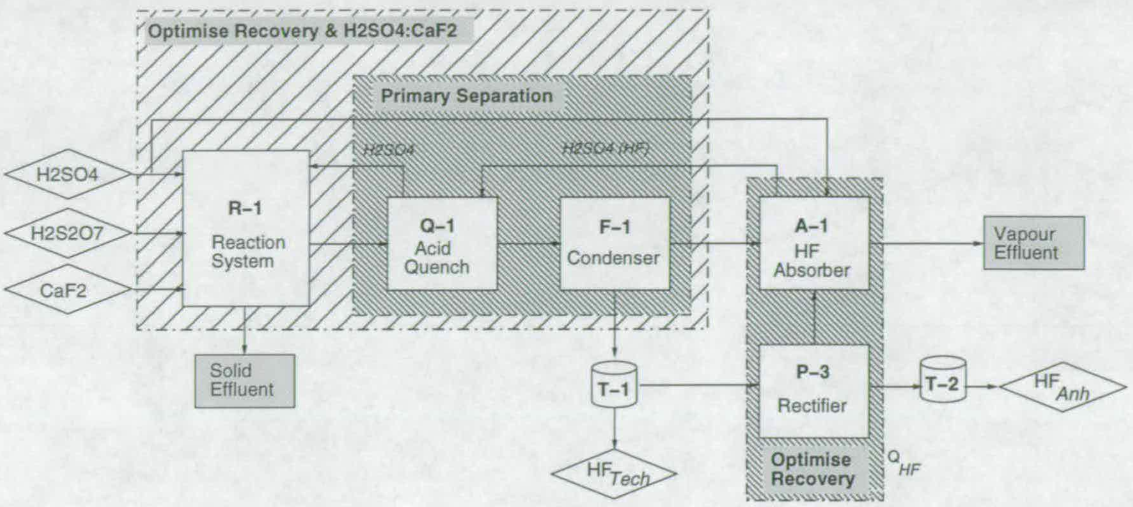


Figure 4.17: Revised operating task hierarchy

secondary measurements for estimation of properties such as composition which cannot be conveniently measured online. As the detailed designs for the process equipment become available the models used in the operating tasks and control schemes should be appropriately refined.

Operability analysis at this stage can make much more use of the methods discussed in chapter 2. The preliminary operating system design has identified some of the constraints and demands on operation. The final equipment design introduces further constraints which can be combined with the failure conditions and demands for a complete flexibility analysis. In fixing equipment sizes more well founded dynamic models can be formulated to validate previous analyses and to perform more complete dynamic resilience tests.

4.7 Conclusions from Case Study

The design of the hydrofluoric acid plant has been developed following the hierarchical steps of the Douglas [57] methodology. In step with the hierarchical development a basic strategy for on-line optimisation and regulatory control of

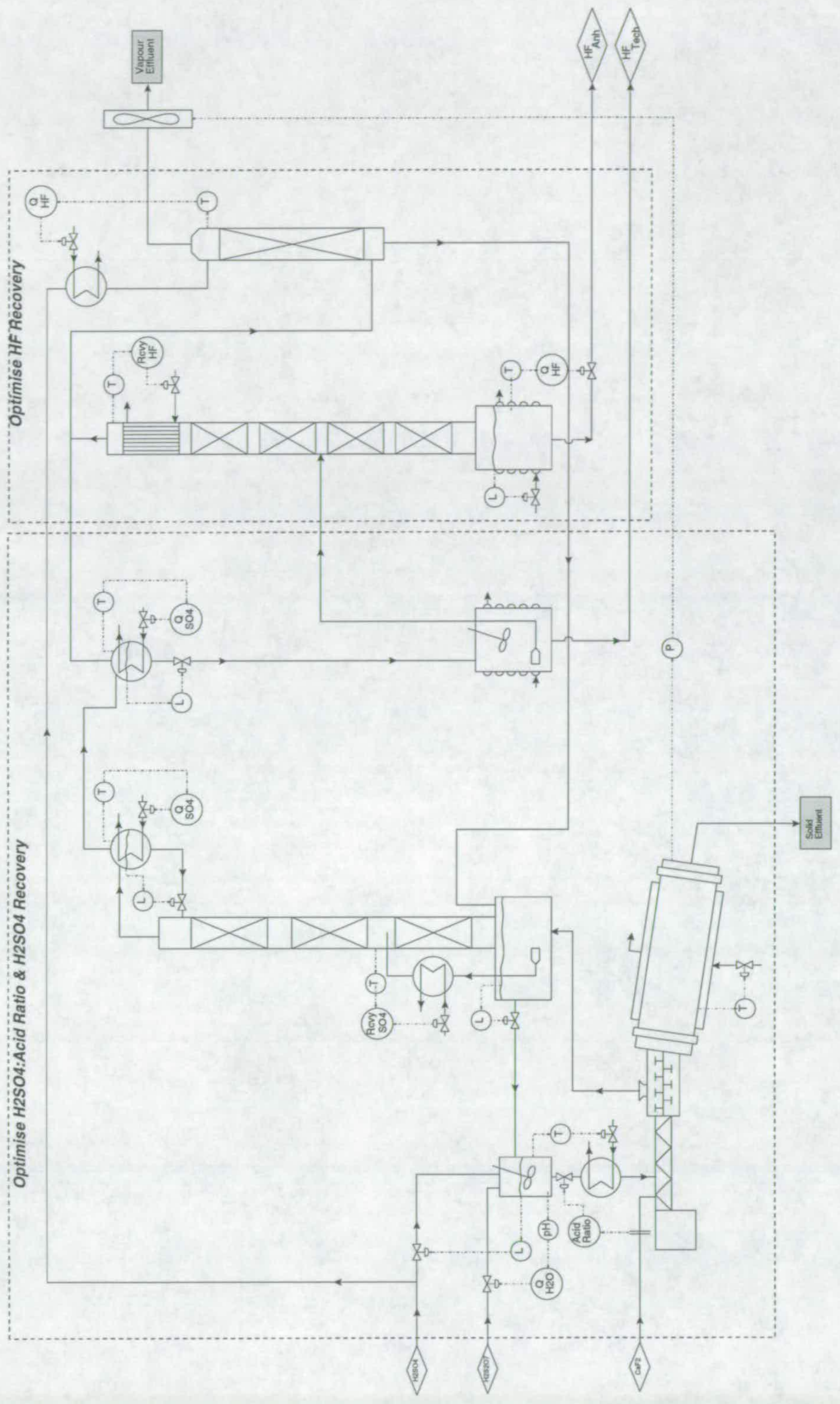


Figure 4.18: Complete HF process with controls

the process has been developed (see figure 4.18). A basic procedure for start up has been also proposed as part of the operating system design. In addition the use of secondary operating strategies to provide a layered response to failure recovery has been illustrated with respect to the pressure control system. The appropriate structure for the operating task hierarchy has not always been clear cut. The pressure control is an example of an operating task that did not fit naturally into the task hierarchy.

A comparison of the proposed operating system design with current control systems is difficult due to the lack of published data on current process performance and operation. The design of the HF process and its control system has evolved over several years. In the process of review and redesign operability problems are expected to be reduced. In following a hierarchical approach the basis for a rational operating strategy has been developed in reasonable time.

Modelling restrictions have been a significant limit to the amount of operability analysis that has been possible during preliminary design. For example, performing flexibility analysis when few of the equipment constraints are known has limited value. However, by working from an understanding of the operating strategy it has been possible to consider focussed studies. While these studies are not as general as the methods reviewed in chapter 2 the link between the process design and control performance is easier to understand. Methods for formulating lumped models for whole process sections, including the prediction of the model parameters, are needed to improve operability analysis for preliminary process designs.

Chapter 5

Summary, Conclusions and Future Directions

5.1 Summary

In chapter 2 a variety of approaches to operability analysis were reviewed. The dominant areas of study in process operability are flexibility and dynamic resilience. Putting together the tools that have been developed in these fields to derive a balanced conclusion is a non-trivial task for the process designer. The evaluation and interpretation of dynamic resilience measures is a particular challenge for process engineers who are not familiar with the underlying control theory. The proposal of this thesis was that, to assist in operability analysis and improve design integration, concurrent design of the process and the control system should be considered.

To support concurrent design a framework for the hierarchical design of a *process operating system* was set out in chapter 3. A process operating system was defined as the complete collection of control schemes, alarms and operating procedures used for managing the process through all phases of operation. The two primary functions of the operating system were seen to be:

- Failure Management: the prevention of and recovery from operating failures.
- Process Optimisation: optimisation of process operation.

The design of an integrated operating system was approached by decomposing the problem into a hierarchy of operating tasks. Three classes of operating task were used in constructing the hierarchy:

- Regulatory tasks: for optimising operation at a steady state.
- Transition tasks: for transferring the process from one regulatory state to another.
- Executive tasks: which manage the response to discrete events such as alarms and failures.

An operating system is implemented by designing a control scheme for each of the tasks in the operating task hierarchy.

A case study on the design of an operating system for a hydrofluoric acid plant was presented in chapter 4. The study followed the hierarchical steps proposed by Douglas [57] for preliminary process design. The resulting operating system design addressed both on line optimisation and failure management. Not all operating tasks fitted naturally into the hierarchical decomposition developed and some consideration of alternative decompositions was needed. The application of general methods for operability analysis during the preliminary design stages was found to be limited by the degree of modelling that was possible. However, by working from an understanding of the planned operating strategy, focussed operability studies could be developed for the preliminary design.

A knowledge based representation has been proposed to support operating system design. Particular attention was given to the problem of supporting concurrent design of the process and operating system. The representation developed links process design alternatives with operating system design alternatives through their association with a common operating task. Though the representation was not employed in the case study it is suitable for extension and incorporation into a more general design support system.

5.2 Conclusions and Future Directions

Process operability is an important design issue. Methods for analysing operability are only part of the solution to addressing this issue. Integrating the development of the process, the control system and the operating procedures is an important additional part of the solution. While on occasion there may be industrial projects that develop the control system design or operating procedures in step with the process design, in general the three are treated as independent activities. The proposed framework for hierarchical operating system design unifies the design of the control system and operating procedures such that they can be developed concurrently with the design of the process.

A case study has shown that the general approach of decomposing operation into a hierarchy of operating tasks is effective in developing an integrated operating system design. Concurrent design also made it possible to formulate focussed operability studies during the preliminary design of the process which are valuable in avoiding potential operability problems.

By hierarchical design of an operating system a unified operating strategy can be developed. Also by timely identification of operability requirements more appropriate process designs can be produced. Integrated design of a process and its operating system is thus a significant aid to designing operable plants.

Further work is proposed in three main areas:

Further case studies. To refine and extend the framework that has been proposed it is important to pursue further case studies. In particular more consideration needs to be given to the design of transition tasks such as startup and shutdown. Also a strictly hierarchical approach to design is restrictive. The evolution of a process design does not always follow hierarchical steps. An appropriate approach to operating system design in such cases needs to be developed.

Models for preliminary operating system design. To improve the operability analysis at the preliminary design stages it is important to be able to formulate suitable models. Some simple models were derived in the case study but their use was limited by the ability to predict appropriate values for the model parameters. A system for estimation of process dynamics based on statistical correlations would be a useful aid.

Knowledge based system support. The focus of the knowledge representation has been on linking process and operating system design. Other aspects of the representation are less developed and could benefit from enhancement. Transferring the representations that have been developed to a more complete design support system is important for the further development of the knowledge representation work.

Appendix A

Bibliography

1. C. J. Ryskamp. Explicit versus implicit decoupling in distillation control. In *Chemical Process Control II*, page 361, 1982.

2. J. M. Douglas and W. R. Fisher. Analysis of process operability at the preliminary design stage. *Comp.& Chem. Eng.*, 9(5):499, 1985.

3. I. E. Grossmann and M. Morari. Operability, resiliency & flexibility: Process design objectives for a changing world. In *Foundations of Computer Aided Process Design*, page 931, 1983.

4. I. E. Grossmann and R. E. Swaney. An index for operational flexibility in chemical process design: Part 1. *A. I. Ch. E. Journal*, 31:621, April 1985.

5. Y. Arkun and W. G. Etzkorn. Computer aided operability analysis via interactive graphics. *Comp.& Chem. Eng.*, 5(4):233, 1981.

6. I. E. Grossmann and C. A. Floudas. Active constraint strategy for flexibility analysis in chemical processes. *Comp.& Chem. Eng.*, 11(6):675, 1987.

7. M. Morari, A. K. Saboo, and D. C. Woodcock. Design of resilient process plants -(VIII). *Chem.Eng.Sci*, 40(8):1553–1565, 1985.

8. I. E. Grossmann, K. P. Halemane, and R. E. Swaney. Optimization strategies for flexible chemical processes. *Comp.& Chem. Eng.*, 7(4):439, 1983.

9. W. R. Johns and M. Lakshmanan. Optimal equipment sizing under technical uncertainty. In *PSE'85*, number 92 in I. Chem. Eng. Symposium Series, page 223, 1985.

10. R. K. Malik and R. R. Hughes. Optimal design of flexible chemical processes. *Comp.& Chem. Eng.*, 3:473, 1979.

11. W. R. Johns, G. Marketos, and D. W. T. Rippin. The optimal design of chemical plant to meet time varying demands in the prescence of technical and commercial uncertainty. *Trans. I. Chem. E.*, 56:249–257, 78.

12. G. Marketos. *The optimal design of chemical plant considering uncertainty and changing circumstances.* PhD thesis, ETH 5607, ETH, Zurich, Switzerland, 1975.

13. E. N. Pistikopoulos and I. E. Grossmann. Optimal retrofit design for improving process flexibility in linear systems. *Comp. Chem. Eng*, 12(7):719–731, 1988.

14. E. N. Pistikopoulos and I. E. Grossmann. Stochastic optimization of flexibility in retrofit design of linear systems. *Comp. Chem. Eng*, 12(7):1215–1227, 1988.

15. E. N. Pistikopoulos and T. A. Mazzuchi. A novel flexibility analysis approach for process with stochastic parameters. *Com. Chem. Eng.*, 14(9):991–100, 1990.

16. O. A. Asbjornsen. Control and operability of process plants. *Comp. Chem. Eng.*, 13(4/5):351–364, 1989.

17. M. Morari, D. F. Marselle, and D. F. Rudd. Design of resilient processing plants - II. *Chem. Eng. Sc.*, 37(2):259, 1982.

18. M. Morari and A. K. Saboo. Design of resilient processing plants - IV. *Chem. Eng. Sc.*, 39(3):579, 1984.

19. M. Morari, A. K. Saboo, and R. D. Colberg. Reshex:- an interactive software package for the synthesis and analysis of resilient hen's, parts I & II. *Comp.& Chem. Eng.*, 10(6):577, 1986.

20. B. Linhoff and E. Kotjabasakis. Sensitivity tables for the design of flexible processes (1). *Chem. Eng. Res. Des.*, 64:97, May 1986.

21. J. Calandranis and G. Stephanopoulos. Structural operability analysis of heat exchange network. *Chem. Eng. Res. Des.*, 64:347, September 1986.

22. Y. Shimizu. A plain approach for dealing with flexibility problems in linear systems. *Comp. Chem. Eng.*, 13(10):1189–1191, 1989.

23. I. E. Grossmann and D. A. Straub. Recent developments in the evaluation and optimisation of flexible chemical processes. Technical Report edrc 06-101-91, EDRC, 1991.

24. P. J. Fryer, W. R. Paterson, and N. K. H. Slater. Robustness of fouling heat exchanger networks and its relation to resilience. *Chem. Eng. Res. Des.*, 65:267–271, May 1987.

25. M. Morari. Design of resilient processing plants - III. *Chem. Eng. Sc.*, 38(11):1881, 1983.

26. Y. Arkun. Dynamic process operability: Important problems, recent results and new challenges. In *Chemical Process Control III*, page 323, January 1986.

27. M. Morari and B. R. Holt. Design of resilient processing plants - V. *Chem. Eng. Sc.*, 40(7):1229, 1985.

28. M. Morari and B. R. Holt. Design of resilient processing plants - VI. *Chem. Eng. Sc.*, 40(1):59, 1985.

29. J. D. Perkins and M. P. F. Wong. Assessing controllability of chemical plants. In *PSE'85*, page 481, 1985.

30. R. D. Johnston and G. W. Barton. Analysis of structural controllability. *Int. J. Control.*, 41(6):1477–1491, 1985.

31. J. D. Perkins and L. W. Russell. Towards a method for diagnosis of controllability and operability problems in chemical plants. *Chem. Eng. Res. Des.*, 65:453, November 1987.

32. F. D. Carvallo, A. W. Westerberg, and M. Morari. An index of controllability for linear deterministic processes. Technical Report 06-58-89, EDRC, 1989.

33. T. J. McAvoy and T. E. Marlin. A short-cut method for process control and operability analysis. In *Chemical Process Control III*, page 369, January 1986.

34. T. J. McAvoy, G. Stanley, and M. M. Galarraga. Shortcut operability analysis 1. the relative disturbance gain. *Ind. Eng. Chem. Proc. Des. Dev.*, 24:1181, 1985.

35. S. Skogestad and E. Wolf. Controllability measures for disturbance rejection. In *Interactions between Process Design and Process Control*, J. D. Perkins, editor, pages 23–29. IFAC, Pergamon Press, September 1992.

36. M. Morari and S. Skogestad. Design of resilient processing plants - IX. *Chem. Eng. Sc.*, 42(10):2425, 1987.

37. Y. Arkun, A. Palazoglu, and B. Manousiouthakis. Design of chemical plants with improved dynamic operability in an environment of uncertainty. *Ind. Eng. Chem. Proc. Des. Dev.*, 24:802, 1985.

38. Y. Arkun and A. Palazoglu. A multi-objective approach to design of chemical plants with robust dynamic operability characteristics. *Comp.& Chem. Eng.*, 10(6):567, 1986.

39. I. D. L. Bogle and M. Rashid. An assessment of dynamic operability measures. *Comp. Chem. Eng.*, 13(11/12):1277–1282, 1989.

40. Y. Arkun and S. Ramakrishnan. Structural sensitivity analysis in the synthesis of process control systems. *Chem. Eng. Sci.*, 39(7/8):1167–1179, 1984.

41. J. R. Rivas, D. F. Rudd, and L. R. Kelly. Computer aided safety interlock systems. *A.I.Ch.E Journal*, 20(2):311–319, March 1974.

42. J. R. Rivas and D. F. Rudd. Synthesis of failure safe operations. *A.I.Ch.E Journal*, 20(2):320–325, March 1974.

43. R. H. Fusillo and G. J. Powers. A synthesis method for chemical plant operating procedures. *Comp.& Chem. Eng.*, 11(4):369, 1987.

44. R. H. Fusillo and G. J. Powers. Operating procedure synthesis using local models and distributed goals. *Comp.& Chem. Eng.*, 12(9/10):10230–1034, 1988.

45. R. H. Fusillo and G. J. Powers. Computer aided planning of purge operations. *A.I.Ch.E Journal*, 34(9/10):558–566, 1988.

46. R. Lakshmanan and G. Stephanopoulos. Synthesis of operating procedures for complete chemical plants I - hierarchical structured modelling for nonlinear planning. *Comp. Chem. Eng.*, 12(9/10):985–1002, 1988.

47. R. Lakshmanan and G. Stephanopoulos. Synthesis of operating procedures for complete chemical plants II - a nonlinear planning methodology. *Comp. Chem. Eng.*, 12(9/10):1003–1021, 1988.

48. R. Lakshmanan and G. Stephanopoulos. Synthesis of operating procedures for complete chemical plants III - planning in the presence of qualitative mixing constraints. *Comp. Chem. Eng.*, 14(3):301–312, 1990.

49. V. Aelion and G. J. Powers. A unified strategy for the retrofit synthesis of flowsheet structures for attaining or improving operating procedures. *Comp. Chem. Eng.*, 15(5):349–360, 1991.

50. M. Morari, J. A. Mandler, and J. H. Seinfeld. Control system design for a fixed bed methanation reactor. *Chem. Eng. Sc.*, 41(6):1577, 1986.

51. M. Morari, B. R. Holt, K. Shimizu, and R. S. H. Mah. Assessment of control structures for binary distillation columns with secondary reflux and vapourization. *Ind. Eng. Chem. Proc. Des. Dev.*, 24:852, 1985.

52. Y. Arkun, A. Palazoglu, and B. Manousiouthakis. Robustness analysis of process control systems: A case study of decoupling control in distillation. *Ind. Eng. Chem. Proc. Des. Dev.*, 23:93, 1984.

53. T. E. Marlin, J. D. Perkins, G. W. Barton, and M. L. Brisk. Benefits from process control: result of a joint industry study. *J. Proc. Cont.*, 1:68–83, March 1991.

54. G. W. Barton, W. K. Chan, and J. D. Perkins. Interaction between process design and process control: the role of open loop indicators. *J. Proc. Cont*, 1:161–170, May 1991.

55. G. W. Barton and T. C. Nguyen. Improved control via process modification. In *Third Symposium on Process Systems Engineering*, pages 39–43, 1988.

56. E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115, 1974.

57. J. M. Douglas. *Conceptual Design of Chemical Processes.* McGraw-Hill, 1988.

58. J. W. Ponton and D. M. Laing. A hierarchical approach to the design of process control systems. *Trans. I. Chem. E.*, 71:181–188, March 1993.

59. J. M. Douglas. A hierarchical decision procedure for process synthesis. *A. I. Ch. E. J.*, 31(3):353, March 1985.

60. H. C. Fielding and B. E. Lee. Hydrofluoric acid, inorganic fluorides and fluorine. In *Modern Inorganic Chemicals Industry*, R. Thompson, editor, pages 149–167. The Chemical Society, March 1977.

61. F. A. Lowenheim and M. K. Moran. Hydrofluoric acid. In *Faith, Keyes, and Clark's Industrial Chemicals*, pages 464–467. John Wiley & Sons, 4 edition, 1975.

62. Process for the continous production of hydrofluoric acid. Patent 1216270 (UK), patent held by Buss AG, October 1967.

63. A. Struthers. *A Knowledge Based Approach to Process Engineering Design.* PhD thesis, University of Edinburgh, 1990.

64. D. Hutton. *Knowledge Based Flowsheeting.* PhD thesis, University of Edinburgh, 1991.

Appendix B

Derivation of Models for Case Study

B.1 Storage requirements for constant production rate

If the nominal production rate is S_{nom} then if sales drop to S_{min} for a sustained period of T_D then stock levels will increase by

$$\Delta V = T_D * (S_{nom} - S_{min}) \tag{B.1}$$

If on the other hand sales increase to S_{max} the stock levels will drop by

$$-\Delta V = T_D * (S_{max} - S_{nom}) \tag{B.2}$$

To guarantee that there will not be a deficit or excess of stock it therefore necessary to have a storage capacity, V, of

$$V = T_D * (S_{max} - S_{min}) \tag{B.3}$$

and it will be necessary to maintain a nominal stock level of

$$V_{nom} = T_D * (S_{max} - S_{nom}) \tag{B.4}$$

B.2 Storage requirements with flexible production rates

First consider a change in sales rate from its nominal value S_{nom} to its maximum S_{max}. If the rate at which production rate can be changed is r then the time required to increase production rate to match sales rate is $(S_{max} - S_{nom})/r$. The lowest peak in stock level occurs when the production rate change is started in advance of the sales rate change by $(S_{max} - S_{nom})/2r$. The peak level will occur half way through the change in production rate when

$$\begin{aligned} \Delta V &= 0.5.(S_{max} - S_{nom})/2r.(S_{max} - S_{nom})/2 && \text{(B.5)} \\ &= (S_{max} - S_{nom})^2/8r && \text{(B.6)} \end{aligned}$$

Similarily for a change in sales rate from S_{nom} to S_{min} the maximum drop in stock level will be

$$-\Delta V = (S_{nom} - S_{min})^2/8r \tag{B.7}$$

Assuming that $S_{max} - S_{nom} = S_{nom} - S_{min}$ then the necessary storage capacity required is

$$V = (S_{max} - S_{min})^2/4r \tag{B.8}$$

B.3 Derivation of dynamic model for pressure control

For a unit with a vapour flow in of $F_{v,in}$ and vapour flow out of $F_{v,out}$ a molar balance combined with the ideal gas law gives

$$\frac{dP}{dt} = \frac{RT}{V}(F_{v,in} - F_{v,out} + G_v) \tag{B.9}$$

where G_v represents the net rate of vapour generation in the unit. The upstream and downstream flows are determined by the pressures upstream (P_{up}) and downstream (P_{dn}),

$$F_{v,in} = K_F(P_{up} - P) \tag{B.10}$$

$$F_{v,out} = K_F(P - P_{dn}) \tag{B.11}$$

The pipe flow constant K_P will be treated as constant throughout the process. Substituting the flow relations back into the pressure derivative gives,

$$\frac{dP}{dt} = \frac{RT}{V}(K_F(P_{up} + P_{dn} - 2P) + G_v) \tag{B.12}$$

Substituting difference variables and taking the Laplace transform gives,

$$\bar{P} = \frac{RT(1 + K_F(\bar{P}_{up} + \bar{P}_{dn}) + \bar{G}_v}{(V\mathrm{s} + 2\mathrm{RTK_F})} \tag{B.13}$$

For the reactor G_v, the rate of vapour generation is related to the feed rate of fluorspar (F_{CaF}) and its conversion (X). Therefore the relationship for P_R is,

$$\bar{P}_R = \frac{RT_R(1 + K_F\bar{P}_S) + \bar{F}_{CaF}\bar{X}}{V_R\mathrm{s} + 2\mathrm{RT_RK_F}} \tag{B.14}$$

where $\bar{P}_S$ is the separation system pressure. For the separation system the downstream pressure is controlled by the exhaust fan. The rate of vapour generation is treated as constant so $\bar{G}_{v,S} = 0$, giving,

$$\bar{P}_S = \frac{RT_S(1 + K_F(\bar{P}_R + \bar{P}_F))}{V_S\mathrm{s} + 2\mathrm{RT_SK_F}} \tag{B.15}$$

B.4 Dynamic relation between conversion and reactor wall temperature

Assuming a lumped parameter system, the temperature of the reactor wall, T_W, is determined by the heat balance

$$\begin{aligned}
M_W C_{p,W}\frac{dT_W}{dt} &= Q_K - UA(T_W - T_R) \\
\text{where } M_K &\equiv \text{Mass of reactor wall} \\
C_p, W &\equiv \text{Heat capacity of reactor wall} \\
UA &= \text{Effective Area} \times \text{Heat Transfer Rate}
\end{aligned} \tag{B.16}$$

If a constant average flow, F, and heat capacity, $C_{p,R}$, are assumed for the material flowing through the reactor then a heat balance on the reactor contents gives

$$\begin{aligned}
M_R C_{p,R}\frac{dT_R}{dt} &= FC_{p,R}(T_I - T_R) + F.X.\Delta H_R + UA(T_s - T_R) \\
\text{where } M_R &\equiv \text{Mass of reactor contents} \\
T_I &\equiv \text{Inlettemperatureofreactants} \\
X &= \text{Reaction Conversion} \\
\Delta H_R &= \text{Heat of reaction}
\end{aligned} \tag{B.17}$$

$$\tag{B.18}$$

Only the relationship between T_W and conversion X is of interest. Taking Laplace transforms (assuming F and T_I are constant) and combining the heat balances gives,

$$\begin{aligned}
M_W C_{p,W}\bar{T}_W\mathrm{s} &= \bar{Q}_R - UA\bar{T}_W \\
&\quad + UA\frac{\bar{X}F\Delta H_R + UA\bar{T}_W}{M_R C_{p,R}\mathrm{s} + \mathrm{FC_{p,R}} + \mathrm{UA}}
\end{aligned} \tag{B.19}$$

From inspection the relation between T_W and X is second order. Rearraned into the standard format for a second order model (*ie.* $\bar{y} = K\bar{x}/(\tau^2\mathrm{s}^2 + 2\tau\zeta\mathrm{s} + 1)$) the model parameters are,

$$\text{System Gain}: \quad K = \Delta H_R / C_{p,R} \tag{B.20}$$

$$\text{Time Constant}: \quad \tau = \sqrt{\frac{M_W C_{p,W} M_R}{UAF}} \tag{B.21}$$

$$\begin{aligned}\text{Damping Factor}: \quad \zeta = & \frac{1}{2\tau} \times (UAM_W C_{p,W} \\ & + F C_{p,R} M_W C_{p,W} + UAC_{p,R} M_R)\tau\end{aligned} \tag{B.22}$$

Appendix C

A Knowledge Representation for Operating System Design

The development of the framework for concurrent hierarchical operating system design was parallelled by the development of a knowledge representation to support operating system design as part of an integrated knowledge based design support system. To an extent the development of the representations also guided the development of the design framework.

C.1 Overview of the Knowledge Representation Constructs

In this section provides an overview of the representation principles employed in this work. Also in this section we shall introduce the terminology that will be used to illustrate examples.

The principle programming language that was employed in this work was an experimental language developed locally by Struthers [63] called CLAP ("Combined Logic And Procedures"). The language is built upon Prolog, a declarative logic language. CLAP adds to this language object oriented facilities and special constructs for defining procedural operations. There are a wide range of object oriented languages in existence employing varied interpretations of the object oriented paradigm. At the core of object oriented programming is the encapsulation of structure and function in objects. Objects with the same function and structure are represented by a single *class*, specific examples of a class being referred to as *instances.* Within object oriented systems modularity is achieved through inheritance mechanisms. A class can be defined to have *parent classes* which means it will *inherit* the structure and functions of those classes. It is then only necessary to define those aspects of the new class which are specific to it. Where object oriented paradigms tend to differ is in the way the structure and functions of objects are defined and how inheritance is controlled.

An important feature of the object oriented paradigm provided in CLAP is division of an object's representation into two parts, *slots* and *relations.* The division is based on a distinction between information that is strictly local to an object (*ie.* will only be referenced through that object) and information that exists separately from the object. To illustrate this consider the following example:

```
Object(process_stream)
  ·flowrate :{value_class: real; range: [0,_]}
  ·component_fractions :{value_class: real_vector}

Object(process_unit)
  ·function :{value_class: string}
  ·capital_cost :{value_class: real; range: [0,_]}

SetRelation(inlet_stream↔{sink})
  ·domain_class: process_unit
  ·range_class: process_stream

SetRelation(outlet_stream↔{source})
  ·domain_class: process_unit
  ·range_class: process_stream
```

The definitions provide a representation for a process flowsheet. The first two items are object definitions listing the *slots* that are defined for that class of object. Following each slot name is a list of slot *meta properties.* Meta properties provide control over the methods that set and access that slot. The example shows two of the standard meta properties used: "value_class" which constrains the type of value that can be assigned to the slot and "range" which constrains the values that can be set. By customising the methods for accessing and setting the slots of an object class support for additional special meta properties can be provided.

The last two definitions in the example declare *relations* which are the primary mechanism used to link objects to each other. Those in the example are used to represent the flowsheet structure, for example the directive:

$assert_relation reactor inlet_stream feed_A.

will create a relation identifying feed_A as an input to the reactor. The latter object is referred to as the *domain* of the relation, the former as the *range* of the relation. As with slots, relations may also have meta properties. Two basic meta properties for relations are domain_class and range_class which control the object classes that a relation may be used with.

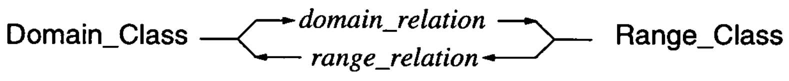
Accessing the first relation is possible in two ways either as the inlet_stream property of the reactor object or as the source property of the feed_A object. The latter is known as the relation inverse. The relation inverse could have been used in the previous assertion,

$assert_relation feed_A sink reactor.

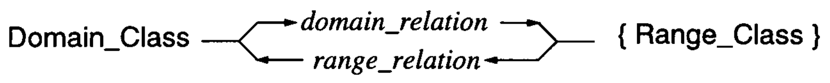
Support for inverse relations is a key benefit from using relations instead of slots to connect objects. The representation in the example above could have been managed using simple slots to hold the information but additional code would then have been required to maintain consistency between the cross-references. Slots are therefore used primarily for associating information with an object that requires no back referencing (*ie.* simple data types such strings and numbers, and objects that are only accessed via the referencing object).

There are three basic classes of relation, *viz*

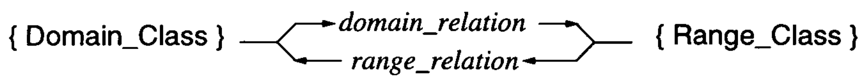
- Binary Relation: shown as domain_relation↔range_relation and



- Set Relation: shown as domain_relation↔{range_relation} and



- Dual Set Relation: shown as {domain_relation}↔{range_relation} and



The simplest form of relation is a *binary* relation which specifies a unique link between two objects. Setting a binary relation overrides any previous relation of the same name that was defined in either of the two objects. For the inlet_stream/sink relation used in the example above while this behaviour is appropriate from the perspective of the stream (*ie.* a stream can only have a single sink) it is not from the perspective of the process unit. The class of relation used instead is a *set relation* which associates a set of range object (*eg.* process streams) with a single domain object (*eg.* process unit). When asserting the inlet_stream/sink relation in the stream object any previously defined sink relation will be overwritten but in the process unit object the new relation will simply be added to the set of input_stream relations. Accessing the input_stream relation of the process unit would return a list of the input streams associated with it by that relation.

The most general form of relation is the *dual set relation* which may be multiply defined in both the domain object and range object. An alternate representation for a process flowsheet could use the dual set relation:

```
DSetRelation({inlet}↔{outlet})
 ·domain_class: process_unit
 ·range_class: process_unit
 ·stream
```

The connection between units is defined directly by this relation. If a stream is associated with the connection it is recorded as part of the relation. As instances of this relation are asserted each process unit accumulates a set of input units and a set of outlet units.

C.2 General Support for Hierarchical Design

An aspect that is common to the development of both the process and operating system is the use of hierarchical design. In this section we consider what general support can be provided for systems employing hierarchical development. One valuable facility a knowledge based system can provide is a convenient mechanism for keeping track of the hierarchical development. Two elements to hierarchical design have been identified:

- Design Refinement: the gradual resolution of detail using successively less abstract descriptions of the design. As part of the refinement process it is important to support the development of alternative solutions.

- Design Decomposition: the division of a design into simpler reduced design problems that are intended to be developed independently.

To provide common support for these operations across different design domains the representations developed have been based on the use of general relations (relations which have no restrictions on domain class or range class). The early experiments in supporting hierarchical design employed two relations, as illustrated in figure C.1.

The parts/part_of relation provides a link between the general design object and the more specific design objects that constitute it. A dual set relation is used since a particular part may be used in more than one design alternative. The refinements/refined_from relation is used to keep track of the design alternatives developed for each design object.

The example shows an initial design specification A for which two design alternatives have been proposed: $A.1$ which has parts B, C & D and $A.2$ which has parts E & F. For parts C and D further refinements have been made reflecting a

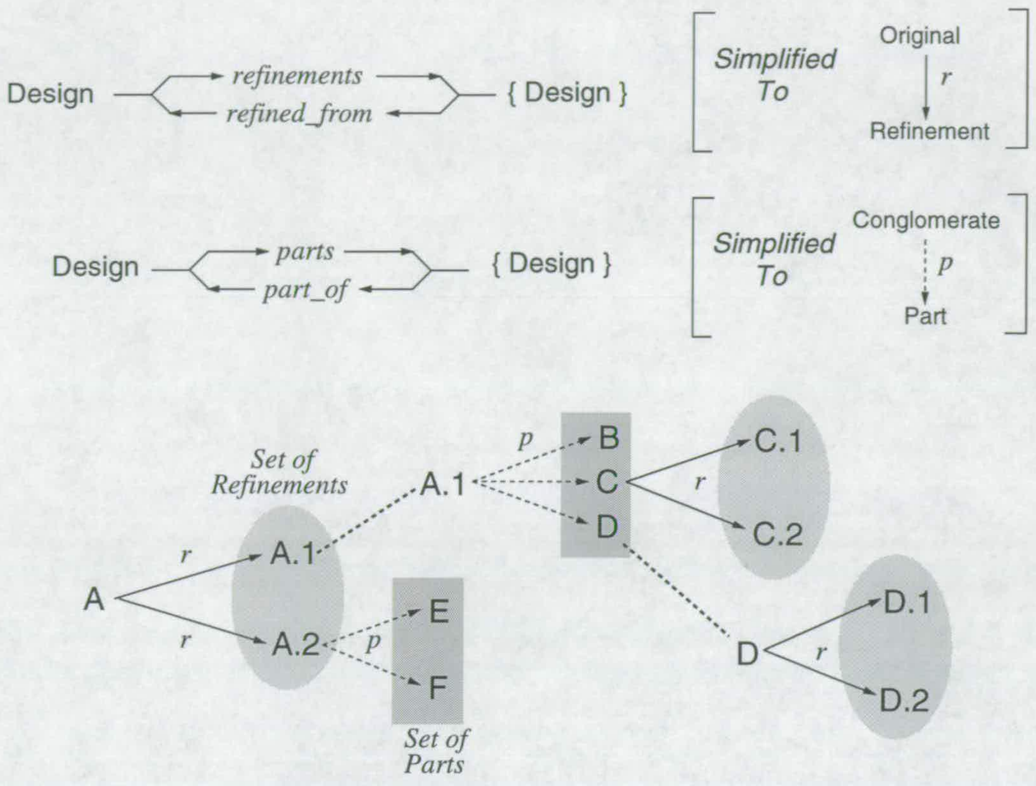


Figure C.1: A simple representation for design hierarchies

decomposition of the design into its separate parts. The design tree is thus built up with alternating branches of parts and refinements. Reconstructing a complete design requires a traversal of this tree selecting the alternative of interest on each branch.

The representation is sufficient provided decomposition of the design always splits the design into its individual parts. It is less satisfactory when the one to one correspondence between parts and decomposition does not hold. Typically this situation arises when there is strong interaction amongst a subset of the parts of a design and so must be developed together. For example, if parts B & C of design $A.1$ are strongly interdependent it is inappropriate to develop C independently without first resolving this interdependence.

To address the interaction between design parts the design tree could be reorganised, introducing an intermediate design object that groups together B & C.

Figure C.2 illustrates how this is done, the parts B & C being replaced in $A.1$ by

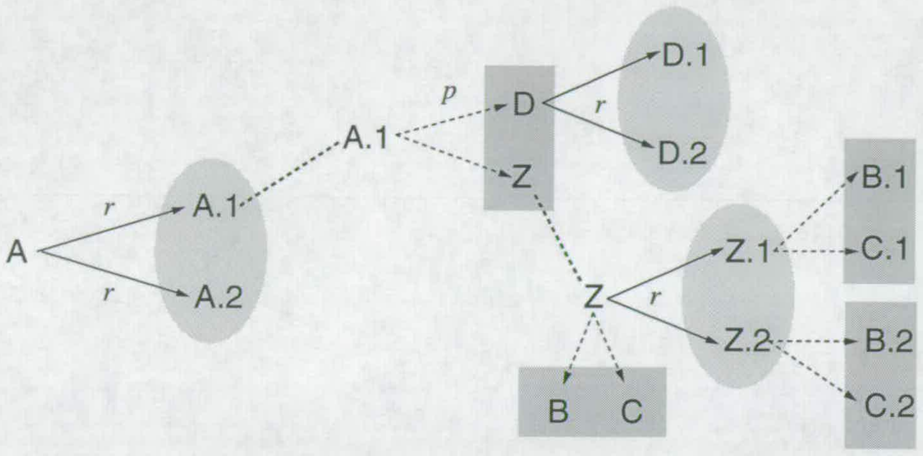


Figure C.2: Restructuring a design hierarchy

a composite design object Z. To keep a record of the original construction of $A.1$ the design object Z has B & C specified as its parts. Refinements however branch from Z not its parts, each alternative reflecting a different balance of the global design factors affecting B and C.

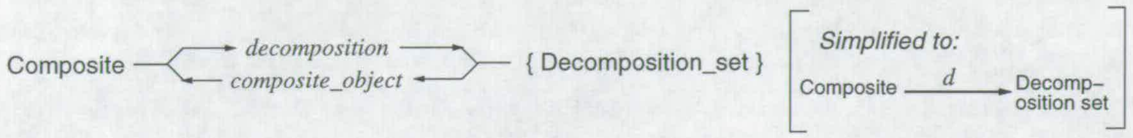
Supporting hierarchical design with the simple set of simple relations just discussed has some undesirable features. First it complicates inspection of the construction of the design object $A.1$ requiring a recursive search through the parts of each object referenced. Secondly retro-actively modifying a designs representation can create confusion causing problems in maintaining consistency in the design project model. Finally this arrangement makes it difficult to consider alternative decompositions.

Rather than relying on an implicit representation for design decomposition an alternative explicit representation has been developed. The revised approach instead of relying on the parts relation to double as a representation of design decomposition uses an additional decomposition_set object class and extra relations, *viz*

```
Object(decomposition_set)
 ·recomposition_guards

SetRelation(decomposition↔{composite_object})
 ·range_class: decomposition_set

DSetRelation({decomposed_objects}↔{part_of_decomposition})
 ·domain_class: decomposition_set
```



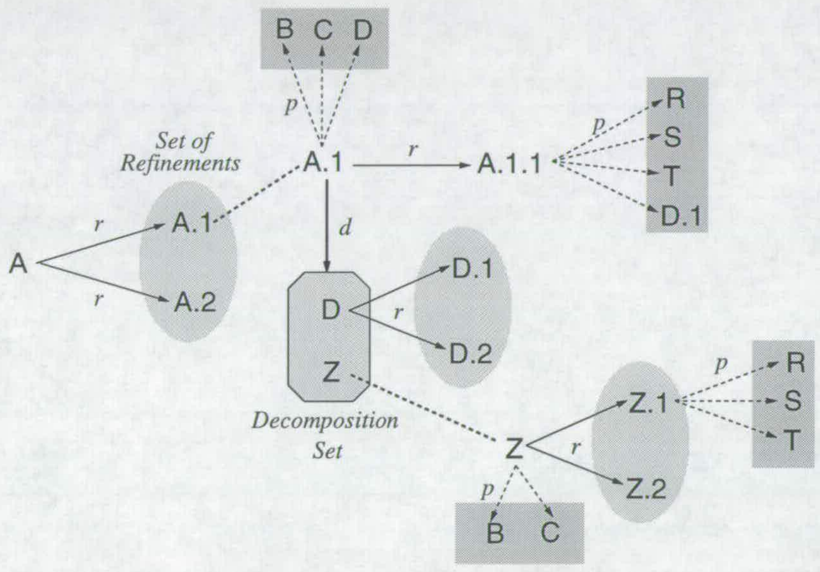


Figure C.3: Explicit representation of design decomposition

Figure C.3 shows how the relations are used to represent the design of the previous example. Design $A.1$ is shown explicitly to have parts B, C & D. It also has a single decomposition set associated with it which divides the design into two parts D and Z. The decomposed_objects/part_of_decomposition relation that connects these designs with the decomposition set is declared as a dual set relation. Using a dual set relation allows a particular reduced design to be used in multiple decompositions as well as possibly being part of the original design.

The use of an explicit representation for decomposition makes it possible to provide more control over design recomposition. Specifically the property recomposition_guards is used to define conditions that must be satisfied for a particular combination of refinements to be valid. For example in process design this may be used to check that streams that were common in the initial reduced designs are still consistent in the refined designs. The validated design resulting from recomposition is recorded as a refinement to the original design (*eg.* design $A.1.1$). This provides a mechanism for reviewing design decomposition, by first recombining the refinements of interest into a refinement of the original design and then developing new decompositions for the recomposed design. To keep track of the source of such design refinements the meta property origin is added to the refinement/refined_from relation which typically identifies the design tree traversal followed in recomposing the design.

C.3 Coordinating Process and Operating System Development

It is important that the development of the process and operating system follow compatible paths. In an ideal situation the decomposition of a design creates independent sub designs. Such perfect decompositions are rare in practice and attention has to be given to coordinating and directing the development of the separate design tasks. The decomposition of chemical plant design into process design and operating system design is a particular example of this. How a knowledge based system can aid in coordinating the development of process and operating system will now be considered

C.3.1 Using a Combined Design Class

One method for keeping the design of process and operating system linked together is to use a merged design hierarchy. For this the production_system class was defined to represent a combined process and operating system design,

```
Object(production_system)

Relation(process↔production_system)
 ·domain_class: production_system
 ·range_class: process

Relation(operating_system↔production_system)
 ·domain_class: production_system
 ·range_class: operating_system
```

The design hierarchy of the production_system substitutes for the separate hierarchies of the process and operating system. An example of the development of process and operating system using a combined hierarchy is illustrated in figure C.4.

The starting point of the example is a base design consisting of initial specifications for the process and operating system {*Pr1.0, OS1.0*}. The first stage of refinement of the production system arises from the development of alternative

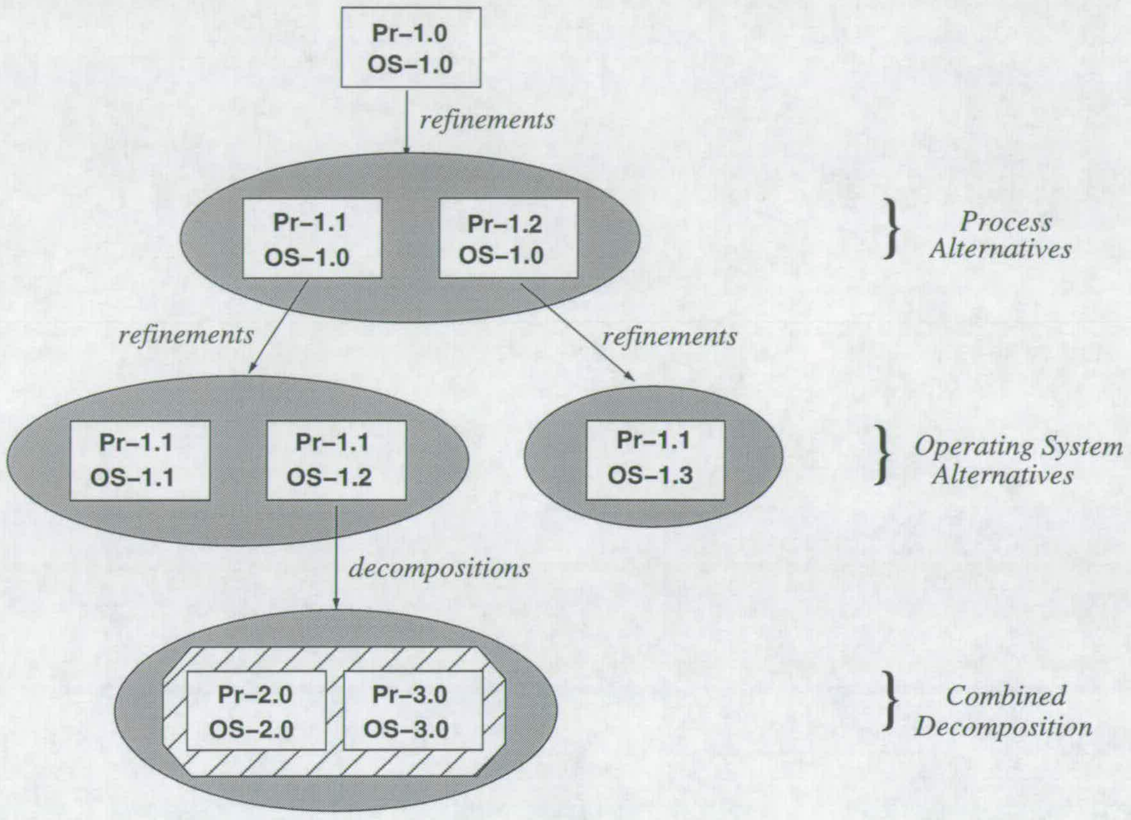


Figure C.4: A simple production system hierarchy

refinements to the base process design. Each process alternative is included in the hierarchy as a production system that uses the new process design with the same original operating system design. The first layer of refinement is shown in the figure C.4.

Having established some preliminary process designs the next step is to refine the operating system design for each production system. The refinements are included into the production system hierarchy as alternatives which employ the same process design but different operating systems. The new production systems form the second layer of refinements in the figure.

With balanced refinement of the process and operating system the production system hierarchy would continue to develop in this manner. Each layer of refinement corresponds alternately with a refinement of the process or the operating system. Eventually the complexity of either the process or operating system design will grow to the point that decomposition is necessary. In the production system hierarchy this is most simply managed by decomposing both process and operating system together, as in the last layer of figure C.4. Unfortunately there is not always a correspondence between the decompositions desired for process development and those for operating system development. A particular case of this is mode based decomposition of the operating system. In such situations while the design of the operating system is divided into a set of reduced operating tasks the process design is not normally decomposed at all.

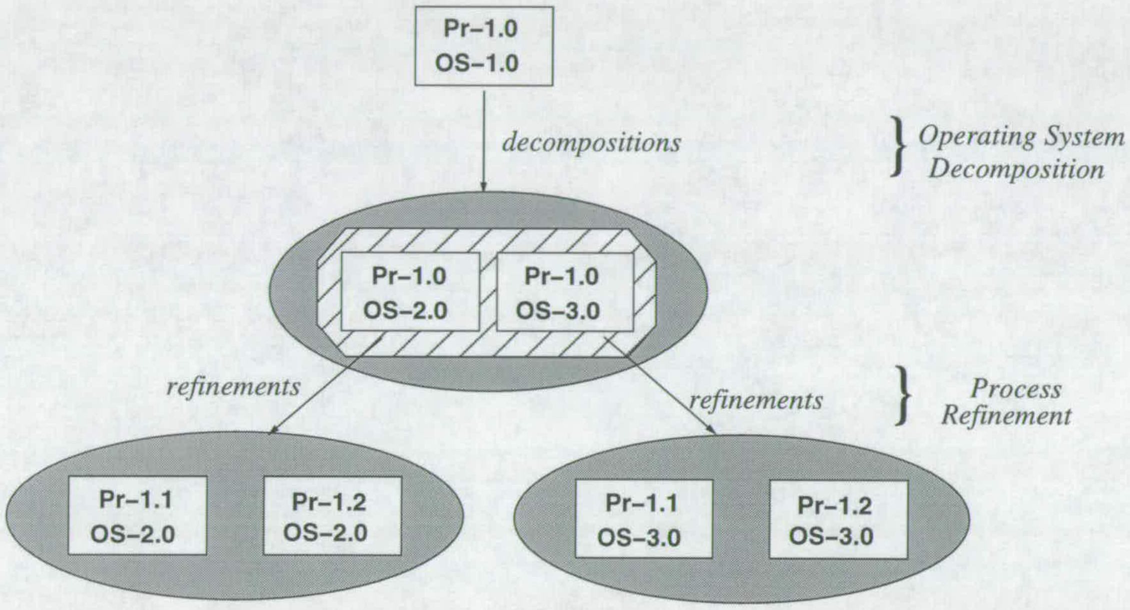


Figure C.5: Production system hierarchy after mode decomposition of the operating system

Figure C.5 illustrates how the production system hierarchy would be structured following mode decomposition of the operating system. In this case the mode decomposition has divided productions system design into two parts. Each part has a separate operating system design but they share a common process design.

Multiple referencing requires more attention to consistency maintenance. Also it can lead to excessive growths in the production system hierarchy. When a set of alternatives are developed for the process each alternative must be incorporated as a production system alternative in each branch of the production system hierarchy. In figure C.5 this is demonstrated with two alternatives to *Pr1.0* leading to the generation of four production system alternatives.

The difficulty of supporting independent decompositions of process and operating system is not the sole problem of the productions system representation. The process of production system development that has been discussed here is essentially a sequential one with alternating phases of process and operating system refinement. While offering improved opportunities for design integration it is likely to be at the cost of increased development time. In the traditional design organisation the process design would be completed straight through without waiting for feedback from control system designers. The development process the production system hierarchy is based upon implies that at each stage the process designers must wait for feedback from the operating system designers.

It would be valuable instead to be able to develop the process and operating system concurrently so that design integration could be achieved without excessive penalties on development times. However supporting such concurrent activity through the production system hierarchy is difficult since it couples tightly the process and operating system development.

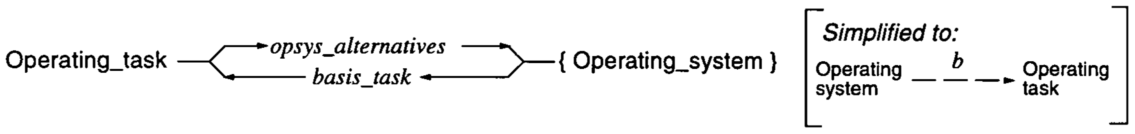
C.4 Decoupling the Process and Operating System Hierarchies

The experiments using a production system hierarchy, as discussed in the previous section, suffered some fundamental limitations. At the core of the approach was a tight coupling between process and operating system development. To gain a more flexible approach to design development, attention shifted to the support of distinct development hierarchies for the process and operating system. The development of the process and operating system are not completely independent, some basis for linking compatible designs and coordinating development is required. The revised representation uses the operating tasks as the common focus of development.

C.5 Operating Systems, Operating Tasks and System Models

Operating tasks define the management functions required from the operating system. The initial formulation of an operating task is based on knowledge of the process design and operating objectives. With the development of a control strategy for an operating task additional constraints on operation are elicited (*eg.* the requirement for a certain amount of flexibility). Additional requirements are recorded as part of the basis task for the control strategy. The basis task constitutes a definition of the conditions under which the corresponding operating system is applicable.

Operating tasks are developed as an additional design hierarchy alongside the process and operating system design hierarchies. The *basis task* relation links an operating system design to the operating task that defines its operating requirements.



A set relation is used to allow a set of alternative operating system designs to be connected with a given operating task.

The specification of an operating task has three key parts,

- Objective: *ie.* the cost function and temporal scope.

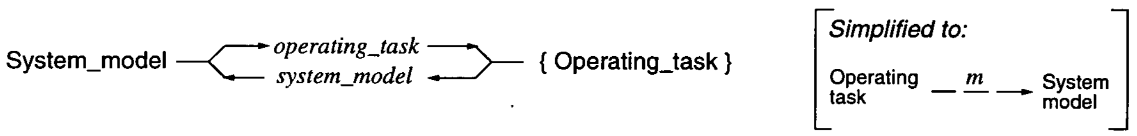
- Constraints: the physical laws relating variables and the physical limits on their range, and the failure conditions.

- Demands: the supervisory demands from the higher levels of control, disturbances sources from the environment.

Some of these elements depend primarily on the system being controlled and not on the mode of operation. For example the basic physical relations and constraints of a process are dependent on the design of the process not on how it is operated. Similarly external demands are determined by the connections between a process and its environment and are not altered by changing the operating objectives. The process related elements of an operating task specification are grouped together as a system_model. The resulting representation for an operating task is:

```
Object(operating_task)
 ·objective
 ·start_conditions
 ·end_conditions
 ·supervisory_demands
 ·supervisory_feedback

Object(system_model)
 ·physical_laws
 ·physical_constraints
 ·disturbance_sources
 ·failure_conditions
 ·controls
 ·measurements
```



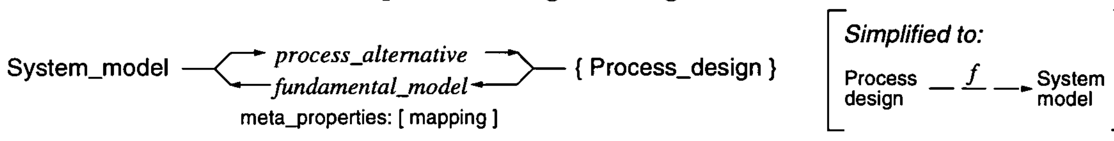
The interpretation of the contents of a system model depends on its context. Most of the slots in these objects are derived from the decomposition summarised above. The "controls" and "measurements" slots list the controls and measures available in the system. The "supervisory_feedback" slot in an operating task is used to specify information that is required as feedback to the supervisory layers.

C.6 Relating Process Designs with Operating Tasks

The system model is the key to relating process designs with operating tasks and so with alternative operating system designs. The basic condition for a process to be compatible with a given operating system design is that it is compatible with the basis operating task of the operating system. Compatibility is determined by the quality of the match between the system model and the real behaviour of the process. The key to relating operating systems with process designs is to establish a relation between system models and process designs.

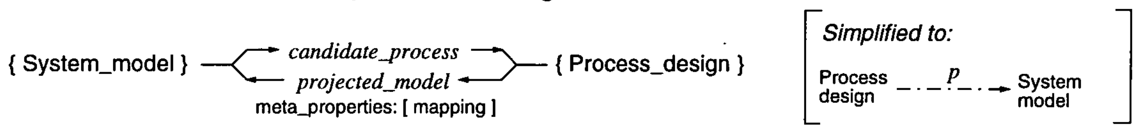
A process design can be modelled to varying degrees of detail and so is capable of satisfying a range of system models. Associating all possible models of a process through a single relation for that process poorly structures the information. Instead to distinguish between different types of model a set of system model relations is proposed.

The first type of system model we identify for the process is the 'fundamental' model. The fundamental model is the most complete model that can be made of the process without making assumptions about unresolved design detail. The model is associated with a process design through the relation:



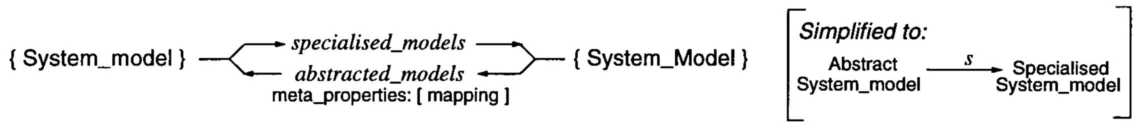
A set relation is used so that processes with similar behaviour may utilise a common model. To further assist the use of a common model a meta property mapping is used which record the correspondence between the variables of the model and properties of the process. The fundamental model is the foundation from which other are derived but in general is itself insufficient for most forms of analysis and design. For example to perform a preliminary costing it is necessary to employ empirical cost functions for various parts of the design. Such functions are approximations and only accurate for a restricted set of the possible design refinements. As such they cannot be considered as part of the fundamental process model.

The models of a process which involve some dependency on as yet undesigned details are referred to as *projected models.* A process may have a set of projected models connected to it by the following relation:



The mapping property again is used to record the correspondence between variables of the model and properties of the process. A dual set relation is used to support both the use of a common projected model for multiple designs and multiple projected models for a single design.

Projected models and fundamental models are also related through an additional relation:



The relation is used to provide a link between a model, its abstractions and specialisations. Projected models are recorded through this relation as a specialised model of the fundamental model. The mapping property in this case is used for recording the equivalence between parts of the two models and any approximations made in establishing that equivalence. The simple refinement/refined_from relation used in the design hierarchies is not used here because it is only a set relation and it was desirable to be able to support multiple abstractions as well as specialisations.

To illustrate the use of these relations consider a simple process described by a single block (see figure C.6). If little is defined for the block other than its inputs and outputs then its fundamental model is restricted to mass conservation constraints for the block (*SM1.0* in the figure). However based on the expected design

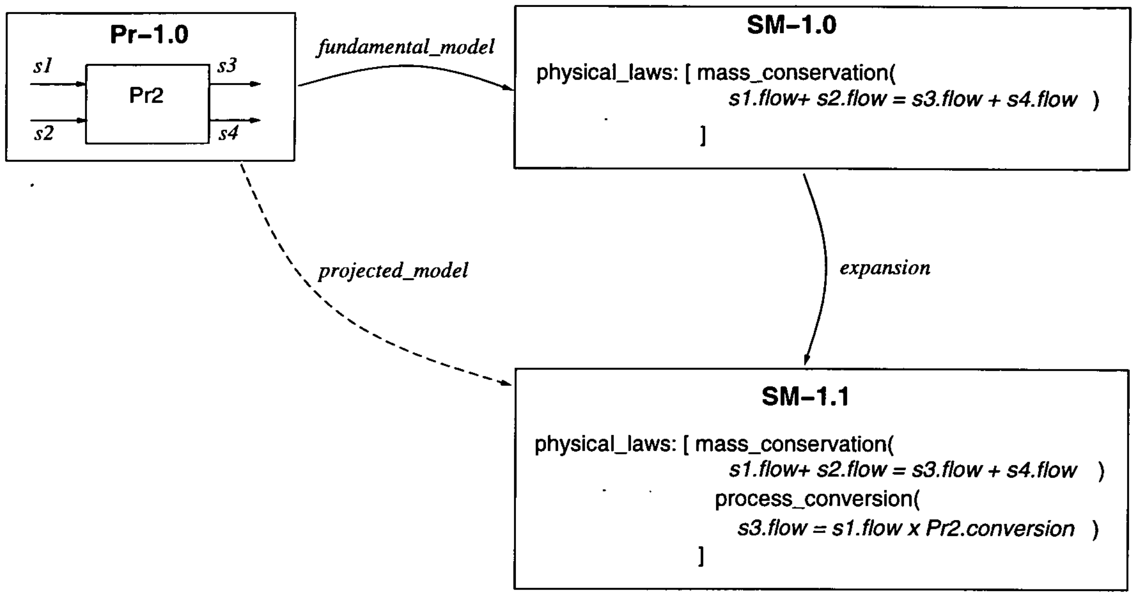

Pr-1.0
s1
s2
s3
s4
Pr2
fundamental_model
SM-1.0
physical_laws: [mass_conservation(s1.flow+ s2.flow = s3.flow + s4.flow)]
expansion
projected_model
SM-1.1
physical_laws: [mass_conservation(s1.flow+ s2.flow = s3.flow + s4.flow) process_conversion(s3.flow = s1.flow x Pr2.conversion)]

Figure C.6: The relations between a process design and its models

for the block it is possible to provide additional constraints on the relationships between the input and outputs. These additional constraints are combined with the fundamental model to form a new model (*SM1.1* in the figure). The model is related to the process design as a projected model and to the original model as a specialisation.

In producing a process refinement it is necessary also to develop a corresponding refinement to the fundamental model. The first step is to find the model that most closely maps to the fundamental model of the new design. The model should be selected from among fundamental and projected models of the original design. In the case of an exact match it is only necessary to add the relation identifying the model as the fundamental model of the new design. If the closest match is one of the projected models the new fundamental model is formed by specialisation of this model and appropriately recorded. Such a procedure helps to reduce model replication and keep the inter relations between models up to date. The closest match may be the previous fundamental model in which case some of the original projected models may also be specialised for the new design. The result of this is illustrated in figure C.7.

$$\begin{tikzcd}
\textbf{Pr-1.0} \arrow[rr, "f"] \arrow[dd] \arrow[dr, dashed, "p"'] & & \textbf{SM-1.0} \arrow[dl, "e"'] \arrow[dd, "e"] \\
& \textbf{SM-1.1} \arrow[dd, "e"] & \\
\textbf{Pr-1.1} \arrow[rr, "f"] \arrow[dr, dashed, "p"'] & & \textbf{SM-1.2} \arrow[dl, "e"] \\
& \textbf{SM-1.2.1} &
\end{tikzcd}$$

Figure C.7: Refinement of a process and it models

C.7 Refinement of the Process and Operating System

The relations discussed above provide a basis for relating compatible process and operating system designs via their association with a common operating task and system model. Coordinating refinement of process and operating system using these relations is now considered.

The first step is to establish the foundation objects. The contents of these objects will be minimal, the key aim being to establish the base objects for the various hierarchies and the relation between them. The initial design state is illustrated in figure C.8. The foundation process design (*Pr1.0*) will typically only identify the basic input/output structure of the process design. The associated fundamental model (*SM1.0*) will correspondingly be equally simple. As well as being the fundamental model of the process the model is also used as the system model for the foundation operating task (*OT1.0*) which provides a basic definition

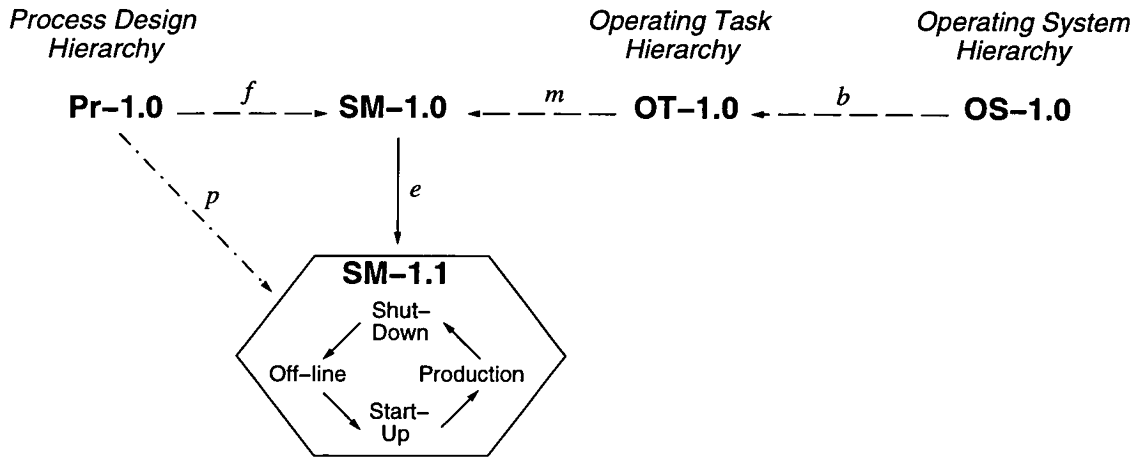


Figure C.8: Initial state of process and operating system hierarchies

of the operation goals and is the basis task of the foundation operating system (*OS1.0*).

The initial focus for design development is the definition of a mode oriented description of operation. For this example a basic mode decomposition is proposed (*ie.* off-line/start-up/production/shutdown). The decomposition is recorded as a projected model of the foundation process design and as a specialised instance of the foundation system model (shown as *SM1.1* in the previous figure).

The simplest approach to development of the operating system is to use this projected model as the basis for refinement. The operating task is first refined using the new system model. The revised operating task is used to develop a preliminary operating system design. If the resulting operating system design requires no further operability conditions than those available from the projected model then the new operating task will also be the basis task for the refined operating system design (as illustrated in figure C.9). Alternatively the resulting operating system design may refine the operability requirements. The additional constraints are incorporated into a further refinement of the operating task and system model. The result is the design state shown in figure C.10.

The projected model may be incompatible with that preferred for operating

$$\begin{tikzcd}
\textit{Process Design Hierarchy} & & \textit{Operating Task Hierarchy} & \textit{Operating System Hierarchy} \\
\textbf{Pr-1.0} \arrow[r, dashed, "f"] \arrow[dr, dashed, "p"'] & \textbf{SM-1.0} \arrow[d, "e"] & \textbf{OT-1.0} \arrow[l, dashed, "m"'] \arrow[d, "r"] & \textbf{OS-1.0} \arrow[l, dashed, "b"'] \arrow[d, "r"] \\
& \textbf{SM-1.1} & \textbf{OT-1.1} \arrow[l, dashed, "m"'] & \textbf{OS-1.0} \arrow[l, dashed, "b"']
\end{tikzcd}$$

Figure C.9: Operating system refinement using projected process model

$$\begin{tikzcd}
\textit{Process Design Hierarchy} & & \textit{Operating Task Hierarchy} & \textit{Operating System Hierarchy} \\
\textbf{Pr-1.0} \arrow[r, dashed, "f"] \arrow[dr, dashed, "p"'] & \textbf{SM-1.0} \arrow[d, "e"] & \textbf{OT-1.0} \arrow[l, dashed, "m"'] \arrow[d, "r"] & \textbf{OS-1.0} \arrow[l, dashed, "b"'] \arrow[dd, "r"] \\
& \textbf{SM-1.1} \arrow[d, "e"] & \textbf{OT-1.1} \arrow[l, dashed, "m"'] \arrow[d, "r"] & \\
& \textbf{SM-1.1.1} & \textbf{OT-1.1.1} \arrow[l, dashed, "m"'] & \textbf{OS-1.0} \arrow[l, dashed, "b"']
\end{tikzcd}$$

Figure C.10: Refinement introducing stricter operability requirements

system design. For example in section 3.6 the preferred operating system design required an additional intermediate 'secure' state with associated regulatory and transition modes. The basis task and associated model for the extended decomposition are represented as alternative refinements to the root task and model. The resulting state of the design space is shown in figure C.11.

In the case of the latter two scenarios, new system models have been introduced either as a specialisation or as an alternative to the original projected model. The projected model provided the operating system designers with an indication of the likely behaviour of future process designs from the process designers perspective. These new models provide the process designer with an indication of the behaviour desirable for future designs from the operations perspective. The revised system

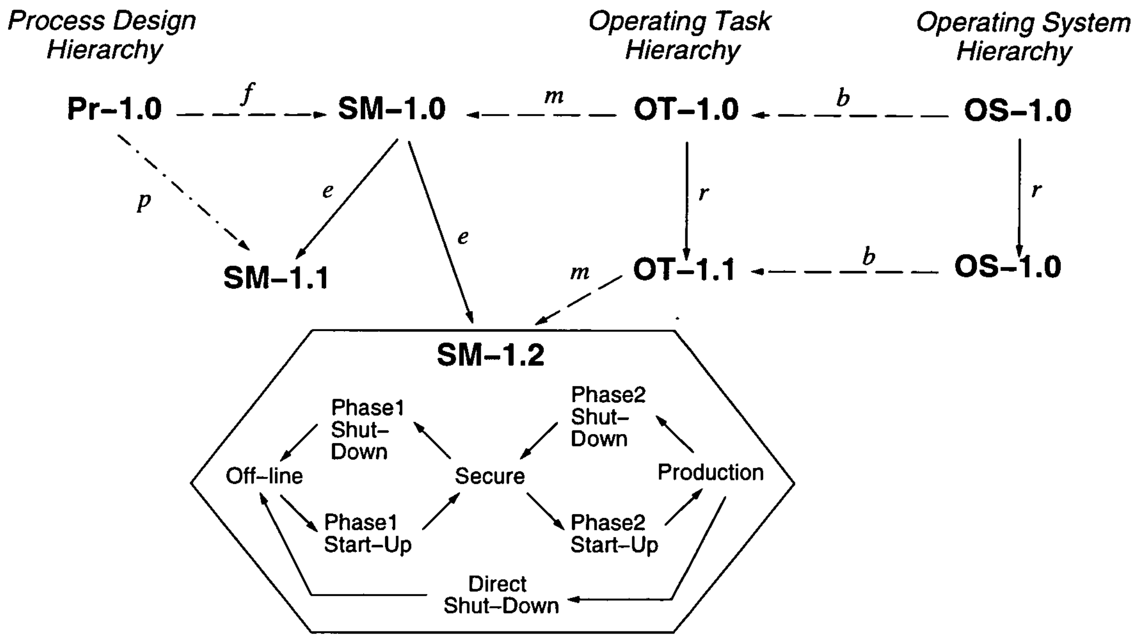


Figure C.11: Refinement introducing a separate system model

models provide direction for the next phase of process design which will generate further refinements of the system models. The system model is used to describe the preferences of the process and operating system designers on a common basis.

Refinement is one part of hierarchical design and this example demonstrates how the refinement of the process and operating system design can be linked. Before discussing how decomposition is managed it is necessary to consider in more detail how the operating system design is represented.

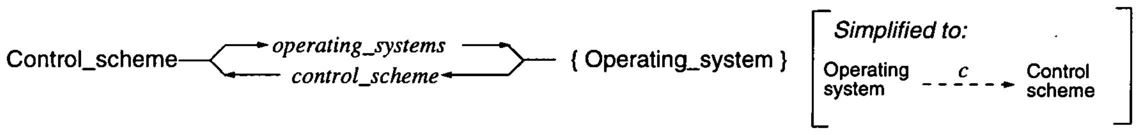
C.8 Representing the Design of an Operating System

As has now been described the design of an operating system starts by defining the required functionality as an operating task. Meeting the specifications of this task the designer has two basic options:

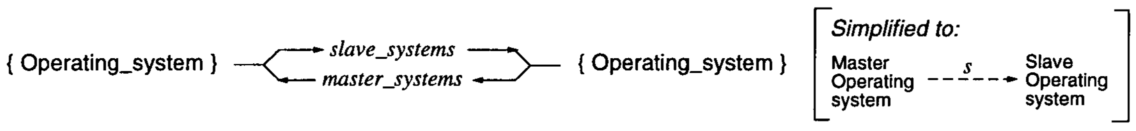
Either: Construct a set of algorithms to solve the operating task as a single problem.

Or: Decompose the operating task into a set of simplified tasks for which simpler sets of algorithms can be devised.

For the former case the set of algorithms are represented as a control_scheme, which is associated with the operating system design by the relation:



In the latter case the operating system design consists of a supervisory control scheme and a set of slave operating systems. The supervisory control scheme for an operating system design is identified by the relation above. The set of slave operating systems are associated with the design through the relation:



The slave operating systems may simply be single control schemes or could further decompose the task extending the hierarchy of operating systems and control schemes. The slave system relation is managed as a dual set relation to support the use of a slave operating system in more than one design and also at more than one point in the hierarchy for a single design. The latter case arises in particular with very specific task (*ie.* single loop controls) that are common to most modes of operation.

The structure of a control scheme is based upon the decomposition discussed in section 3.7. Four basic functions were identified:

- Optimisation
- Adaptation

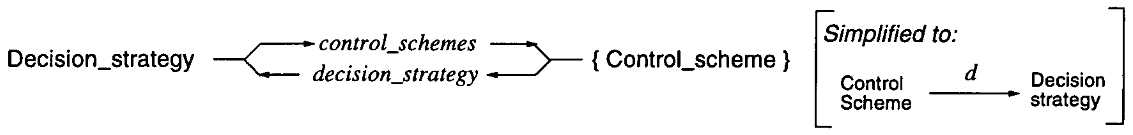
- Control Distribution

- Identification

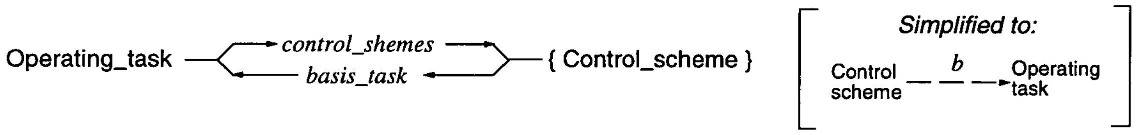
The latter two of these are intended to provide an interface between the abstract model employed by the adaptation and optimisation elements and the 'reality' of the underlying system. The primary intent in making this distinction is to encourage design clarity, keeping explicit the models that the design is founded on. The division of function is also reflected in the representation of a control scheme:

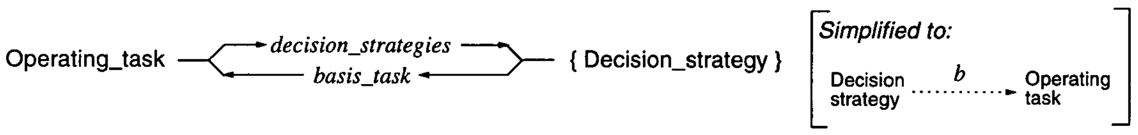
Object(control_scheme)
·control_distribution_algorithm
·identification_algorithm

Object(decision_strategy)
optimisation_algorithm adaptation_algorithm



Both of these objects use the operating task representation to record their design basis via the "basis_task" relation:





A decision_strategy describes how control actions are determined and the associated basis task defines the modelling abstraction used in formulating the algorithms. A control_scheme can be viewed as a shell around the decision_strategy which provides an interface between its internal model and the 'reality' described by the basis task of the control scheme. With this division in the design representation it is possible to make use of a general decision strategy in multiple control schemes without the need to replicate the design of the decision strategy.

For example, in the preliminary design stages decision strategies are derived without the need for greater abstraction than already exists. The operating system, control scheme, and decision strategy all share a common basis task (figure C.12). Following the next phase of process design the detail of the original

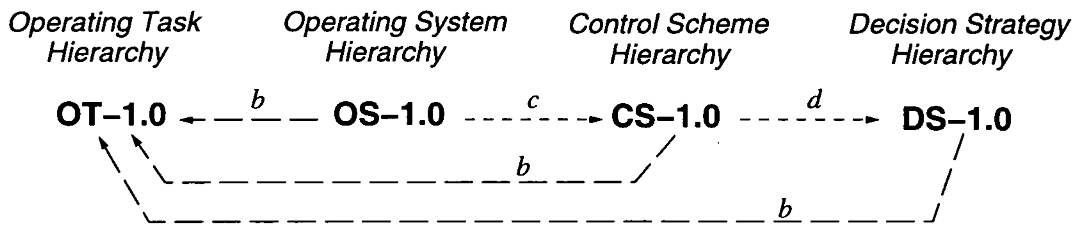


Figure C.12: Relationships for a non-abstracted design

task is refined. If the refinement of the operating task only affects the inputs and outputs that are available then the operating system design only requires refinement of the control distribution and identification algorithms. The decision strategy then remains the same (figure C.13). If, as is more likely, additional deci-

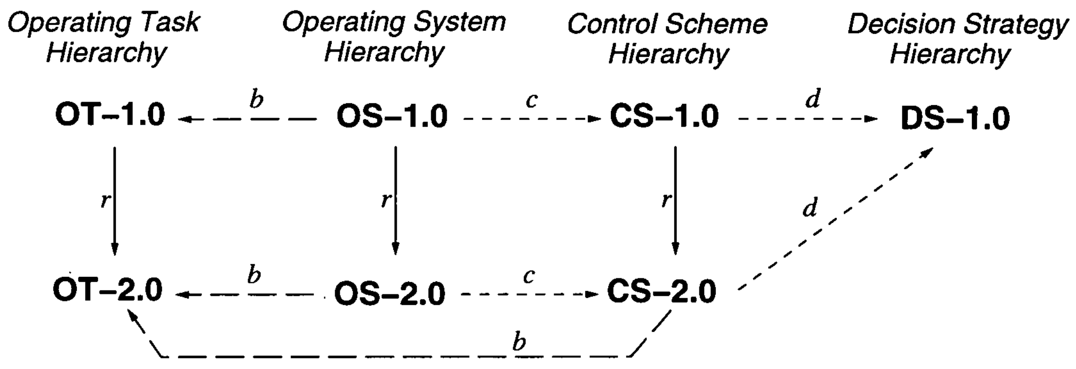


Figure C.13: Simple re-use of a decision strategy

sion variables and demands are introduced the decision strategy on its own will be insufficient. If the original task was formulated well it identified the key decision factors for optimisation of process operation. The additional factors introduced only refine the general optimum in which case the original control scheme design may be preserved and used as part of the hierarchical strategy to meeting the goal task.

Supporting a Library of Standard Schemes

The approach discussed here provides a convenient basis for organising and accessing a library of decision strategies. Many theoretical algorithms are based on an abstract model of an operating task that is to be solved. With the approach discussed here, these theoretical tasks are represented explicitly. The suitability of a theoretical algorithm is determined by how well the goal task maps to the theoretical task. Using the algorithm in a control scheme simply requires the implementation of that mapping, there is no need to re-invent the algorithm. For general control algorithms (*eg.* PID control) there may be a set of decision strategies which reflect the different adaptation or 'tuning' strategies that can be adopted for the control algorithm dependent upon the nature of the operating task. Equally a library of decision strategies could describe alternative column control configurations, where the detail of the associated operating tasks defines the column conditions and behaviour for which they are best suited.

C.9 Combined Process and Operating System Decomposition

Decomposition of the process design is the most straight forward aspect of developing a combined decomposition. The decomposition_set representation discussed earlier is used (see figure C.14). For each of the reduced process designs a fundamental model will be required. These are generated by decomposition of the fundamental model of the original process.

The decomposition of a system model however is not simply a case of partitioning its parts appropriately. In breaking a model into parts, cause and effect links that existed between the parts can be lost. If the information about these links is not preserved it will appear as if the decomposed system has extra degrees

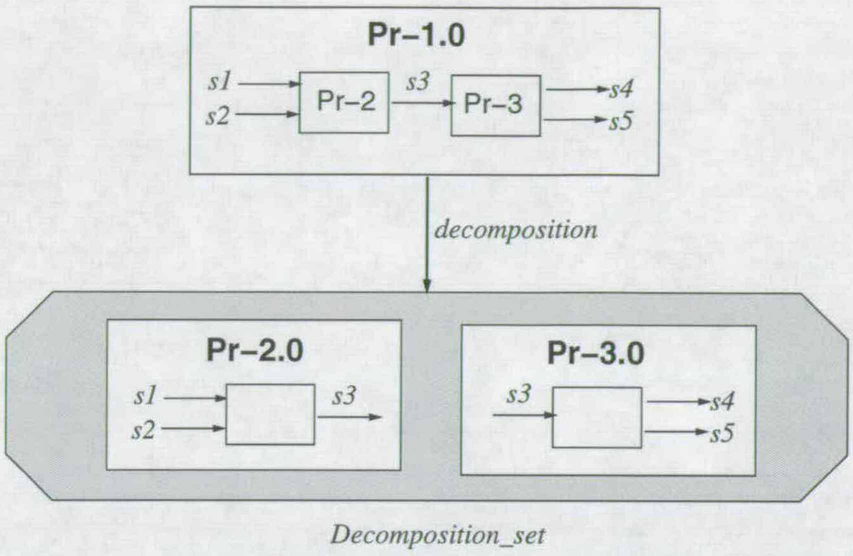


Figure C.14: Decomposition of the process

of freedom. To preserve such information in the decomposed models *peer demands* are associated with the shared variables. A peer_demand is treated as a special class of demand used in the representation of system_models. It is a combination of two other basic demand types disturbance_source and failure_source. At the source of a broken cause and effect link it is used as a failure_source, at the sink as a disturbance_source.

To illustrate, consider the case of a decomposition which splits a stream connecting two unit models. The connection links the outputs of the upstream unit model with the inputs of the downstream unit model. When decomposed these inputs and outputs appear as free variables unless additional constraints are added. The constraint is provided through the use of a peer demand. In the downstream model it is added as a disturbance source associated with the input. The same peer demand is added to the upstream model as a failure source associated with the output.

The description of the peer demand is used in different ways in each system model. In the downstream model it is used as a model of how the input is expected to vary. In the upstream model its interpretation is as a limit on the variations permitted for the output. For the fundamental model the information that can be included in the peer demand is restricted. However, as part of a projected model, the peer demands provide a mechanism for specifying the extent to which disturbances can be passed on to other parts of the design. Recording the decomposition of the fundamental model again uses the basic decomposition set representation figure C.15.

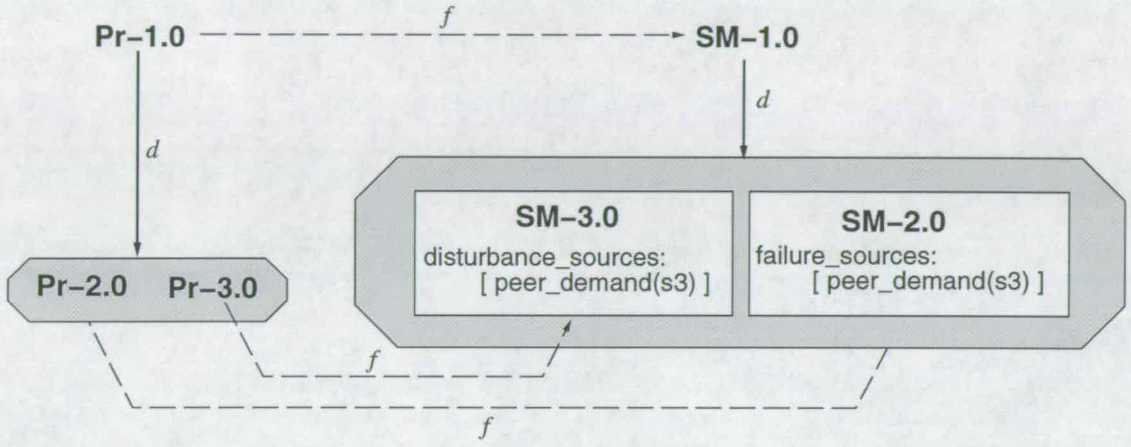


Figure C.15: Decomposition of the fundamental model

To follow the same decomposition in the development of an operating system design implies the use of a hierarchical operating system design. The hierarchy is represented by a supervisory control scheme and a set of slave operating systems. The operating task definitions for these are derived from a special form of decomposition set used for the operating task hierarchy:

```
Object(task_decomposition)
  ·class_of: decomposition_set
  ·supervisory_task
  ·slave_tasks
```

A slave task is created for each part of the decomposed system model. The model decomposition however does not provide a basis model for the supervisory operating task. Nor does it define the supervisory_demands or supervisory_feedback for the slave tasks. To complete the task decomposition it is necessary to formulate the supervisory task which uses an abstraction of the full system model. One option is to utilise the tasks and models developed during earlier phases of the design. If these have been well formulated for hierarchical development they will pick out the factors of global importance. Using a task from the preliminary design as the abstraction for a supervisory task has the advantage that design effort put into developing the preliminary control schemes is not wasted.

Once the formulation of the supervisory task is complete the final details of the slave tasks can be completed. First the supervisory_demands and supervisory_feedback of the slave tasks can be derived from the controls and measurements employed in the supervisory task model. In addition to identify in the slave tasks what demands are managed at a superior level the slots supervised_disturbances and supervised_failures are added to the operating task representation.

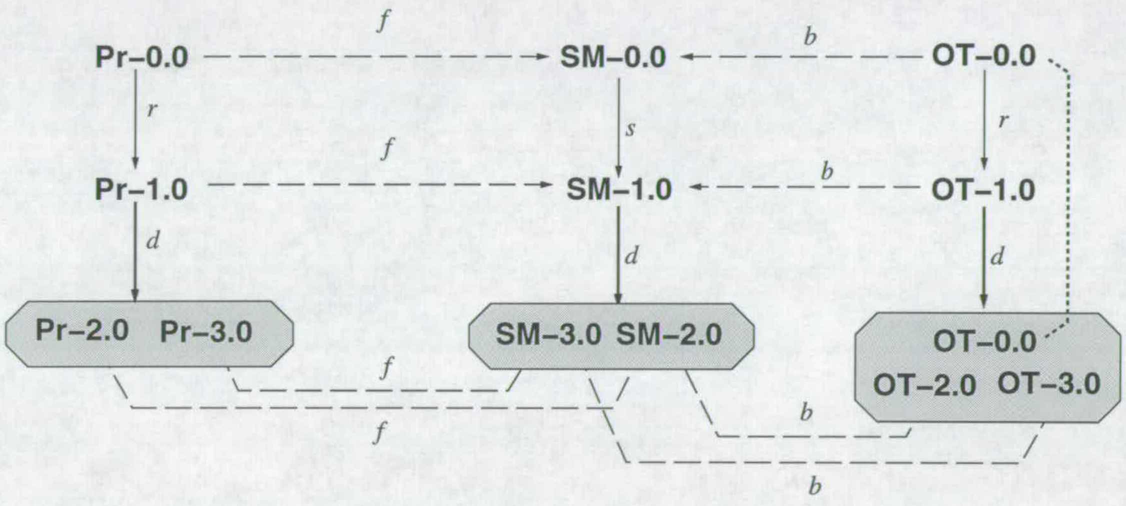


Figure C.16: Decomposition of the goal operating task

Figure C.16 shows the relations that result from a decomposition that uses an early design abstraction as the model for the supervisory task. For a single process and system model decomposition several possible decompositions of the operating task will exist.

Having established the task decomposition, development of an operating system based on this decomposition has two parts.

- Formulation of the Supervisory Control Scheme: If the task abstraction used corresponds to a previous task specification then this could use any of the corresponding preliminary control schemes.

- Evolution of the Slave Operation System Designs: The slave operating systems can be refined in step with the design of the corresponding process sections following the same procedure as has already been discussed.

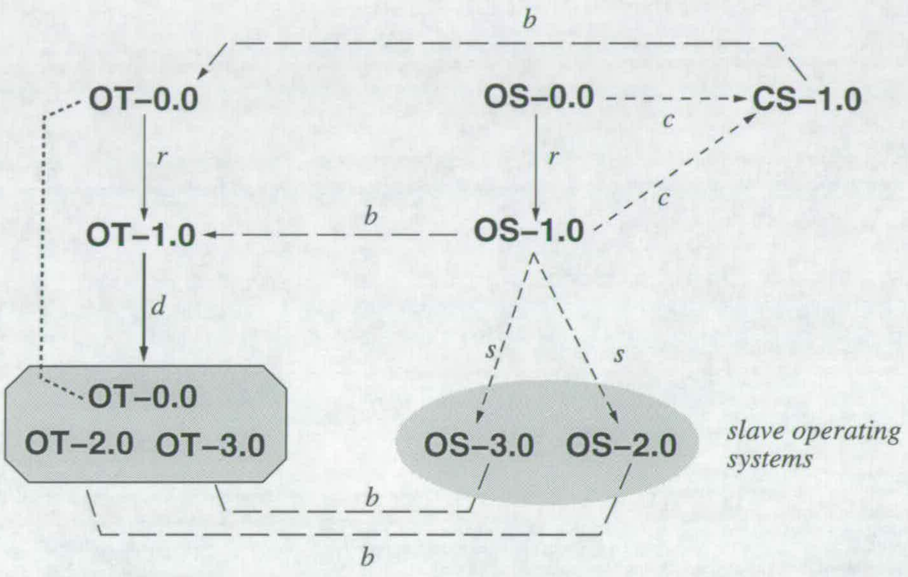


Figure C.17: Decomposition reusing preliminary design as the supervisory scheme

Figure C.17 shows the initial relations for a decomposed operating system design. The design of the supervisory control scheme is only reviewed when the alternatives developed from the decomposition are recombined into specific refinements of the full operating system design.

C.10 Decomposition of the Operating System Alone

One problem indicated with the production_system representation was maintaining different decomposition paths for the process and operating system designs. A situation where this can arise is mode oriented decomposition of the operating system. In such cases there is rarely any equivalent decomposition of the process. For example, the decomposition shown in figure C.18 has been developed not as the result of a process decomposition but from independent decomposition of the operating system. Process development is on the whole process and as a result process refinements update the fundamental model of the whole process not the decomposed parts used by the operating system. To map the new information to the models used by the operating system the decomposition must be reapplied to the refined system model (see figure C.18).

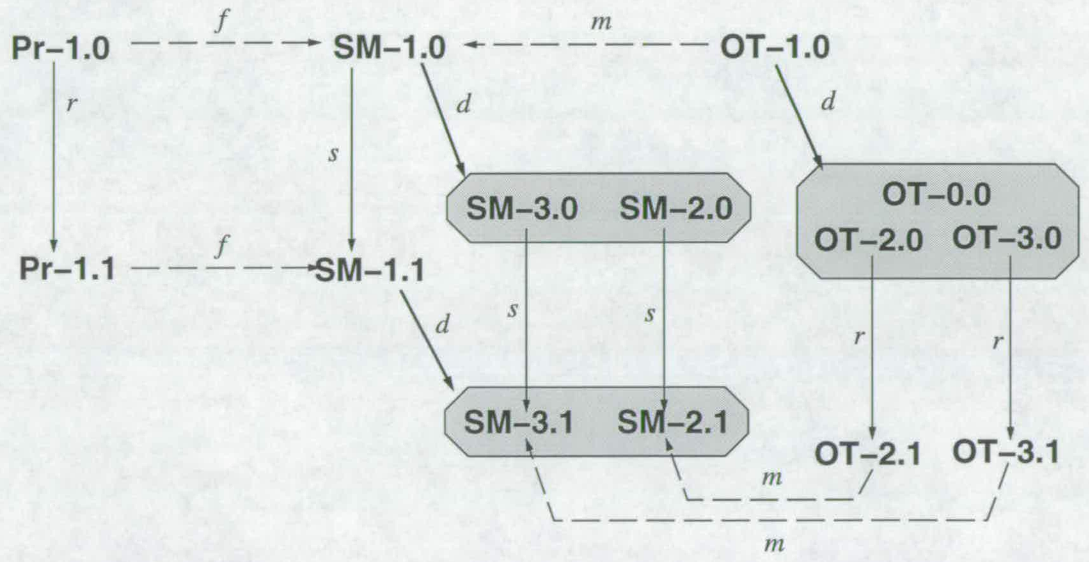


Figure C.18: Global refinement of a decomposed operating task

Feeding back information to the process designer is more complex. The development of the operating system is focussed on the decomposed tasks not the global design. Specialised requirements of the slave operating systems are added as refinements to the decomposed operating tasks but this has no direct feedback to the global models employed for the process design (figure C.19).

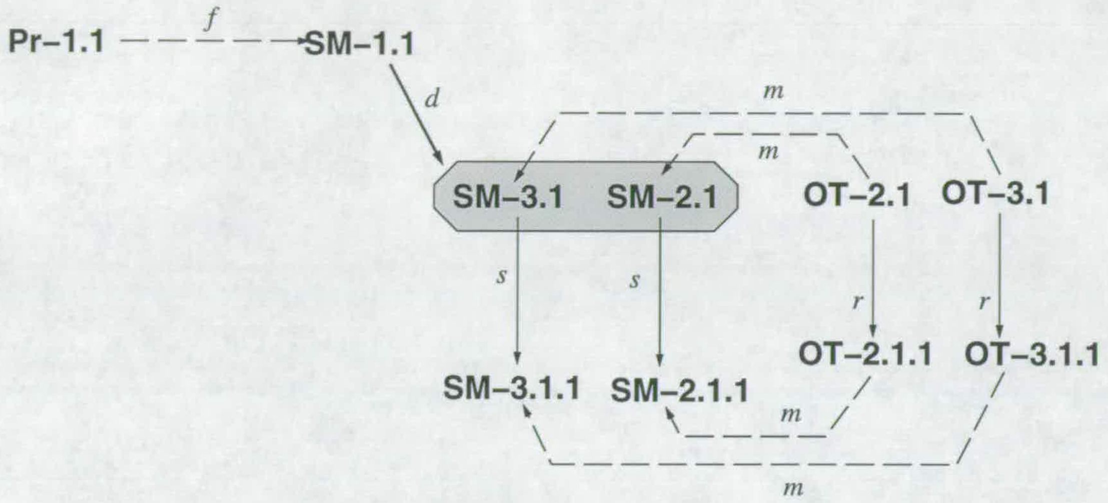


Figure C.19: Specialisation of decomposed operating tasks

For the process designers to discover what the special operating system requirements are it is necessary to inspect not only any refinements of the fundamental model but also refinement to decompositions of this model. If there are many levels of operating system decomposition this is an inconvenient process. Instead the process designers may request the operating system designers to collate the alternatives in the decomposition into a set of complete models reflecting probable design requirements. The result is the introduction of refinements to the global models and decomposition sets grouping together the design parts (figure C.20).

If the process and operating system design follow different decompositions this problem is further compounded. To compare requirements and capabilities composite models must be collated for both the process and operating system design. Where there is a significant difference in the decompositions the implication is that there is a mismatch in the priorities assigned to demands and decisions. Under

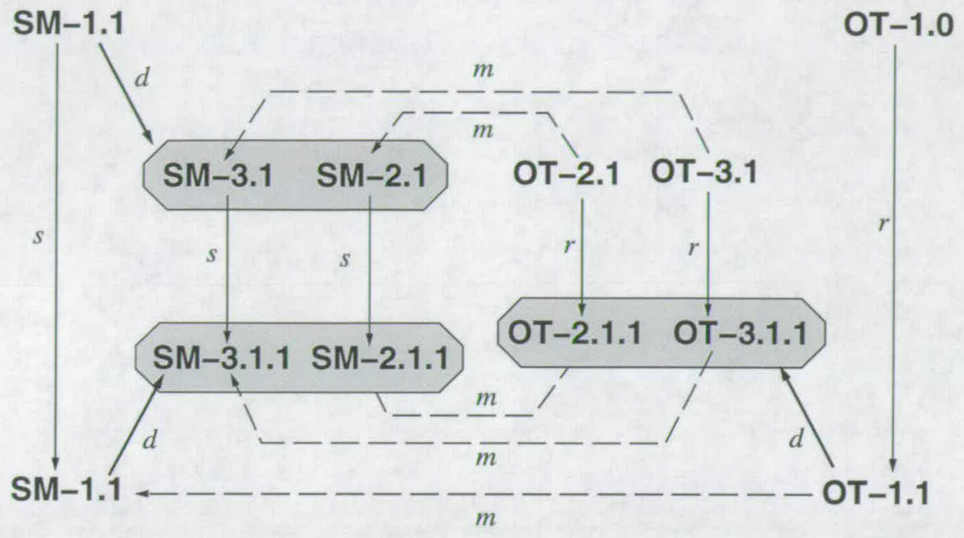


Figure C.20: Recomposing the refinement to a task decomposition

such circumstances it would be wiser to review the decomposition strategy and unify on a common approach.

C.11 Summary

The representations that have been developed are founded on a novel object oriented paradigm with a strong emphasis on the use of relations to connect objects. At the core of the representation scheme is the use of operating tasks and system models to link concurrently developing process and operating system designs. A basic set of relations have been proposed to organise the system models developed during design but this relies on the user identifying and connecting related models. Support for developing process models using the CLAP language has been considered more completely by Hutton [64].

Appendix D

Glossary and Nomenclature

D.1 Definitions

Control scheme:	The set of algorithms used to implement an operating task.
Dynamic resilience:	The impact of process design on the regulatory control of a process.
Executive task:	An operating task to coordinate alarm response and select the active operating tasks.
External demand:	Causes of disturbance to the process.
Flexibility:	Steady state test of the feasibility of a process for the expected variations in uncertain parameters.
Operating mode:	A distinct phase of process operation.
Operating system:	A unified system for control and operations management.
Operating task:	The specification of a control or management function required of the operating system.
Regulatory task:	An operating task for steady state control and optimisation.
Supervisory demands:	Operating target set by supervising operating tasks.
Transition task:	An operating task for control of a process through switches in operating mode.
Robustness:	Insensitivity of control to model uncertainty.

D.2 Abbreviations:

HEN:	Heat exchanger network.
IAE:	Integral of absolute error.
IE:	Integral of error.
IER:	Integral error ratio.
IMC:	The internalmodel control structure.
ISE:	Integral of the square of the error.
MIMO:	Multiple input multiple output.
NMP:	Non-minimum phase.
RDG:	Relative disturbance gain.
RGA:	Relative gain array.
SISO:	Single input single output.
SSV:	Structured singular value.

D.3 Nomenclature for Operability Review

α	error projection of a model
ω	frequency
σ	singular value
σ_m	minimum singular value
σ_M	maximum singular value
$\sigma_{M,i}$	i'th largest singular value
C	cost function
D	disturbance sensitivity
E_p	expected value function for uncertain parameters p
F	feedback filter
G_c	control system transfer matrix
G	real process transfer matrix
G_-	invertible part of process transfer matrix

G_+	non-invertible part of process transfer matrix.
H	setpoint sensitivity
L	uncertaint matrix
L_A	Additive uncertainty
L_I	Input multiplicative uncertainty
L_O	Output multiplicative uncertainty
$\tilde{G}$	process model transfer matrix
d	(flexibility) vector of design variables
d	(dynamic resilience) disturbance vector
$\tilde{d}$	estimate of disturbance vector
$f()$	the susbstitution of equalities $h()$ into inequalites $g()$ to eliminate state variables.
$g()$	vector of process inequalities
$h()$	vector of process equalities
$l(\omega)$	frequency domain bound on uncertainty
p	vector of uncertain parameters
r	setpoint input vector
u	vector of control variables
x	vector of state variables
y	process output vector

D.4 Nomenclature for Case Study

A	molar ratio of H_2SO_4:CaF_2at reactor inlet
C_{HF}	sales price for HF
C_{CaF2}	purchase price for CaF_2
$C_{p,W}$	heat capacity of reactor wall
$C_{R,F}$	annualised fixed costs for the reactor
$C_{R,O}$	annual operating costs for the reactor
C_S	annualised separator costs
C_{SO4}	purchase price for H_2SO_4
F_{HF}	annual production rate of HF
K_F	pipe flow constant
M_K	mass of reactor wall

M_R	mass of reactor contents
P_F	downstream pressure created by vent fan
P_R	reactor pressure
P_S	separation system pressure
R_{SO4}	fractional recovery of H_2SO_4in the separation section
S_{max}	maximum sales rate
S_{min}	minimum sales rate
S_{nom}	nominal sales rate
T_D	time for which a peak deviation persists
T_I	inlet temperature of reactants
T_R	reactor temperature
T_W	reactor wall temperature
UA	effective area × heat transfer rate
V	volume
V_{T1}	storage capacity for technical grade HF
V_{T2}	storage capacity for anhydrous grade HF
V_R	reactor size
X	fractional conversion of $CaF_2$to HF
r	speed of response of production rate