



University Library

Author/Filing Title Zhou, D.

Class Mark T

Please note that fines are charged on ALL
overdue items.

FOR REFERENCE ONLY

0403191661



Computer-aided HAZOP of Batch Processes

by

Dingfeng Zhou

A Master's Thesis


Submitted in partial fulfilment of the requirements

for the award of

MPhil of Loughborough University

31st October 2005

© by Dingfeng Zhou (2005)

 Loughborough University Pilkington Library
Date JAN 2006
Class T
Acc No. 0403191661

Abstract

The modern batch chemical processing plants have a tendency of increasing technological complexity and flexibility which make it difficult to control the occurrence of accidents. Social and legal pressures have increased the demands for verifying the safety of chemical plants during their design and operation. Complete identification and accurate assessment of the hazard potential in the early design stages is therefore very important so that preventative or protective measures can be integrated into future design without adversely affecting processing and control complexity or capital and operational costs. Hazard and Operability Study (Hazop) is a method of systematically identifying every conceivable process deviation, its abnormal causes and adverse hazardous consequences in the chemical plants. However, this technique when applied to an entire new design is very time consuming, and often cannot be attempted until relatively late in the design of the plant. This problem is exacerbated in the case of a batch processing plant, which relies more on procedures to be followed by the operators. Each step of the operation will result in changes in the plant which is different from steady-state operation of continuous plants. It is, therefore, beneficial to use computer tools to reduce the amount of effort and time spent on Hazop analysis of batch chemical plants.

This thesis proposes using a state-based approach for automating batch Hazop. Given a batch plant and an operating procedure, the aim is to infer the state changes in the plant units after each step of the procedure so that the combined effects of the complete system behaviour can be simulated and described. Based on the state-based simulation approach, any undesirable consequences due to operating errors can be identified and reported. A novel framework for describing actions and modelling their effects, together with a simulation algorithm is described. A prototype system, called CHECKOP, is designed and implemented based on the proposed framework. Three examples are used to test the approach and the results are described.

Keywords: Computer-aided HAZOP, Batch Plants, Safety Engineering

Acknowledgements

I would like to thank my supervisors, Professor Paul Chung and Dr Steve McCoy for their invaluable technical advice and the independence they allowed me to have in doing this research. They, probably more than any other single individual, have taught me about the true scope of systems engineering. I am also extremely indebted to them on continuing support. The fundamental understanding of the behaviour of batch chemical plants that they taught me played an important role in the development of the language for reasoning system.

I am also indebted to Dr Acar, my Director of Research, and Dr Yang, internal examiner for my first year report, for useful advice about research. Without their support the systems approach taken could not have been enabled.

I would like to extend my love and appreciation to my parents as well as Jia Cheng Zhou, our unborn baby at the time of the first year of my research study. They are, and have been, a source of continual inspiration and deserve a great deal of credit for everything that I have or will achieve.

Finally, and most importantly, I would like to thank my best friend, companion, and spouse; the most significant person in my life: Jie Chen. Her belief and undying support is the single reason this task was possible.

Contents

List of Figures	ix
List of Tables	x
1. Introduction	
1.1. Background	1
1.2. Objectives and Contributions	3
1.3. Structure of the Thesis	4
2. Overview of Hazard Identification Methods	
2.1. Hazard Identification	5
2.2. Hazard Identification Methods	7
2.2.1. Checklists	7
2.2.2. Safety Reviews	7
2.2.3. Hazard Indices	8
2.2.4. Preliminary Hazard Analysis	9
2.2.5. "What If"	9
2.2.6. Failure Modes, Effects, and Criticality Analysis	9
2.2.7. Fault Tree Analysis	10
2.2.8. Event Tree Analysis	11
2.2.9. Cause-consequence Analysis	12
2.2.10. Human Error Analysis	12
2.3. Hazard and Operability Studies (HAZOP)	13
2.4. Summary	16
3. HAZOP of Batch Processing Plants	
3.1. HAZOP of Batch Processes	17
3.2. Coffee-Making Example	18

3.2.1. Information Needed	18
3.2.2. Hazop Analysis	19
3.3. Issues with Automating Batch HAZOP	25

4. Related Technologies and Applications

4.1. Signed-Directed Graphs	27
4.1.1. Overview	27
4.1.2. Related Applications	28
4.1.3. Discussion	29
4.2. Petri Nets	29
4.2.1. Overview	29
4.2.2. Related Applications	36
4.2.3. Discussion	42
4.3. Qualitative Reasoning	46
4.3.1. Overview	46
4.3.2. Related Applications	49
4.3.3. Discussion	50
4.4. AI Planning	51
4.4.1. Overview	51
4.4.2. Related Applications	56
4.4.3. Discussion	57
4.5. Conclusions	57

5. System Design

5.1. State-based Approach	59
5.2. CHECKOP System Architecture	60
5.3. Plant Description	62
5.4. Operating Procedure Description	64
5.5. The Action Model Library	65
5.6. The Deviation Generator	67
5.7. Simulation Engine	68
5.8. Flow Path Analysis	70
5.9. Reaction Model	71

5.10. Overall Algorithm	72
5.11 Summary	72
6. Examples and Discussion	
6.1. Tea-Making Example	74
6.1.1. Plant Description	74
6.1.2. Procedure Description	74
6.1.3. Result Analysis	76
6.2. Batch Reactor Example 1	78
6.2.1. Plant Description	78
6.2.2. Procedure Description	79
6.2.3. Result Analysis	79
6.3. Batch Reactor 2	81
6.3.1. Plant Description	81
6.3.2. Procedure Description	81
6.3.3. Result Analysis	83
6.4. Summary and Discussion	85
7. Conclusions and Future Work	
7.1. Conclusions	87
7.2. Future Work	89
References	91
Appendix A. Batch Reactor Example 1	
– CHECKOP Output	109
Appendix B. Batch Reactor Example 2	
– CHECKOP Output	110
Appendix C. List of Publications	114

List of Figures

Figure 4.1 A Simple Petri Net	31
Figure 4.2 Transition of a Petri Net	31
Figure 4.3 A Petri Net with an Inhibitor Arc	32
Figure 4.4 Petri Net Primitives	33
Figure 4.5 A Timed Petri Net	35
Figure 4.6 Representation Using Hierarchical Petri Nets	43
Figure 4.7 A Simple Planning Problem	55
Figure 5.1 CHECKOP System Architecture	62
Figure 5.2 A Simple Batch Plant	63
Figure 6.1 Plant Diagram for Batch Reactor Example 1	78
Figure 6.2 Plant Diagram for Batch Reactor Example 2	81

List of Tables

Table 2.1 Example Process Parameters used in Hazop Studies	14
Table 2.2 Example Hazop Guide Words and Their Meanings	15
Table 3.1 Additional Guidewords for Batch Hazop	17
Table 3.2 Batch Hazop Result for Coffee-Making Example	19
Table 5.1 Explanation of Simple Plant Description	63
Table 6.1 Sample Output of CHECKOP for Tea Making Example	76
Table 6.2 Sample Output of CHECKOP for Batch Reactor 1	80
Table 6.3 Sample Output of CHECKOP for Batch Reactor 2	83

Chapter 1

Introduction

1.1 Background

The safety of chemical processing plants is vital to their successful operation. Quality design technology relies on the ability to identify and eliminate inherent design weaknesses in advance of operations. Identification and assessment of the hazard potential in the early design stages is particularly important, so that preventative or protective measures can be integrated into the design without adversely affecting processing and control complexity or capital and operational costs.

The modern batch chemical processing plants have a tendency of increasing technological complexity and flexibility which complicates hazard evaluation. Since changes often occur in the plant, the need to identify hazards as early as possible in the development stages does not imply that hazard identification ends when the design specifications have been approved. Approval of a design in fact means only: "at the time of the study the study team believes that, provided the plant is constructed and operated in accordance with their recommendations, the plant will be acceptably safe" [Lowe and Solomon, 1983]. The first uncontrolled change during construction, or the first unapproved modification during operation, negates this approval. Consequently, hazard identification is a continuing ingredient of safe operations and should be applied, sometimes in a very simple form, to control any changes from the original intentions of the designers.

There are a number of hazard identification and hazard analysis methods used to systematize hazard identification, which can be either qualitative or quantitative in nature [Lees, 1996]. Qualitative methods can be seen as methods which deal with events at an abstract level, while quantitative methods are supporting

methods to deal with detailed analysis and evaluation of specific identified problems [Mushtaq and Chung, 2000].

One approach, the Hazard and Operability Study (Hazop), is a method developed by ICI in the 1960s for identifying hazards in chemical plant design. The method facilitates the systematic exploration of every conceivable process deviation, its abnormal causes and adverse hazardous consequences in a chemical plant [Kletz, 1999].

Hazop has been a key tool in carrying out safety analysis in the process industries. Compared to other methods, Hazop has proved to be the most powerful and effective method available for identifying hazards. It provides a systematic methodology which facilitates the extensive identification of potential hazard scenarios. On the other hand, doing Hazop needs a group of experts and it usually takes quite a long time to do. These will raise the cost. Furthermore, since Hazop is a technique for identifying the hazards of a whole system, it requires a great deal of effort from the team. It depends on the experience of the experts. Inexperienced engineers can easily miss potential problems which should have been followed-up.

In the light of the advantages and disadvantages, it is the aim of this project to develop a system which is able to handle this labour intensive job quickly, efficiently and effectively.

There are researchers working in automated Hazop [Larkin, Rushton et al., 1997; Leone, 1996; Venkatasubramanian and Vaidhyanathan, 1994; Wakeman, Chung et al., 1997; McCoy, Wakeman et al., 1999 & 2000]. But these works are limited to consideration of continuously operating plants. Limitations can also be found in other published work on computer aided risk assessment [Catino, Grantham et al., 1991; Vaidhyanathan and Venkatasubramanian, 1995; Chae and Yoon, 1994; Shimada, Suzuki et al., 1996].

A small number of researchers have looked at the problem of modelling batch plant behaviour for risk assessment [Mau, Nolan et al., 1996; Shimada, Yang, et

al., 1995; Nam, Jeong et al., 1996; Srinivasan and Venkatasubramanian, 1996]. However, these approaches were usually developed for specific plants only. A more generic system that could deal with different batch plants is required.

1.2 Objectives and Contributions

As stated in the project proposal, the overall aims of this project are:

- To enhance the current state of the art in qualitative modelling of process systems, to include batch plants.
- To test a state-based approach to physical systems modelling on a particular real-world problem, with a view to later deployment of similar approaches in other fields of application.
- To examine how different types of information can be integrated within a model to allow reliable prediction of the behaviour of a system.
- To improve the understanding of how hazardous scenarios develop in batch processing plants, in order to help reduce the future incidence of life-threatening accidents in such plants.

The specific activities necessary to achieve these aims are:

- Conduct an analysis of the types of knowledge required to capture the operation of batch plants using model-based reasoning techniques.
- Produce a definite system formalism for modelling the state-dependent behaviour of process plants through time.
- Investigate the definition and use of new models, expressed in the specified formalism, by means of software tools for user modelling of batch plant systems.
- Create a software tool for animating the models produced, allowing the user to control the process of plant simulation by means of an appropriate interface.
- Examine the requirements for systematic risk assessment in batch plants, and consider how these can be addressed by software in general and model-based systems in particular.

- Build a proof of concept software application to demonstrate one or more risk assessment techniques on a batch processing plant case study.

The implementation of this above proposed framework constitutes a major advance in the identification of hazards. Successful demonstration of this methodology will result in the following contributions:

- Formal systematisation of hazard analysis;
- Automation of hazard analysis and the detection of hazardous scenarios;
- Development of an expressive modelling language for chemical batch process representation and generation;
- Generation of an expressive and process generic software package;

1.3 Structure of the Thesis

Chapter 2 gives an overview of the conventional hazard identification and analysis methods. This forms the background on the range of tools that engineers can apply to identify potential hazards in the process industries. Chapter 3 focuses on HAZOP analysis of batch processes and explains why it is difficult to automate. Chapter 4 reviews the technologies and applications related to hazard identification of batch processes and points out what weakness they have in modelling batch processes. Chapter 5 proposes a novel approach to represent batch processes and illustrates the overall system architect of CHECKOP for modelling and simulating batch processing plants in qualitative terms. Chapter 6 uses three typical examples chosen to illustrate the HAZOP capability of CHECKOP and to identify the limitations of the current prototype. Chapter 7 concludes the current work and address the future work based on the limitations identified in Chapter 6.

Chapter 2

Overview of Hazard Identification Methods

2.1 Hazard Identification

Hazard analysis is defined as the systematic investigation of inherent, acute hazards of a process, under normal operating conditions as well as under reasonably foreseeable abnormal conditions [Lawley, 1974]. In an analysis the inherent hazards, the properties of all chemicals involved, as well as the characteristics of the process must be studied. The objective of the analysis is to determine the safe limits of the process parameters and to appreciate the effects when the process parameters move outside these limits.

The overall goal of a typical hazard analysis of a chemical process is to develop operating and design criteria intended to prevent or lessen the effects of identified hazards. In order to accomplish this goal, the analysis must begin by identifying critical material characteristics, operating conditions and faults that would cause a processing plant to deviate from its normal operating condition into a hazardous situation.

Individual experience, broadened by information from the experience of others, is the fundamental requirement for hazard identification. National and international codes and standards are examples of how this expertise is captured. The authors of a code are implying that “based on their collective experience, equipment designed to their code will be acceptably safe” [Brannegan, 1985]. Consequently, codes and practices provide minimum standards against which deviations from safe practices can be identified and appraised. Systematic

comparison of a design specification with recognised codes and practices forms the basis of one family of hazard identification methods which can be described as comparative methods. An important advantage of these methods is that the lessons learned through many years of experience are incorporated in the company's practices and thus are available to be used at all stages in the design and construction of the plant. The main task of the hazard identification study is to ensure that the company's practices, and therefore lessons learnt from its past experience, have indeed been incorporated in the design.

In the case of new processes, which are outside the scope of existing codes of practice, prior experience may not be easily available. Consequently, the hazard identification methods used in such cases have to be directed to stimulating the team members to utilize their own experience of safe and unsafe process conditions as the standard against which to evaluate the design. These methods are essentially structured ways for stimulating a group of people to apply their knowledge to the task of identifying hazards mainly by asking a series of questions.

Various procedures and techniques have been developed to aid identification of hazards throughout the different stages of new projects as well as in existing operating units. These techniques include [Lees, 1996]:

- Checklists
- Safety review
- Hazard Indices
- Preliminary Hazard Analysis
- Hazard and Operability Studies (HAZOP)
- "What if" analysis
- Failure Modes, Effects, and Criticality Analysis (FMECA)
- Fault Trees
- Cause-Consequence Analysis
- Event Trees
- Human error analysis

The next section gives an overview of the conventional hazard identification methods used in the process industries. This forms the background about what methods are used to identify the potential hazards and how they are applied in the process industries. Section 2.3 focuses on one particular hazard identification method, namely Hazard and Operability Studies, which provides the necessary background for the next chapter and the rest of the thesis.

2.2 Hazard Identification Methods

2.2.1 Checklists

Checklists manifest themselves as experience-based questionnaires and, as such, are limited to the experience base of the authors. They are engineered to demonstrate design compliance with standard procedures and provide direction for standard evaluation of chemical plant hazards. Checklists can be as detailed as necessary to satisfy the specific situation, but should be applied conscientiously in order to identify problems that require attention and to ensure that standard procedures are being followed.

A checklist is easy to use and can be applied to each stage of a project or plant development. A checklist is a convenient means of communicating the minimal acceptable level of hazard evaluation that is required for any job, regardless of scope.

2.2.2 Safety Reviews

Safety reviews can vary from an informal walk through on-site inspection that is principally visual, with emphasis on housekeeping, to a formal week-long examination by a team with appropriate backgrounds and responsibilities. Such a program is intended to identify plant conditions or operating procedures that could lead to an accident and significant losses in life or property. Safety review

allows engineers to minimize the potential hazards due to both equipment and operating procedures by assuring that all of their elements are in place and functional or corrective action is taken when the inspection results fall outside of acceptable limits.

2.2.3 Hazard Indices

Hazard indices such as that developed by Dow Chemical Company [Fire and Explosion Index, 1976] and extended by Lewis [Lewis, 1979], are methods which are designed to give a quantitative indication of the potential for hazardous incidents associated with a given plant design. The methods assign penalties and credits based on plant features. Penalties are assigned to process materials and conditions that can contribute to an accident. Credits are assigned to plant safety features that can mitigate the effects of an accident. These penalties and credits are combined to derive an index that is a relative ranking of the plant risk. Estimates of consequences in terms of cost and outage time can also be included in the evaluations. These methods are particularly useful in the early stages of hazard assessment in that they require a minimum of process and design data and can graphically demonstrate which areas within the plant require more detailed attention. They can also help to identify which of several competing process routes will contain the least inherent hazards.

The primary difference between the Dow Index and the Mond Index is that the latter specifically addresses material toxicity in addition to flammability and reactivity in assigning material factors to a process unit. The Dow Index may actually be easier to use because of the extensive use of tables and graphs in place of traditional equations, but both use the same basic calculation method.

2.2.4 Preliminary Hazard Analysis (PHA)

The main purpose of this analysis is to recognize hazards early. This is achieved by speculating about possible causes, consequences, and corrective measures early in the design process. It is generally applied during the concept or early development phase of a process plant and can be very useful in site selection. PHA is a precursor to further hazard analyses. The identified hazards can then be categorised whether they are accepted or needing further investigation. When safety is an issue, such hazard will be tracked in a hazard log subject to further review. This information will be used to reduce the severity or build-in safeguards against the effects of the identified hazards.

2.2.5 “What If”

The purpose of a “What If” analysis is to consider carefully the result of unexpected events that would produce an adverse consequence [Lees, 1996]. The method involves examination of possible deviations from the design, construction, modification, or operating intent of the plant. The questions are divided into specific areas of investigation (usually related to consequences of concern), such as electrical safety, fire protection, or personnel safety. It requires a basic understanding of what is intended and the ability to mentally combine or synthesize possible deviations from design intent that would cause an undesired result. The “What if” method enables the team to identify the potential impact of a variety of initiative events upon the original design intent. These forecasted changes will then help the team to target and prioritize the scenarios with the greatest expected benefits.

2.2.6 Failure Modes, Effects, and Criticality Analysis (FMECA)

The result of a FMECA is a tabulation of the system/plant equipment, their failure modes, each failure mode’s effect on the system/plant, and a criticality ranking for each failure mode. The related “Failure Modes and Effects

Analysis” (FMEA) is equivalent to FMECA without a criticality ranking. The failure mode is a description of how equipment fails (open, closed, on, off, leaks, etc.). The effects of the failure mode are the system responses or potential accidents resulting from the equipment failure. FMECA identifies single failure modes that either directly result in, or contribute significantly to, an important accident. Human/operator errors are generally not examined by an FMECA; however, the effects of maloperation are usually described by an equipment failure mode. FMECA is not efficient for identifying combinations of equipment failures that lead to accidents.

The method is especially useful for the analysis of very critical processes but is extremely time consuming if applied on too broad a scale. The FMECA can be performed by as few as two analysts or by a multidisciplinary team of professionals.

2.2.7 Fault Tree Analysis (FTA)

Fault Tree Analysis is a deductive technique that focuses on one particular event and provides a method for determining causes of that accident event [Lees, 1996]. The fault tree itself is a graphical model that displays the various combinations of equipment faults and failures that can result in the accident event. The solution of the fault tree is a list of the sets of equipment failures that are sufficient to result in the accident event of interest. These sets of failures are known as “cut sets”, and the smallest sets of failures sufficient to cause the top event are known as the “minimal cut sets” of the fault tree. FTA can include contributing human/operator errors as well as equipment failures.

Ranking the minimal cut sets is the final step of the fault tree analysis procedure. Structural importance is reflected by the number of basic events that are in each minimal cut set. In this type of ranking, a one event minimal cut set is more important than a two event minimal cut set; a two event set is more important than a three event set; and so on. This ranking implies that one event

is more likely to occur than two events, two events are more likely to occur than three events, etc.

The strength of FTA as a qualitative tool is its ability to break down an accident into basic equipment failures and human errors. This allows the safety analyst to focus preventive measures on these basic causes to reduce the probability of an accident.

Both failure modes and effects analysis and fault tree analysis are useful aids to hazard identification as they both structure and document the analysis. However, because they involve very detailed analysis of components and operations, their use in the process industry is mainly limited to identification of special hazards where they form the basis of quantification of risks.

2.2.8 Event Tree Analysis (ETA)

Event trees are modified form of the decision trees traditionally used in business applications. Event trees provide a precise way of recording the accident sequences and defining the relationships between the initiating events and the subsequent events that combine to result in an accident. Then by ranking the accidents, or through a subsequent quantitative evaluation, the most important accidents are identified. Event trees are well suited for analysing initiating events that could result in a variety of effects. An event tree emphasizes the initial cause and works from the initiating event to the final effects of the event. Each branch of the event tree represents a separate effect (event sequence) that is a clearly defined set of functional relationships.

Event tree analysis is a technique for evaluating potential accident outcomes resulting from a specific system failure or human error known as an initiating event. Event tree analysis considers operator response or safety system response to the initiating event in determining the potential accident outcomes. The

results of the event tree analysis are accident sequences; that is, a chronological set of failures or errors that define an accident. These results describe the possible accident outcomes in terms of the sequence of events (successes or failures of safety functions) that follow an initiating event. Event tree analysis is well suited for systems that have safety systems or emergency procedures in place to respond to specific initiating events.

2.2.9 Cause-consequence analysis (CCA)

Cause-consequence analysis combines the forward thinking features of event tree analysis with the reverse thinking features of fault tree analysis. The result is a technique that relates specific accident consequences to their many possible basic causes.

The solution of the cause-consequence diagram for a particular accident sequence is a list of accident sequence minimal cut sets. These sets are analogous to fault tree minimal cut sets because they represent all the combinations of basic causes that can result in the accident. A quantitative analysis using these sets can provide estimates of the frequency of occurrence of each accident event sequence.

A major strength of cause-consequence analysis is its use as a communication tool: the cause-consequence diagram displays the interrelationships between the accident outcomes (consequences) and their basic causes. The method can also be used to quantify the expected frequency of occurrence of the consequences if the appropriate data are available.

2.2.10 Human Error Analysis (HEA)

Human beings are key components of industrial processes. They are involved in each step of process design, operation, maintenance, etc. Human error analysis

is a systematic evaluation of the factors that influence the performance of human operators, maintenance staff, technicians, and other plant personnel in the plant. Behavioural and/or causal guide words are systematically applied to what human people do, usually a task or a scenario so that situations where the problems lie could be found. Behavioural guide words take a so called phenotypical perspective as a starting point, which might be an action omission, actions in a wrong order, time error (too early, too late) and qualitative error (too much, too little). Its primary purpose is to identify potential human errors and their effects, or identify the cause of observed human errors.

2.3 Hazard and Operability Studies (Hazop)

Hazop is a technique which was first developed by ICI in the 1960's, for examining process designs for hazards and operability problems. Since then it has been widely adopted in the chemical process industries and adapted for use in many other domains. The essence of the traditional Hazop study is to examine the detailed process design, as expressed in the piping and instrumentation drawings (P&IDs) of a new plant.

Firstly, a Hazop team (usually comprised of around 6 people) is assembled. The members of the team should include process engineers involved in the design of the plant, operational staff concerned with day-to-day operations, mechanical and control engineers with an interest in the project. A team leader must be appointed, who has no formal connection with the project, but has the sole duty of managing the Hazop study meetings. A scribe may also join the team, in order to record the actions arising from the study.

Meetings should then be scheduled when all team members are able to attend – the duration should not be too long (max. 2 or 3 hours) and the meetings should not be scheduled too closely together, as they are quite intensive for the team members. The P&IDs should be allocated to meetings in the series, so that the whole process is covered in a fairly logical sequence and the most important areas of the process are examined early on in the sequence. The team leader

should allocate these according to experience of time needed for Hazops, and under guidance of the project team, who know the process design.

In the Hazop meetings, the team identify a number of sections of a P&ID, called “nodes”, to be examined in sequence. Each node should cover a process line or well-defined process equipment item.

In examining each study node, the team firstly agree on the process design intention of the node. Then, they identify all the possible ways that the plant could fail to achieve this design intention, using a method for generating deviations and assessing whether the deviations are possible or not.

Each deviation is composed of a process parameter (taken from Figure 2.1) and a guide word (taken from Figure 2.2). [Tables adapted from: IChemE Safety Training Package 034, “Hazop and multistage hazard study”, 1999]. All sensible combinations of process parameter and guide word are examined for each study node. For each deviation, the team consider whether there are any possible causes of the deviation and whether there are any hazardous consequences or operability problems arising from the deviation itself or from the identified causes of that deviation.

Flow
Pressure
Temperature
Mixing
Stirring
Level
Viscosity
etc...

Figure 2.1 Example process parameters for use in Hazop Studies

Guide Word	Meaning
None	None of the design intent is achieved
More of	Quantitative increase in a parameter
Less of	Quantitative decrease in a parameter
More than (As well as)	An additional activity occurs
Part of	Only some of the design intent is achieved
Reverse	Logical opposite of the design intention occurs
Other than	Complete substitution. Another activity takes place
Before / After	The step (or some part of it) is effected out of sequence
Early /Late	The timing is different from the intention

Figure 2.2 Example Hazop Guide Words and their Meanings

Any significant problems are recorded in a tabular format, giving the deviation, causes and consequences. The team also record any instrument systems intended to protect against the problem, as well as any suggested actions arising from the discussion. The purpose of the Hazop meeting is NOT to solve the problem identified – that will normally be taken care of outside of the meeting.

After the Hazop meetings, the actions are dealt with by the various engineering departments (process, mechanical, etc.), and should be sorted out before design is approved for construction.

Hazop is known to be the best technique for hazard identification, as it is systematic, complete and rigorous in looking at all combinations of events which could present problems. However, the method does have a high cost in terms of the quantity of expert time needed to conduct the meetings, which makes them difficult to schedule. Hazop meetings also impose a significant strain on participants and, because they are usually conducted after the detailed process design of a plant, can delay the completion of a project. For this reason, there is much interest in methods which can reduce the burden of Hazops, in order to free up engineers' time for other tasks.

2.4 Summary

This chapter has given an overview of the different methods that are available for hazard identification and analysis. Hazard indices is a good tool in the early stages of hazard assessment of the plant with minimum process and design data. Checklist is easy to use and can be applied to each stage of a project but it is experience-based questionnaires. Preliminary hazard analysis categorise the identified hazards. It is generally applied at the concept or early development stage of a process plant. "What if" analysis examines the adverse impact of unexpected events upon the original intent. FMECA is useful for the critical process analysis but could be very time consuming for a fairly large scaled system. Fault tree analysis is a deductive and qualitative technique which breaks down an accident into basic equipment failures and human errors. It is limited to identify special hazards based on quantified risks. ETA evaluates potential accident outcomes resulting due to an initiating event such as system failure or human error. This technique is suitable for those systems with safety systems or emergency procedures. CCA combines features of event tree analysis and fault tree analysis by relating consequences to their possible causes. HEA is used to identify the impact of the performance of human operators. Particular attention has given to the description of HAZOP as it is one the most widely used and respected hazard identification techniques used in the chemical process industry. HAZOP is very similar to engineer's analysis on the real process plant. In Hazop, each node of P&ID is studied and identified all the possible ways that the plant could fail to achieve design intention by generating deviations with guide words and assessing whether these imagined deviations are possible or not. The causes and consequences of these deviations are recorded for further preventive actions.

Chapter 3

HAZOP of Batch Processing Plants

3.1 HAZOP of Batch Processes

The issues in HAZOP analysis for batch processes are significantly different from those for continuous plants. In a batch process, operations are performed in a sequence of “operating instructions”. Rather than the plant remaining at a “steady state”, the process variables associated with operations could change during the execution of an individual instruction. Therefore, the methods applied to Hazop continuous processes are not sufficient to Hazop analysis for batch process. Recently, Mushtaq and Chung [2000] proposed a systematic Hazop procedure for batch processes and an application of this method to pipeless plants. New guidewords are introduced for Hazop of batch processes in addition to the basic set used in continuous processes. Table 3.1 below lists these guidewords for batch Hazop:

Batch Hazop Guidewords
Reverse
Early
Late
Before
After
Quickly
Slowly

Table 3.1 Additional Guidewords for Batch Hazop

In addition to applying guidewords to the process variables in the batch process, which is similar to the method used in continuous processes, Hazop analysis of

a batch process also takes into account the 'deviations' from operating instructions, some of which are:

- Keywords *Before* and *After* – used for situations where something happens before or after it is expected in terms of sequence;
- Keywords *Early* and *Later* – used for situations where something happens earlier or later than expected relative to clock time;
- Keywords *Quickly* and *Slowly* – used for situations where something happens quicker or slower than expected;
- Keyword *Reverse* – used for situations where something opposite the original intention is achieved.

3.2 Coffee Making Example

In this section a Hazop analysis of a batch process – making a cup of coffee – is carried out manually by applying guidewords as discussed above in order to show the kind of output expected of an automated batch Hazop system.

3.2.1 Information Needed

To do a batch Hazop on making a cup of coffee, the following information is needed:

- A procedure of making a cup of coffee to be analysed.
- Guidewords such as More, Less, Early, Quickly to be applied to the process or process variables.
- Standard set of process variables such as Flow, Temperature, and Level, to which guidewords are applied.
- A set of Instruction words such as Mix, React, and Heat, to which guidewords are also applied.

3.2.2 Hazop Analysis

Firstly, a set of instructions to be analysed is prepared:

1. Fill water in an electrical kettle
2. Plug the socket into the main
3. Turn on the switch on the kettle (the red light is on showing it is connected)
4. Wait for water in the kettle to boil, while filling in a cup with some instant coffee with a spoon
5. Pour boiled water into the cup
6. Add milk into the cup
7. Add sugar into the cup with a spoon
8. Stir immediately with a spoon until sugar is dissolved.

Then perform a batch Hazop by applying guide words to the above procedure as well as process variables in each step.

Step	Guide Words	Deviation	Possible Causes	Consequences/Output
1	NO	No water in the kettle	1.No water supply 2.Tap fails open	No coffee
1	MORE	Too much water in kettle	3. Tap is left open for too long.	Spillage while overflowing due to heating expansion.
1	LESS	Less water in kettle	4.Operator fails to check the water level	No boiled water
1	OTHER THAN	Maintenance	5.Loss of containment of the kettle	No coffee
1	Early	Fill water in the kettle too early.	6. Operator decided to make coffee, but then changes his mind.	Water in the kettle not fresh.

Step	Guide Words	Deviation	Possible Causes	Consequences/Output
1	Late	Fill water in the kettle too late.	7.Operator forgets to fill water in the kettle before someone reminds him or he realised about it.	Resistor or fuse burns.
1	After	Fill water in the kettle after the kettle switch is turned on.	8. Operator fails to follow the instruction.	Resistor or fuse burns.
1	Quickly	Fill water in the kettle too fast.	9.The water in the kettle overflows or spills out of kettle.	Spillage while overflowing due to heating expansion.
1	Slowly	Fill water in the kettle too slowly.	10. Water is below the required level. 11. It takes too long preparing coffee.	No boiled water or time consuming
2	NO	No socket available	12. Socket is being used for some other purpose.	No coffee
3	NO	Switch Not turned on	13.Operator forgets turn on the switch 14. No electrical supply. 15. Socket fails to plug into main properly.	No boiled water

Step	Guide Words	Deviation	Possible Causes	Consequences/Output
3	OTHER THAN	Maintenance	16. Main power socket failures 17. Kettle socket failures 18. Red light failures 19. Electrical circuit failures	No coffee
4	NO	No bottled instant coffee	20. Instant coffee previously used up	No coffee
4	NO	No spoon available	21. Spoon was lost or used for other purpose.	Operator would fail to control the amount of sugar added into the coffee.
4	MORE	Too much coffee in the cup	22. Operator is not experienced or has no sense about how much instant coffee should be put into the cup.	Coffee is too strong
4	LESS	Less coffee in the cup	Covered by (22)	Coffee is too weak
4	EARLY	Operator fills instant coffee in the cup too early.	Covered by (22)	Coffee is solidified and tastes not well.
5	LESS	Temperature	23. Water is below boiling point.	Coffee smells not strong

Step	Guide Words	Deviation	Possible Causes	Consequences/Output
5	MORE	Pour more water in the cup	24. Water overflows	Coffee is too weak
5	BEFORE	Pour out water before it boils.	Covered by (23)	Coffee smells not strong
5	QUICKLY	Flow	25. Water spillage or overflow over the cup.	Coffee contaminated the desk
5	SLOWLY	Temperature	26. Water becomes cooler.	Coffee tastes not well
6	NO	No milk available	27. Milk was used up.	Coffee tastes not smoothly
6	MORE	Too much milk in coffee	28. Operator is not experienced or has no sense about how much milk should be put into the cup.	Coffee is too weak.
6	LESS	Less milk in coffee	Covered by (22)	Coffee tastes not smoothly
6	QUICKLY	Flow	Covered by (25)	Coffee contaminated the desk
6	SLOWLY	Temperature	Covered by (26)	Coffee tastes not well
7	NO	No sugar available	29. Sugar was used up	Coffee doesn't taste sweet.
7	NO	No spoon available	Covered by (21)	Operator would fail to control the amount of sugar added into the coffee.

Step	Guide Words	Deviation	Possible Causes	Consequences/Output
7	MORE	Too much sugar in coffee	30. Operator is not experienced or has no sense about how much milk should be put into the cup.	Coffee is too sweet
7	LESS	Less sugar in coffee	Covered by (22)	Coffee is not sweet enough.
7	QUICKLY	Flow	Covered by (25)	Sugar or coffee spillage out of cup.
7	SLOWLY	Temperature	Covered by (26)	Coffee tastes not well.
8	LESS	Operator failed to stir the liquid properly.	31. Distraction 32. Operator is not experienced	Coffee tastes not very well.
8	MORE	Operator stirred the liquid too long	33. Covered by (26)	Coffee tastes not well
8	OTHER THAN	Maintenance	34. Spoon suddenly broke when stirring.	Coffee tastes not very well
8	BEFORE	Operator stirred before step 1 or 4.	Covered by (8)	Coffee tastes not very well
8	AFTER	Operator stirred after drinking.	Covered by (8)	Coffee tastes not very well

Step	Guide Words	Deviation	Possible Causes	Consequences/Output
8	QUICKLY	Flow	Covered by (25)	Sugar or coffee spillage out of cup
8	SLOWLY	Temperature	Covered by (26)	Coffee tastes not well

Table 3.2 Batch Hazop Result for Coffee-Making Example

In a batch process, the operator plays a key role in the individual processing instructions. An inadvertent operation by the plant operator could lead to direct or indirect hazards. The operator could perform the operations in a sequence different from that required by the instructions. For example, the operator could pour water out of the kettle before it boils in step 5. The direct consequence that step 5 is performed before step 4 is that the operator can't get boiled water. The indirect consequence for that is the operator can't get a right coffee in step 8. The keywords such as Before, After are used to generate such situations.

The operator could also induce hazards by initiating or terminating a step either earlier or later than indicated in the instruction. For example, the operator may fill the instant coffee in the cup too early in step 4. The direct impact is the instant coffee is solidified. The indirect impact is that the coffee may taste not very well finally. The keywords like Early, Late are used to generate such incidents.

The operator may work faster or slower than required by the instructions. For example, in step 8, he may stir the coffee in the cup faster than expected. This can often lead to hazards, i.e., the coffee will spill out of the cup. The keywords like Quickly, Slowly are used to generate such situations.

Process variable deviations in batch process are similar to those in continuous process. The guide words such as No, More, Less, Part of, As well as, Other

Than are used to modify the process variables like Flow, Temperature, and Level in each individual step to generate process variable deviations.

3.3 Issues with Automating Batch HAZOP

In batch processes, the plant P&ID is obviously not sufficient to represent the plant operations, called instructions. The instructions emphasise the sequential nature of the batch process. An instruction can comprise of a set of processing steps. After the first step is completed, the second step is started and so forth till the last step is finished. Each step is performed in an equipment unit and causes the state of the equipment, or process variables within the equipment, to change. When the next step is started, the previous process variable or variables may remain or reach another particular value or values. That process variable may have a direct impact on the equipment unit or indirect impact on the other equipment units by propagation leading to hazards, or an accumulative adverse consequence or consequences on equipment unit(s) when performing another consequent step(s).

The greatest difficulty in automating batch Hazop is in modelling batch processes. The methods used for modelling continuous plants, such as signed directed graphs (SDGs), cannot account for the discontinuities in batch operations. The methods used for modelling discrete events such as Petri nets are too complex and not flexible enough to deal with changes in the operating procedure. Even a simple coffee example could have 33 nodes to be analysed. An ideal representation should represent the instructions in a proper way so that none of the necessary information is lost. Depending on the representation chosen, an atomic step in a set of instructions could deliver information in very different configurations. In the coffee example, step 1 'Fill water in an electrical kettle', contains information about where the step is performed, what action is performed, what object it is being performed on, and what intention is to be reached. However, step 4 'Wait for water in the kettle is boiled' just contains information about what action is performed and what intention is to be reached.

This makes it difficult to generate a uniform and structured format with reusability and scalability, suitable for implementing batch Hazop automation.

Another difficulty is in representing the causal relationship between state variables. An instruction contains information about the sequence of each action to be performed. However, it does not contain any information about the cause and effect relationships between process variables. When a team of experts perform Hazop analysis of a plant, they use their mental models of the process to obtain cause and effect relationship. This causal model of each step is critical for conducting batch Hazop analysis.

In batch operation, a reaction could lead to the change of the states such as the amount or heat content of the materials in an equipment unit. In the coffee example, the operator stirs the content of the cup not only making the coffee tastier but also makes the content more homogeneous. Modelling the transformation of the materials should also be included.

The duration for which each step is performed is important for batch Hazop analysis. In the coffee example, if heating is performed on the water for a shorter time than indicated, the temperature of the contents is quite low which results in failure to make a cup of coffee. An action should be terminated when the expectation is reached.

Equipment malfunction is process independent and occurs when there are deviations in the state variables. In the coffee example, when the tap fails to open, there is no flow path from the water tank to the kettle, so the kettle cannot be filled. Therefore, the propagation modeling is also important for batch Hazop analysis.

Chapter 4

Related Technologies and Applications

This chapter reviews the technologies and applications related to hazard identification of batch processes, which is the focus of this thesis.

4.1 Signed-Directed Graphs

4.1.1 Overview

SDG model is an important technology of building qualitative model, especially for hazard evaluation and fault diagnosis. SDG model can be used to express the complex cause and effect relations, and has great potential for modelling process knowledge. The structure of SDG model is straightforward and easy to modify.

The key technique of using SDG for qualitative simulation is the high efficiency two-direction inference engine, which is performed by software automatically. A SDG contains the process variables, for example, flow in a physical system. The variables are connected by arcs to reflect the influence one variable has on another variable. Each arc is labeled “+” or “-” to indicate what kind of influence it is. For example, $X \xrightarrow{+} Y$ indicates Y will increase/decrease if X is increased/decreased. $X \xrightarrow{-} Y$ indicates Y will decrease/increase if X is increased/decreased. Up to now, SDG has been the most effective method in computer aided process hazard assessment.

4.1.2 Related Applications

HAZOP analysis is very laborious and time consuming, so there is a significant motivation to automate this activity. An automated system can help to make the analysis more thorough and detailed, and minimize human errors as well. A lot of work has been done in the effort to automate the Hazop analysis for continuous chemical plants using SDG technology.

A number of research projects have been carried out with the aim of developing automated HAZOP tools. [McCoy, 1999] gives a brief history of the work done at Loughborough University on automating HAZOP. Chung [Chung, 1993], on the other hand, developed a generic design tool for HAZOP and other techniques, called QUEEN (Qualitative Effects Engine) by using SDG (signed directed graph). The tool QUEEN was used to connect a frame-based unit modelling system, in which each sub-system was modelled by a SDG at a sub-level, so as to build up the whole plant model. Jefferson et al. [Jefferson, Chung et al., 1995] extended Chung's work to develop CHEQUER (Computer HAZOP Emulation using Qualitative Effects Reasoning). McCoy et al. [McCoy, Wakeman, et al., 1999a; 1999b; 1999c; 2000a; 2000b] further extended the QUEEN and HAZOP algorithms and developed a tool known as HAZID within the STOPHAZ project. HAZID is now being developed into a commercial tool [McCoy et al, 2004]. However, all of these previous works on automating HAZOP focused primarily on handling continuous plants. The basic technology employed are not suitable for handling batch plants.

Another significant group doing research in this area is based at Purdue University. Vaidhyanathan and Venkatasubramanian [1996] developed "HAZOPEXpert", a system based on G2 system which is a model based framework and a process-related expert system for automating the HAZOP analysis of continuous plants. A kind of SDG representation called HDG (Hazop Digraph) is used to add nodes representing faults and consequences related to process variable deviations. A semi-quantitative reasoning methodology is used for filtering and ranking the HAZOP results using

additional quantitative information in the form of design and operating specifications of the process units and process material property values. The qualitative values “high”, “low”, “zero” or “normal” are applied to process variable nodes which are connected by the directed arcs to indicate the qualitative causal influence between the process variables at a “positive” or “negative” gain level. However, they did not discuss about how they deal with the situations like “zero” or “normal”.

4.1.3 Discussion

A SDG appears to be well-suited to depicting the continuous plants. However, it seems not to be applicable to the context of batch processing. A batch process has a special nature, in which the behaviour of the plant changes over time, and is dependent on the states of its equipment items.

4.2 Petri nets

4.2.1 Overview

The Petri nets representation was first proposed by Carl Adam Petri in his doctoral dissertation work on communication with automata in 1962 [Petri, 1962]. Causal relationships between events in a computer system are described using a net. A.W.Holt [Holt, Saint et al., 1968] and others of the Information System Theory Project of Applied Research, Inc. in the United States illustrated how Petri nets could be used to model and analyse systems of many concurrent components. Petri’s work also came to the attention of The Computation Structure Group at Massachusetts Institute of Technology, led by Professor J.B.Dennis [Dennis, 1970]. Several doctoral theses and technical reports were published during the early 1970s. Most of the publications on Petri nets before 1980 were listed in the annotated bibliography of the first book on Petri nets [Peterson, 1981]. The work done in Europe on Petri nets and the published papers are presented in the second Petri net book [Resig, 1985].

Starting in the late 70's, Petri nets became a very active area, especially in Europe. Annual conferences on Applications and Theory of Petri nets have been held since 1979 and the proceedings published in the series of Lecture Notes of Computer Science by Springer Verlag. Most of the studies focused on information processing systems in the computer science community. An excellent tutorial paper was given by Professor T. Murata in 1989, which comprehensively presented properties, analysis, and applications of Petri nets and a list of references of significance [Murata, 1989].

A number of variations have been developed to enrich the modelling power of Petri nets, but most of these variations are simply additions to a basic Petri net. A basic Petri net structure consists of a finite set of places, a finite set of transitions, a finite set of arcs, and a set of tokens. The tokens, which are small solid dots, define markings. The set of arcs joins some places to some transitions, or some transitions to some places. A directed arc never joins a place to a place or a transition to a transition. Places are represented by circles and transitions by rectangles or bars. A Petri net containing tokens is called a marked Petri net. In a marked Petri net, transitions may be enabled and fired. Once the transition fires, the tokens are redistributed. This results in a new marking. For a formal definition of an ordinary Petri net, please refer to Murata's or Peterson's tutorial materials [Murata, 1989; Peterson, 1981]. Figure 4.1 is an example net containing all components of a Petri net [Kimblér,1997]. An arc has capacity 1 by default; if other than 1, the capacity is marked on the arc. Places have infinite capacity by default, and transitions have no capacity, and cannot store tokens at all.

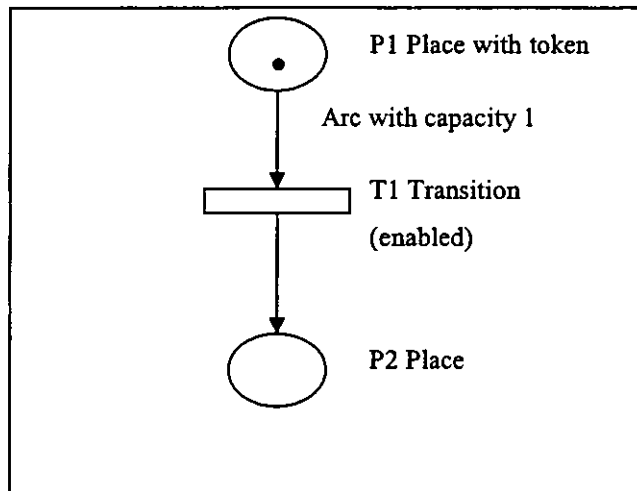


Figure 4.1 A Simple Petri Net

A transition is enabled when the number of tokens in each of its input places is at least equal to the arc weight going from the place to the transition. An enabled transition may fire at any time. When fired, the tokens in the input places are moved to output places, according to arc weights and place capacities. This results in a new marking of the net, a state description of all places, as shown in figure 4.2.

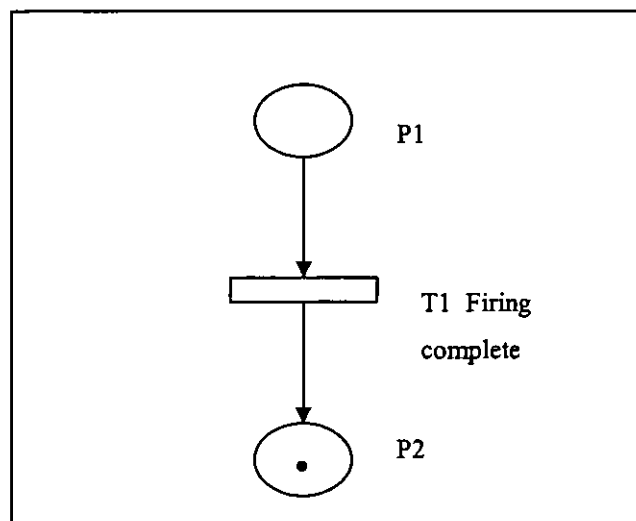


Figure 4.2 Transition of a Petri Net

A special kind of arc, the inhibitor arc, is used to reverse the logic of an input place. With an inhibitor arc, the absence of a token in the input place enables the transitions, not the presence. The transition in figure 4.3 cannot fire, because the token in P2 inhibits it.

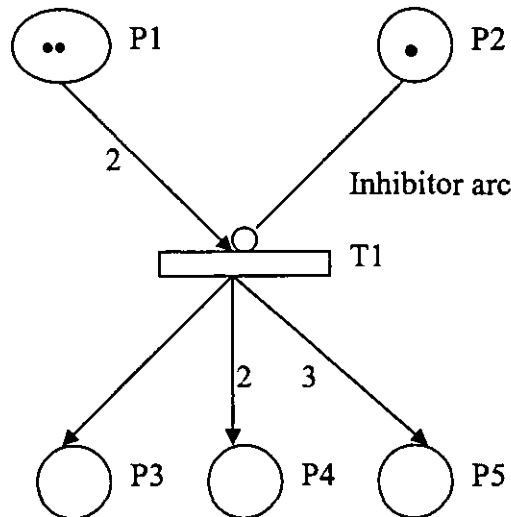


Figure 4.3 A Petri Net with an Inhibitor Arc

The real world has very sophisticated logic and control structures which can be developed using so called the primitives (Figure 4.4) including sequence, conflict, concurrency, synchronization, confusion, merging and priority and inhibit [Kimbler, 1997]. Sequence is obvious - several things happen in order. Conflict is not so obvious. The token in P4 enables three transitions; but when one of them fires, the token is removed, leaving the remaining two disabled. Unless we can control the timing of firing, we don't know how this net is resolved. Concurrency is obvious - many systems operate with concurrent activities, and this models it well. Synchronization is also modelled well using Petri Nets; when the processes leading into P8, P9, and P10 are finished, all three are synchronized by starting P11.

Confusion is another not so obvious construct. It is a combination of conflict and concurrency. P12 enables both T11 and T12, but if T11 fires, T12 is no longer enabled.

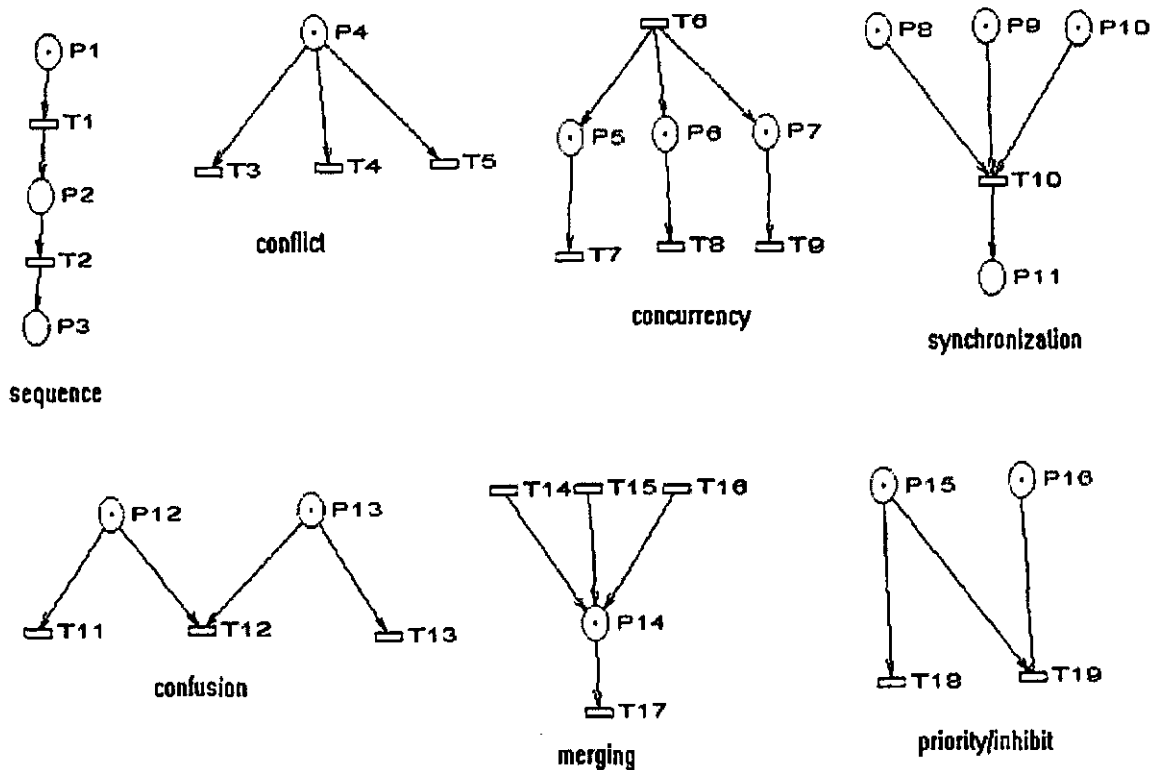


Figure 4.4 Petri Net Primitives

Merging is not quite the same as synchronization, since there is nothing requiring that the three transitions fire at the same time, or that all three fire before T17; this simply merges three parallel processes. The priority/inhibit construct uses the inhibit arc from P16 to control T19; as long as P16 has a token, T19 cannot fire.

The Petri net as a mathematical tool possesses a number of properties, such as reachability, boundness, conservativeness, safety and liveness. These properties can be referred to [Murata, 1989; Proth and Xie, 1996; Zhou and Venkatesh, 1998] for further details.

To deal with the temporal performance and the net size and complexity, various extensions of Petri nets such as timed Petri nets, stochastic Petri nets, predicate/transition (Pr/T) nets, and coloured Petri nets have been developed.

When time is associated with places, it is called a timed place Petri net, or p -timed Petri net. A token held in a place becomes available only after a certain period of time. Only available tokens can enable transitions. When time is associated with transitions, it is called a timed transition Petri net, or t -timed Petri net. In a timed transition Petri net, there are two modes of firing. One mode is that tokens are removed from the input places when a transition becomes enabled. The transition fires after a certain period of time and deposits tokens on the output places. Another mode is that tokens remain on the input places of an enabled transition. After the holding/delay time, the transition fires by removing tokens from the input places and depositing tokens on the output places. Formal definitions of timed place Petri net can be found in the literature, e.g. [Zhou and Venkatesh, 1998].

In chemical process plant, the place sometimes represents an operation which will take time. Time has to be added into the PN to represent the system behaviour. An example of p -timed PN is shown in figure 4.5. In this PN, a token arriving at place P1 will not be available for 0.25 hours before the transition T2 can be fired.

Stochastic timed Petri net (STPN) is introduced when time is considered as a random variable, or probabilistic distributions are added to the above timed Petri net. It is conventional to associate time delays with the transitions only in STPN. When the STPN allows for immediate transition firings, it is called a generalised stochastic Petri net.

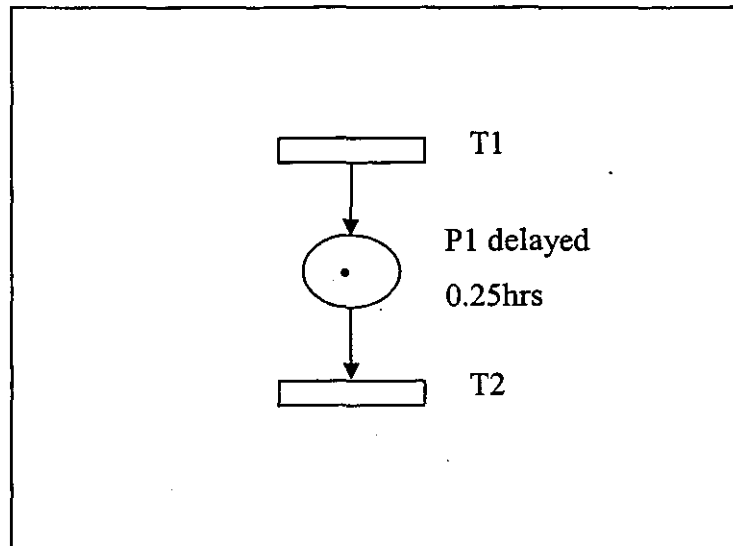


Figure 4.5 A Timed Petri Net

High-level Petri nets such as Pr/T nets (Predicate Transition Nets) and coloured Petri nets (CPN) are introduced to deal with the net size and complexity of the other types of Petri nets. Pr/T nets are developed from classical Petri nets by adding the concept of individual tokens. The tokens in Pr/T nets are no longer indistinguishable black dots, but have their own individuality and are therefore distinguishable. Places are called predicates and may carry one or more individual tokens which are limited by a defined capacity. Arcs transfer one or more individual tokens given by the token weight. An inscription on the arc denotes the types of tokens transferred by that arc. Transitions may be labelled with logical expressions. The execution rules are similar to those of classical Petri nets with the extension that the individual tokens must be true. CPN emphasize the types (e.g., production types), variables (e.g., capacities in buffers) and expressions (e.g., representations of the firing rules).

CPN use different colours to distinguish tokens. In chemical process plant, we can use coloured tokens to carry complex information or data. For example, a coloured token can represent a mixture of Solvents C and D, their proportionality, the temperature, level, pressure, etc. When the transition fires, i.e. an event happens, the token will output to the next place through the transition. The state or states of the corresponding places will then change. This

will facilitate the analysis of the system behaviour and alleviate the complexity of a classical Petri net representation [Johnsson, 1992].

4.2.2 Related Applications

4.2.2.1 Modelling and Analysis

Batch processing operations, such as start-up and shutdown, are typical examples of discrete-event dynamic systems. Petri nets can be used to describe the behaviour of discrete event activities in batch processes systematically and graphically.

In Petri net modelling of batch processes, there are two primitive concepts: events and conditions. Events are actions that take place in the system. The occurrence of these events is controlled by the state of the system. The state of the system can be described as a set of conditions. A condition is a predicate or logical description of the state of the system. For an event to occur, it may be necessary for certain conditions to hold. These are the preconditions of the event. The occurrence of the event may cause the preconditions to cease to hold and may cause other conditions, postconditions to become true.

In Petri net modelling, the places can be used to represent the status of a resource (e.g. its availability), a process plant operation (e.g. whether it is operating or not) and a condition (e.g. whether it is satisfied or not). The transitions are used to model events, e.g. the start and end of an operation. The presence of a token in a place indicates if a resource is available, a process plant operation is undergoing, or a condition is true. Multiple tokens imply availability of multiple resources. Ordinary Petri nets, Pr/T Petri nets, coloured Petri nets, and timed Petri nets have been used in the logical and temporal modelling of batch processes.

Yamalidou, Patsidou and Kantor [1990] examined three discrete event system techniques including Petri nets in batch processes. A multipurpose batch plant

with three units and three products was modelled as a p -timed Petri net. Yamalidou and Kantor [Yamalidou and Kantor, 1991] later used a set of modelling rules in their development of CPN modelling for networks of processing equipment, such as reactors, heat exchangers, separation vessels, as well as the connecting pipes. In their modelling, the chemical batch process is divided into elementary units. Petri net models for each atomic element, such as batch task or operation, equipment unit, valve and pipe, are defined. These sub-nets are then connected to each other to form a global Petri net model of the batch process.

Andreu, Pascal, Pingaud, and Valette [Andreu, Pascal et al., 1994] worked on a hierarchical approach in manufacturing system applications to batch process plants, where the possibility of using a hierarchical Petri net based approach for describing the discrete aspect of batch systems was introduced. The hierarchical approach allows the decomposition of the whole problem into smaller sub-problems within three different levels: local control level, co-ordination level and supervision level. The possible use of various Petri net classes at each hierarchical level offers an important advantage for consistency. They applied this method for modelling batch control through a multipurpose batch plant.

Valette [Valette, 1995] addressed the benefits of a Petri net based approach in modelling event-driven operations of process systems. In contrast with state transition graph, a Petri net not only describes the system functioning (behavioural model), but also captures the system structure (structural model). Silva et al. [1998] presented a tutorial on the utilisation of Petri nets model in several stages of a batch process life cycle through a selected set of examples.

Tittus et al. [1995] introduced Petri net models for plant resources and recipes. Two generic classes of resources: processing devices (tanks, reactors, and other container-like units) and transporting devices (valves and connecting lines) are considered. Recipes are formulated as composed of five general elements: operation sequences, moving materials, and different ways to mix materials,

adding materials during an operation, and splitting of the materials. The Petri net model for the whole batch plant can then be constructed by synchronising different resource models with the recipe model.

Teiji Kitajima et al [Kitajima et al., 2000] describe a modelling and analysis method for batch sequential control system by using colour Petri net model. They propose systematic procedures to create CPN models from various information based on the ISA S88 standard [Batch Control, 1995], i.e. master recipe, equipment information and batch schedule. Batch sequential control systems are easily simulated and certain conflicts of operation in the process can be found out. This is a novel approach to model and analyse the batch sequential control systems.

Bottom-up, top-down and hybrid approaches for Petri net based modelling have been widely used in the manufacturing field [Zhou and Venkatesh, 1998]. In the bottom-up approach, combining sub-Petri net models, where modular models of special operations or activities and combining rules are required, develops the global Petri net model. In the top-down approach, the transitions and places in a Petri net model are replaced by more detailed sub-Petri nets so that an arbitrarily large Petri net model can be obtained. Bottom-up focuses on a correct construction of interactions among subsystems or detailed operations. The hybrid approach uses stepwise refinement (top-down) followed by a bottom-up approach. In addition, CASE tools to support batch process modelling are used to enhance industrial applications.

4.2.2.2 Verification and Diagnosis

Valette [Valette, 1995] addressed the verification issue of deadlock free discrete event chemical processes. Through a batch process case study, the main issues regarding deadlocks were discussed. These included the reduction rules, where redundant places were eliminated and the deadlock-free batch process was validated.

Srinivasan and Venkatasubramanian [Srinivasan and Venkatasubramanian, 1998] combined high-level Petri nets and digraphs with an object-oriented knowledge representation to provide a general framework for automating HAZOP analysis. In this framework, a system for automating HAZOP analysis of batch chemical processes, called *Batch HAZOPExpert*, was implemented in the object-oriented architecture of Gensym's real-time expert system G2. Tests on real case studies show that *Batch HAZOPExpert* can generate the results of conventional hazard review when process units and product recipe have been modelled.

4.2.2.3 Supervisory control

A batch process control system can be refined into four levels: planning, scheduling, supervision and co-ordination, and local control. Planning handles the production strategy according to a total representation of the plant. Scheduling addresses the route of operations in light of resource capacities of the process units. The aim of supervision and co-ordination is to implement real-time scheduling and realise the co-ordination of sub-systems in terms of resource availabilities and actual states. The local control implements the real-time control, maintaining safe and steady operation of the process despite disturbances of process variables. The supervision and co-ordination level mainly deals with sequence, synchronisation, concurrence, conflicts, and resource sharing of discrete activities, in which Petri net has found its successful applications.

In the Petri nets modelling, one of the main supervisory control tasks is to guide the system from a given initial marking or state to a desired final marking or state. Yamalidou et al. [Yamalidou and Kantor, 1991] presented a formulation based on linear optimisation. The optimisation objective is to find the firing sequence to bring the system from its initial state to a certain specified final state. The constraints include two sets: dynamic and operational constraints. The former is imposed by the dynamics of Petri nets model. The latter is derived from Boolean expressions of the system operational restrictions.

Batch plants are normally composed of several processing units working in parallel and/or sequence coupled by flows of mass and energy. The sequence, synchronisation and resource sharing are the main concerns in co-ordinate control. Hanisch [1998] proposed the co-ordination control modelling approach via ordinary Petri nets and coloured Petri nets, where the co-ordination control law was embedded in the Petri nets model.

Boissel and Kantor [1993] addressed the supervisory control problems associated with batch process plants whose specifications can be expressed in terms of forbidden states. Forbidden states represent undesirable or catastrophic situations in which the production goals cannot be satisfied. Two kinds of forbidden state problems, resource conflict and deadlock, were formulated. A pipeless batch process plant to be controlled was modelled using a Petri net. The purpose of control rule synthesis was to find another simple Petri net to be added to the original system net, and the overall Petri net model would provide the optimal system behaviour. This optimal control problem was solved using a simulated annealing technique.

4.2.2.4 System design

Batch plants combine features of the continuous world as well as the discrete one. A comprehensive design approach for batch plants should include both discrete event and continuous aspects, i.e. hybrid system design. Typically, different techniques from each system are adopted. Petri nets can be used in the design of both the discrete event and interaction modules. Several approaches try to combine these two frameworks into one unified scheme of extended Petri nets able to handle mixed discrete/continuous behaviour.

Andreu et al. [1995] addressed the interaction and co-ordination between a Petri net model of the discrete part of a system and a continuous model consisting of a set of differential algebraic equations. They did not try to integrate the continuous aspect within the framework of Petri net theory. The Petri net model of discrete section was broken down into two parts: the reference model and the

control model. The interaction was established by running another event generator parallel to the control module and reference model. A Petri net monitors a set of differential algebraic equations, and has been tested on a multipurpose plant with 26 units and two products in the food industry, for supervisory control and validation of scheduling policies.

In another project, Srinivasan and Venkatasubramanian [1996] used Petri net digraph models for automating HAZOP analysis of batch process plants. High-level Petri nets with timed transitions and coloured tokens are used to represent the characteristics of batch operation such as operating procedures and operator actions in plant operation as well as the discrete event character of batch processes. Subtask digraphs are used to represent the causal relationships between process variables at each stage of the batch recipe. The salient aspect for this system is that it can deal with process specific information, and generic information on process components which are common for many batch process plants. But it was limited to simple process units and didn't take the control loops and interlocks into account. It also generated a large number of unrealisable hazardous consequences compared to the Hazop expert team. Subsequently, Srinivasan and Venkat Venkatasubramanian [1998] developed Batch HAZOPExpert in G2. The knowledge about task and sub-tasks in a batch process are modelled hierarchically using high level Petri nets. Cause and effect relationships between process variables within a subtask are represented using subtask digraphs. Petri nets and subtask digraphs interact with each other in a two-tier organization to model the behaviour of batch processes. The salient aspects for this framework are 1) both the continuous and discrete nature of batch operation are represented explicitly, 2) operator actions and errors are modelled. This knowledge based framework for automating HAZOP analysis for batch chemical processes was applied to a process used for manufacturing a drug. However, this system is limited to process units and subtasks for which models have been developed. It still can't deal with control loops and interlocks.

4.2.3 Discussion

Petri nets are a mature technology for representing sequences of events in the context of simulating discrete event systems. It is worthwhile to consider whether Petri nets are a suitable representation for the operating instructions in batch processes. In order to use them, we need to introduce a correspondence between the elements of a Petri net and the objects in the plant system:

Transitions correspond to actions which cause a change in the state of the plant.

Places correspond to states of the plant and its equipment.

Arcs connect places to transitions and vice-versa.

Tokens correspond to the associated state condition holding in the plant model.

In addition to the above elements (which are standard parts of the classical Petri net model), to represent operating procedures, we must label one place in the net as the “start” (s) and one as the “finish” (f). This is necessary to allow us to know when a given procedure has been completed. Given the batch plant example introduced earlier in chapter 3, we could represent its operation as the Petri net shown in Figure 4.6 (Top Level View), which is shown in the state where we are waiting for the reaction to complete. When a token reaches the place marked (f), the procedure is complete.

It is also natural to define a way of decomposing operations into smaller actions or steps so that commonly repeated operations can be modelled using a template or “model” for the operation. Therefore, using this Petri net notation, a decomposition of high level actions into sub-actions is achieved by defining “sub-nets”, each of which corresponds to a single action type and gives the detailed sequence of actions needed to complete that action.

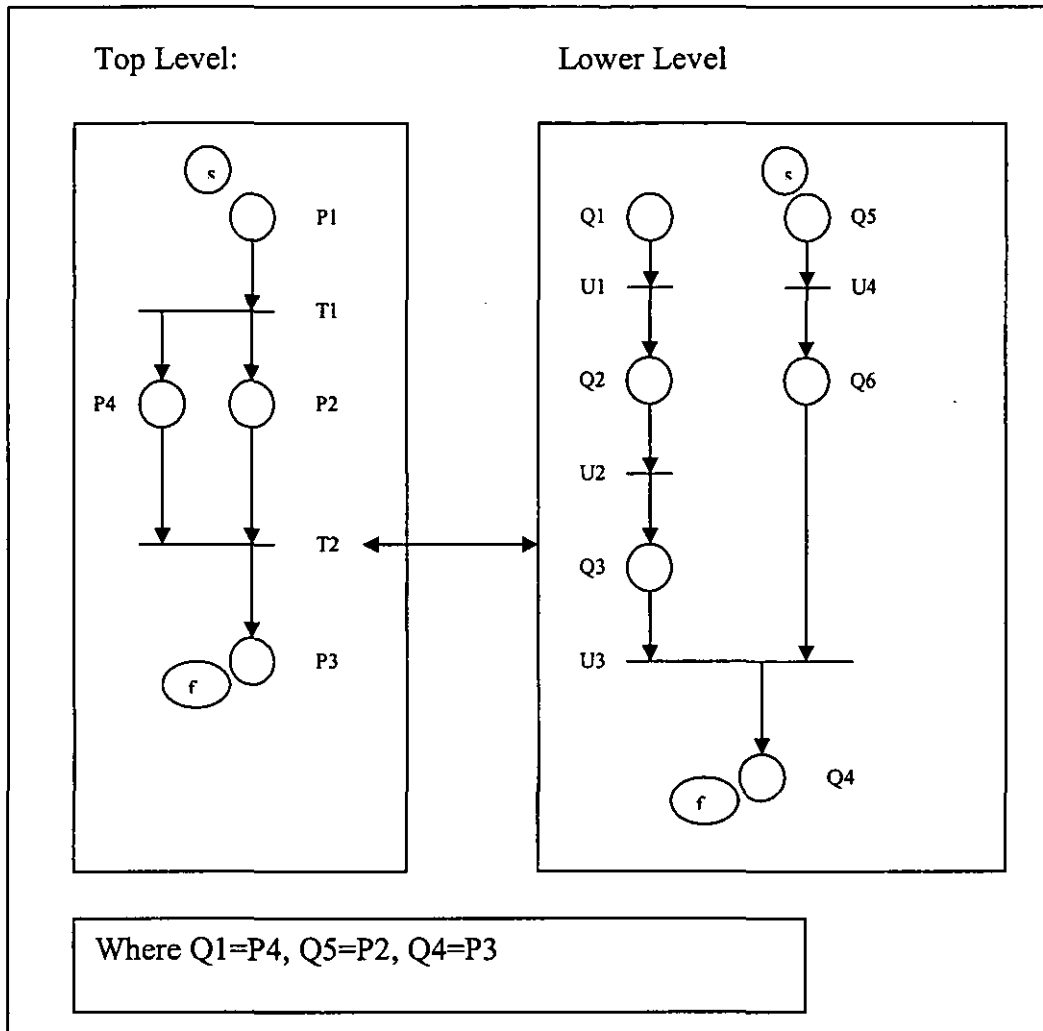


Figure 4.6 Representation using Hierarchical Petri nets

As an example, consider a refinement of the process operation shown in the Top Level View of Figure 4.6. When the transition is first started the top level net is used. However, when transition reaches P2 or P4 then the sub-net shown in the lower level view is used to determine the detailed steps required to achieve the operation. Using this technique, a Petri net can be seen as a hierarchical structure, where some operations are achieved by sub-nets, which hide details from the top level view of the Petri net transitions.

Petri Nets model the inherent sequence/parallelism in a fully formed plan very well. They are therefore quite an attractive model for visualising the operations being modelled. However, Petri nets do not allow a flexible enough

representation of the “alternative plans” which arise from deviations from the intended operations of a plant. Thus, if we wish to consider what would happen if the order of two tasks were swapped, we find that it is very difficult to modify the Petri net to take account of this.

Additionally, an experiment into the use of Petri nets to model a simple batch process (making a cup of tea) demonstrated that this type of representation is not best suited to presenting operations in a simple, easy to understand way. The Petri net constructed was complex and difficult to interpret, even for such a simple “plant”, due to the inherent complexity of modelling the process at such a detailed level.

The weakness of Petri nets for this application is that they integrate the places and transitions (i.e. the states of the plant and the actions that are performed on the plant) very closely. It is therefore very difficult to reason about the effect of varying elements from either domain.

Consider what happens when some part of the plant is modified, or new equipment is added – the Petri net and the associated operating sequence are very likely to be wrong and the correction is hard to identify.

If the order of operations in the procedure is changed or new ones are added (whether intentionally or through operator error), the effects on the Petri net are likely to be complex to predict, effectively meaning that a new Petri net needs to be constructed from scratch.

Given these observations, it is best to use a representation which de-couples the operations from the equipment in a plant, so that variations in either domain can be considered more clearly.

Therefore, there are a number of problems with using Petri nets for this type of plant system, even without considering the issue of how to interface the Petri net representation to a potentially continuous plant model and its continuous feedback control loops. Even with the visual appeal of Petri net for simpler systems, these difficulties make Petri net unsuitable for the application under consideration.

For the purposes of our hazard identification system, we aim to model all the possible operating sequences of the plant by using local constraints between actions – without having to commit to a particular complete plan. This method can be used to determine if a given whole plan satisfies the constraints, or (ultimately) to generate an optimal plan sequence for the operation from a number of actions specified.

Some final remarks on Petri nets serve to illustrate the range of problems which must be tackled when modelling batch processes accurately:

- 1) Petri nets are not able to represent how the process parameters such as flow, temperature, and level propagate within the system.
- 2) In batch hazard identification, Petri nets are not likely to be able to represent failure mode, for example, when a failure state occurs, how the operator reacts. Petri nets also seem not to be able to represent the unhealthy states when a fault occurs.
- 3) Petri nets are not able to reason between cause(s) and consequence(s), for example, what will happen if an event takes longer or less time than usual, or what will happen if some of events are missed or swapped.
- 4) Petri nets are also unlikely to represent the control loop with continuous behaviour which appears to be very common in batch processing.
- 5) Using Petri nets, the modelling suffers from too much complexity. In chapter 3, even a small case study needs a fairly complicated representation.
- 6) Using Petri nets, the models tend to oversimplify the real scenarios. Thus making it difficult to model some real world situations.

4.3 Qualitative Reasoning

4.3.1 Overview

Qualitative reasoning (QR) was originated by de Kleer's investigation on how qualitative and quantitative knowledge interacted in solving a subset of simple mechanics problems [Kleer, 1977]. Qualitative reasoning is motivated by two objectives [Weld and Kleer, 1990] [Williams and Kleer, 1991]:

- 1) modelling real world situations and human artefacts in order to support the reasoning activities of engineers;
- 2) imitating engineers' thinking. Since it focuses on scientific and engineering domains, it is often also known by another name: qualitative physics.

Qualitative reasoning can be used to model physical systems with incomplete information from which engineers can obtain useful information without differential equations. In our daily lives, we can usually figure out what is happening around us and how we will be affected with less precise and complete information. We know the basic relations between the variables in a system. Qualitative models can be used to capture and simulate the incomplete knowledge in a model to obtain a rough outline of the system behaviour. Furthermore, as more information about the system becomes available, a more accurate description can be provided.

There are mainly three ontologies in QR approach, relating to the main emphasis of modelling, i.e., device, process and constraint ontologies [Werthner, 1994]. The most commonly used ontologies are the device ontology [Kleer and Brown, 1984] and the process ontology [Forbus, 1984]. The device ontology is motivated by network theory and system dynamics. It interprets physical systems as networks of devices whose interactions are through a fixed set of units. The process ontology interprets that the change of the system is caused by process changes, corresponding to real world phenomena. Another approach, constraint ontology refers to the mathematical description of dynamic

systems in the form of qualitative differential equations [Kleer and Brown, 1984].

Some distinguishable remarks in qualitative reasoning are as discussed as followed:

Compositionality: One goal of qualitative physics is to formalize the modelling process itself. Compositionality concerns the ability to combine representations for different aspects of a phenomenon or system to create a representation of the phenomenon or system as a whole.

Resolution: The level of information detail in a representation. One goal of qualitative reasoning is to understand how little information is needed to draw useful conclusions. High resolution draws more precise conclusions while low resolution reveals what the interesting questions are. The conclusions drawn from low resolution information usually suffer from the problem of ambiguity.

Qualitative representation: Qualitative representation - what to represent, and how to represent it qualitatively is a core issue in QR, since it is used to draw the conclusion desired. The first step in qualitative representation is to indicate a quantity by whether or not it is “normal” [Abbot, 1988]. This is useful for certain diagnosis and monitoring tasks because it is can express the difference between something working and not working. The sign algebra is used to represent continuous parameters as either -, + or 0 corresponding to quantitative values which are negative, positive, or zero. Ordinal relations or the quantity space [Forbus, 1984] supports qualitative reasoning about dynamics. Landmark values [Kuipers, 1986] are constant points of comparison introduced where the qualitative value of a variable changes. Finite algebras have also been used based on a finite set of labels, i.e., very small, small, normal, large, very large.

Qualitative state: A set of propositions that characterize a qualitatively distinct behaviour of a system [Forbus, 1996]. A qualitative state describing a falling ball, for instance, would include information about what physical processes are occurring (e.g., motion downwards, acceleration due to gravity) and how the

parameters of the ball are changing (e.g., its position is getting lower and its downward velocity is getting larger). A qualitative state can abstractly represent an infinite number of quantitative states. Although the position and velocity of the ball are different at each distinct moment during its fall, the qualitative state of its motion is unchanged until the ball collides with the ground. The time over which the state of the ball falling holds is thought of as an interval, ending when the ball collides with the ground. The collision with the ground can be represented via a transition between two states. A collection of such qualitative states and transitions is called an envisionment [Kleer, 1977]. Many dynamical conclusions including the discovery of new landmark values can be drawn from an envisionment.

Time: Time is very important in QR because significant change of the states evolves along with the time. However, significant landmarks which decide these changes are not predetermined. They are discovered via simulation. New landmark values then modify the qualitative sets of variables which further decide when and what change in states will happen.

Qualitative Simulation: The purpose of qualitative simulation (e.g., QSIM [Kuipers, 1986, 1994]) is to derive the behaviour of a dynamic system with weak information about it. Physical systems are modelled with qualitative differential equations. Qualitative simulation requires neither a complete structural description of the physical system nor a fully specified initial state. The major strength of qualitative simulation is the prediction of all physically possible behaviours derivable from this incomplete knowledge. In engineering [Forbus and Falkenhainer, 1990], qualitative simulation is mainly used for monitoring and diagnosis.

Non-function-in-structure: A basic component of a device should not be a function of the entire device, i.e., the effects of the basic component are local, and should not refer to any other component.

Causality: what events can cause what other events. There are three relationships between them: 1) no relationship; 2) one event causes another event only; 3) one event causes another event via a causal link.

4.3.2 Related Applications

4.3.2.1 Computer aided Hazop

Catino et al. [Catino, Grantham et al., 1991] from Pennsylvania University adopted a process approach to plant modelling which is based on the Qualitative Process Theory of Forbus [Forbus, 1984]. A set of constraints between qualitative variables can be generated, as SDG models or QSIM (Qualitative Simulation) constraints, by the process model so as to determine the state of the plant and its possible behaviours. This approach is powerful in that it supports automatic generation of the processes to suit the different states of the plant. However, it suffers the problem of increased computational complexity because there are many variables to be processed and prediction of the plant behaviour is often ambiguous.

4.3.2.2 Planning

Qualitative physics is used to provide predictions with incomplete information and to determine what methods might achieve a desired effect. This makes it reasonable that qualitative reasoning could be carried out entirely in a planner, by compiling the domain theory and physics into operators and inference rules [Hogge, 1987]. Another different approach is to treat actions as another kind of state transition in qualitative simulation [Forbus, 1989]. This can be effective if qualitative reasoning is interleaved with execution monitoring [Drabble, 1993], or if used with a mixture of backward and forward reasoning with partial states [Coste, 1994].

4.3.2.3 Monitoring and diagnosis

Monitoring a system requires summarizing its behaviour at a level of description that is useful for taking action. Diagnosis requires isolating the

causes of a problem using incomplete knowledge about which particular parts have failed. Qualitative models provide sufficient resolution and the framework for fault isolation and detection.

Operative diagnosis tasks are those where the system being monitored must continue being operated in spite of faults. One example of operative diagnosis is diagnosing engine trouble in civilian commercial aircraft. FaultFinder [Abbott, Schutte et al., 1987] is intended to detect engine trouble and provide easily understood advice to pilots. FaultFinder compares engine data with a numerical simulation to detect the commencement of a problem. A causal model, using low resolution qualitative information (“working” or “not working”) is used to construct failure hypotheses, to be communicated to the pilot in a combination of natural language and graphics.

QR is also applied in the safety area of Process Engineering [McCoy, et al., 1999] where the most common applications are diagnosis of faults in operating plants and identification of potential hazards in a plant design. The frameworks adopted by the researches are either component based or process based approach where rules and causal links between variables are added.

4.3.3 Discussion

One of the major problems of QR has been the control of ambiguity in the predictions produced by its models. Many simple arithmetic operations such as addition are entirely ambiguous when transposed into the qualitative domain. This type of ambiguity results in a severely branching tree of predicted behaviours, and seriously limits the size of models whose behaviour can be simulated – and presented to a user in an intelligible way.

For this reason, we chose to develop a more strongly object-oriented, state-based, component-centred approach to system modelling, in which numerical quantities could be used as well as supporting qualitative reasoning in the shape of local constraints between objects considered to be physically connected.

Ambiguity of behaviour will doubtless remain within this type of model, but we hope that it will be better controlled.

4.4 AI Planning

4.4.1 Overview

Planning is designing the behaviour of some entity, either an individual, a group, or an organization [Rich, 1991]. The output of the design is called a plan. Automating planning is motivated by two reasons:

- (1) it might cast light on how people design their behaviour
- (2) complex planning problems might be solved better with the aid of computers.

There are a wide variety of planning problems differentiated by the types of their inputs and outputs. Typically, planning problems get more and more difficult as more flexible inputs are allowed and fewer constraints on the output are required. The classical approach to planning problems is to start from specifications of the effects of actions, and then try to infer a set of actions that bring about a particular state of interaction.

When planning research started in the 1960s, it was mainly seen as an application of two standard AI techniques: search and theorem proving. Search was seen as crucial to AI from the beginning until today. Many problems can be solved by applying a sequence of transformations starting from an initial problem state. At each step, there is usually a choice of which transformation to apply, most of which won't eventually lead to a complete solution, so it's necessary to keep track of partial solution states and return or backtrack to them when previous choices don't work out. The 'transformations' are usually called operators.

The classical planning approach has the following assumptions:

- Assume plans are sequences of actions;

- Take the purpose of a plan to bring about a situation satisfying a description;
- Treat the outcome of every action as perfectly predictable.

The assumptions are quite reasonable in various applications. For example, planning a route through a city can be thought of as finding a series of blocks to traverse. It is not in fact perfectly predictable that the attempt to traverse a block will get you from one intersection to the next, but in most cases it is reasonable to treat it as predictable and worry about untraversable blocks when they are encountered.

The other strand that led through classical planning was the reduction of planning to theorem proving. In 1960, John McCarthy, proposed the use of predicate-calculus reasoning to guide intelligent behaviour, and the first big realization of this idea was Green's [Green, 1969] program QA3, which solved a variety of simple problems expressed in predicate calculus. A set of planning problems expressed in terms of McCarthy's situation calculus in which axioms about what actions led to what situations were used to deduce action sequences. However, this proposal is hard to solve the problem since there would be a lot of situations generated most of which are not what we are interested in. On the other hand, GPS (General Problem Solver) [Newell and Simon, 1963] had the flaw that in expressing a new class of problems, it required the representation of not just the legal operators, but also domain dependent procedures for matching search states, i.e., define all the possible states based on domain knowledge. An "operator-difference table" that recorded which operators were relevant to reducing the differences was found by the matcher. Creating all these procedures and tables was tedious, and often seemed to amount to giving the program too many hints.

In 1969, the AI group at Stanford Research Institute in Palo Alto, California, found a way to get the best of both approaches while avoiding many of the weaknesses. This group [Fikes and Nilsson, 1971] devised a version of GPS that worked directly from action definitions stated in a form similar to that of the

situation calculus. Each action was defined in terms of its preconditions and effects, stated as predicate-calculus atomic formulas. The action definition was used to edit descriptions of situations instead of deducing properties of situations. An action's effects were of two varieties: additions and deletions. Generating a new situation description from an old one was a matter of deleting all the atomic formulas in the delete list and adding all the ones in the add list. All other formulas in the old situation description were carried over. This problem solver was called STRIPS ("Stanford Research Institute Problem Solver"). It was able to solve bigger problems than previous approaches, and was used as the planner for the Shakey robot [Fikes and Nilsson, 1971]. STRIPS remains influential, especially for the ideas it embodies about representation and temporal change.

Given a set of subgoals, a non-interleaved or linear planner can find plans to solve each subgoal, but then it can only combine them by placing all the steps for one subplan before or after all the steps of the others. Many early planners of the 1970s were non-interleaved, and thus were incomplete – they could not always find a solution when one existed. HACKER, introduced [Sussman, 1975] the idea of protecting subgoals. The first true nonlinear (partial order) planner, though, was NOAH [Sacerdoti, 1975] which was further improved upon by the program Nonlin [Tate, 1977]. These programs explored a search space of partial plans, collections of plan steps that achieved some of the goals in the problem statement. Each plan step referred to a single action that would be part of the final plan. Actions have preconditions, which would become new subgoals to be achieved. Taking a step in the search space meant committing to achieving subgoals with a new or existing plan step. In these planners, the plan steps did not have to be kept in a linear order, and thus they have often been referred to as "nonlinear". Nowadays they are more likely to be called partial order planners. Subsequent planning systems, such as TWEAK [Chapman, 1987] used constraint posting as a central technique.

One important research strand was the idea of goal regression. Given a goal that must be achieved at a point in a partial plan, what must be true before a previous action for it to become true at the right point? For example, suppose a partial plan contains the steps Drive truck 3 to Smithville, and Put load 15 into warehouse A in Smithville, and suppose that a precondition of the latter step is that load 15 be in Smithville. Regressing the precondition across the truck driving step yields the new goal: Either load 15 is on truck 3 or it is already in Smithville, which must be true before the truck is driven to Smithville if that step is to result in load 15 being in the right place at the right time. This idea was first articulated in the field of program analysis and synthesis [Dijkstra, 1976]. It was applied to planning by Waldinger [Waldinger, 1977] and Warren [Warren, 1974] in the mid-seventies, and formalized by Pednault in the eighties [Pednault, 1989]. Their systems searched a space of partial plans that are totally ordered throughout. However, total ordering does not prevent the insertion of new steps (e.g., Put load 15 onto truck 3) between existing steps.

In the late eighties, McAllester and Rosenblitt [McAllester and Rosenblitt, 1991] proved the completeness of a partial order planning algorithm which is now called SNLP. It developed from partial order planners like SIPE and Nonlin. McAllester and Rosenblitt's algorithm uses only a basic STRIPS style representation of actions. The output of the algorithm is a totally ordered sequence of actions, but it produces them by working through a search space of partial plans, each represented as a collection of four things:

- A partially ordered set of steps;
- A set of precondition goals associated with each step, which were conditions to be made true before that step in every totally ordered completion of the partial plan;
- A set of causal links that commit one step to achieving a precondition of another;
- A set of separation links that commit a step to be ordered so that it cannot interfere with a causal link.

The operators in this search space add steps and links until a plan is reached that has no unachieved preconditions.

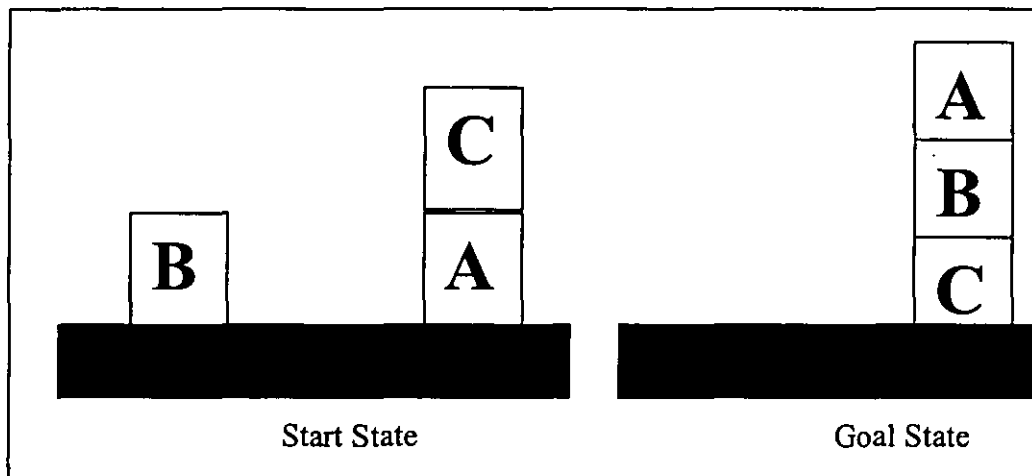


Figure 4.7 An example planning problem

A typical planning example, known as the Sussman anomaly, is shown in Figure 4.7. It can be explained as follows.

Start state: There are three blocks labelled A, B, and C where the preconditions are $\text{Clear}(B)$, $\text{Clear}(C)$ and $\text{on}(C A)$, i.e. blocks B and C are clear, and block C is on block A.

Operators: An action an agent might take is “Move a block from the table to the top of another block”. The operator can be generalised as $\text{stack}(?x ?y)$ where $?x$ and $?y$ are variables. The preconditions of this action are $\text{clear}(?x)$ and $\text{clear}(?y)$. The postcondition is $\text{on}(?x ?y)$ and the condition $\text{clear}(?y)$ is removed. Another action is “Move a block which is currently on top of another block to the table.” The operator can be generalised as $\text{unstuck}(?x ?y)$. The precondition of this action is $\text{clear}(?x)$ and the postconditions are $\text{clear}(?y)$ and $\text{on}(?x \text{ table})$.

Goal-state description: The desired situation is block A on top of block B and block B on top of block C, i.e. $\text{on}(A B)$ and $\text{on}(B C)$.

Planning: To achieve $\text{on}(A B)$, the sub-plan is $\text{unstuck}(C A)$ followed by $\text{stack}(A B)$; to achieve $\text{on}(B C)$, the sub-plan consists of only one action which is $\text{stack}(B C)$. However, given the initial state no matter how the two sub-plans

are order one before the other the goal state cannot be reached because of the conflict in preconditions and postconditions. A feasible plan can only be achieved by inserting the sub-plan $stack(B\ C)$ in between the two actions of the other sub-plan, which results in $unstuck(C\ A)$, $stack(B\ C)$ and $stack(B\ A)$. This example shows clearly that to achieve an overall goal, it is not sufficient just to consider how each sub-goal is achieved and then ordering the sub-plans; this is because the sub-plans may have to be interleaved.

4.4.2 Related Applications

The past researches just discussed concentrated in the field of planning theory. It seems that AI planning has reached a significant maturity in which the specialised techniques it offers are being successfully applied to real word problems [Aylett, Petley et al. 2000]. The approach for dealing with the complexity of general purpose planning is to specialize the domain still further, and to try to exploit restrictions that may arise. It attempts in expert systems to solve particular planning problems in very particular domains.

One such case is that of planning plant operating procedures for chemical plant [Aylett, Soutter et al., 2000]. It's interesting to think about the relationship of this problem to classical planning. It obeys the classical assumption (e.g., the assumption that the world is passive and perfectly known), but would be difficult to translate into STRIPS style add lists and delete lists. The translation would probably involve an exponentially increasing number of action definition rules. Thus, Least commitment planning and Hierarchical task networks were used to reduce search spaces [Aylett, Petley et al. 2000].

A similar situation is scheduling, which aims to finding a good order to perform a series of known tasks. Scheduling problems arise repeatedly in industry. Scheduling problems come in many different forms. They differ in the ways that tasks consume resources. For example, in job shop problems a task will require a machine, which it releases at the end, while in transportation problems, fuel can be consumed at a rate independent of the rate at which it is replenished.

Problems also differ in the kind of ordering constraints they allow for and they differ in how free the scheduler is to change steps, e.g., if each task must be executed in a different location, then changing them consequently changes the total travel time of the schedule. Because of all this variety, it is impossible to work out a single general purpose scheduling algorithm that works well in all cases. Each problem must be approached on its own, and its solution almost always requires the use of heuristics. In short, it is an excellent field for the application of tools from the AI toolkit.

It is usually fairly easy to find a feasible schedule, that is, one that does not violate any ordering or resource-bound constraints. Then one can focus on ways to improve it. It is not usually necessary to get all the way to optimality in order for the effort to be worthwhile.

4.4.3 Discussion

AI planning is able to choose and order the sequence of actions needed to achieve a set of objectives and help detect and resolve conflicts between the steps needed to achieve different objectives. Hence, AI planning has its potential ability to solve the real problem, such as safety analysis of batch process plants.

4.5 Conclusions

Some of the important messages of this work so far are:

Concentration on Petri Nets technology for representing operating plans is inappropriate for this domain. Petri Nets are not sufficiently flexible to model the effect of operations on a state-based plant model where either the plant or the instructions are subject to variation.

Modeling actions in a plant with simple STRIPS operators is insufficient, because the problem of determining if an action will succeed is a non-local search, initiated at runtime, wherever flow is concerned.

The connectivity of a plant may be determined dynamically by run-time conditions during operation of the plant. In domains such as the example of making a cup of tea, connections may be contingent on spatial relationships between equipment items (e.g. the kettle must be above the cup in order to pour water from the kettle to the cup).

Chapter 5

System Design

5.1 State-based Approach

The proposed system is modelled and implemented using a state-based approach as the state-based approach is very useful for capturing clear, concise, and unambiguous specifications [Bowen, 2005]. To apply the state-based approach the following tasks must be completed:

- Model the state of each unit in the system;
- Write down the state invariant, i.e., all the reasonable conditions describing the relationships between parts of the state;
- Specify the precondition of each operation which must be true before an operation can be carried out;
- Give appropriate initial values to the state of each unit.

The state of a unit is a collection of the essential attributes, e.g., temperature, level and pressure. Each of the state attributes should satisfy the state invariant. The operations are a set of sequential actions that will be applied to the plant for the purposes of start-up, shutdown, etc. Each time when an action is executed, it will transform the state of the plant from its current state (the state before the action) to the goal state (the desired state after the action). At the beginning, each component in the plant needs to be given an initial valid value. The precondition of an operation is the state of the component(s) under which the action can be applied successfully. If the precondition is met then the operation will be applied and the plant will take on the new state. Otherwise, the action will not be applied then the whole operation fails and the plant will not reach its goal state.

In the coffee making example given in chapter 3, the kettle has the state *level*, which is a non-negative integer type. The state *level* should satisfy the invariants at the beginning, i.e., the level should not exceed the maximum capacity of the kettle, $0 < \text{level} < \text{max}$ before the operation starts. When the kettle is moved under the kitchen tap and the kitchen tap is open, the level of the kettle will be incremented which indicates the kettle is being filled. The kettle is permitted to be filled by water only if the kettle is not already full. That is, before performing the action “fill”. Hence, the state before the action is $\text{level} < \text{max}$. The state after the action might be 50% of the maximum capacity subject to what the post-condition was specified for the action. If the precondition was not satisfied, for example, the initial state of the kettle had already been 100% then the action “fill water in an electrical kettle” would not have been performed successfully.

As a summary, the state-based approach focuses on the state of the components in the plant and all the required actions that will be performed. It uses the mathematical and logical values to form an abstract view of the system. In the example discussed above, the state invariants are represented with mathematical values, e.g., 50% level, while the precondition of each action is expressed as logical conditions. Therefore, this approach facilitates the modelling and simulation of process plants.

5.2 CHECKOP System architecture

The prototype CHECKOP system is an application developed by the author aiming at automating the HAZOPing chemical batch processing. It has gone through two iterations of design and implementation. The initial prototype of CHECKOP was written using the knowledge-based system toolkit CLIPS to prove the concepts. CHECKOP has also been re-designed and re-implemented in C++ to run under windows.

Figure 5.1 shows the overall system architecture of CHECKOP. The system has three main components: the *Parser*, the *Deviation Generator* and the *Simulation*

Engine (see Figure 5.1). The *Parser* reads the input files prepared by the user and converts the information into an internal form for processing by the other two components. The information provided by the user is specific to the plant that is required to be HAZOPed. One of the files gives details about the items of equipment in the plant, their connectivity and their current states. The other file contains a set of operating instructions to be applied to the plant to bring the plant from its current state to its goal state, while also achieving the production of a batch of product.

The *Deviation Generator* systematically applies the deviation guidewords – no, early and late – to the operating procedure so that the *Simulation Engine* can infer what will be the consequence if a certain instruction in the procedure is not executed, or the instruction is carried out too early or too late. Having gone through all the deviations, the *Simulation Engine* will produce a report file providing warnings against any undesirable situations that may result from the deviations. To carry out the simulation the *Simulation Engine* requires the Action Model Library which provides information about actions that can be performed on different pieces of equipment and the effects of those actions.

The pseudo code of CHECKOP based algorithm is given in the appendix at the end of this chapter.

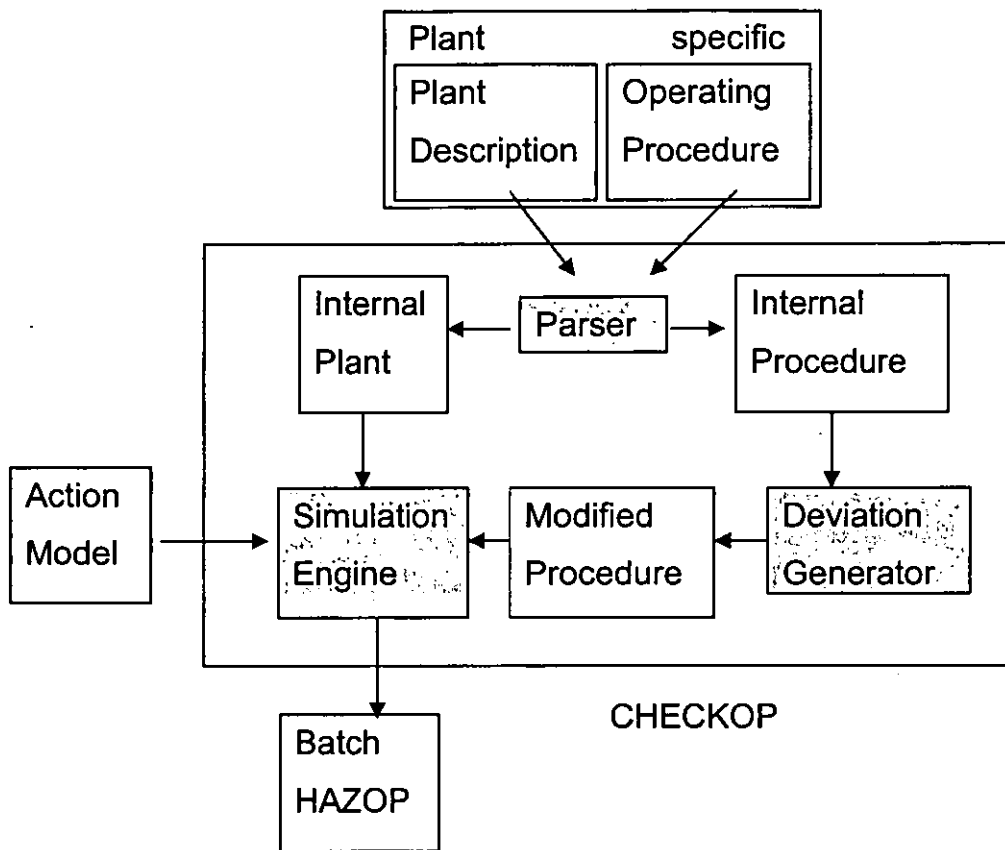


Figure 5.1 CHECKOP System Architecture

5.3 Plant Description

An object-oriented approach is used to describe the plant. Consider the batch plant as shown in Figure 5.2; each item in the plant is declared in the plant description file. For each item at least the following basic information is given:

- The type of unit it belongs to;
- Which other plant items it is connected to;
- What subunit it contains.

Other appropriate information related to a plant item will also be stored with that plant item. Table 5.1 provides some example descriptions of the plant items found in figure 5.2.

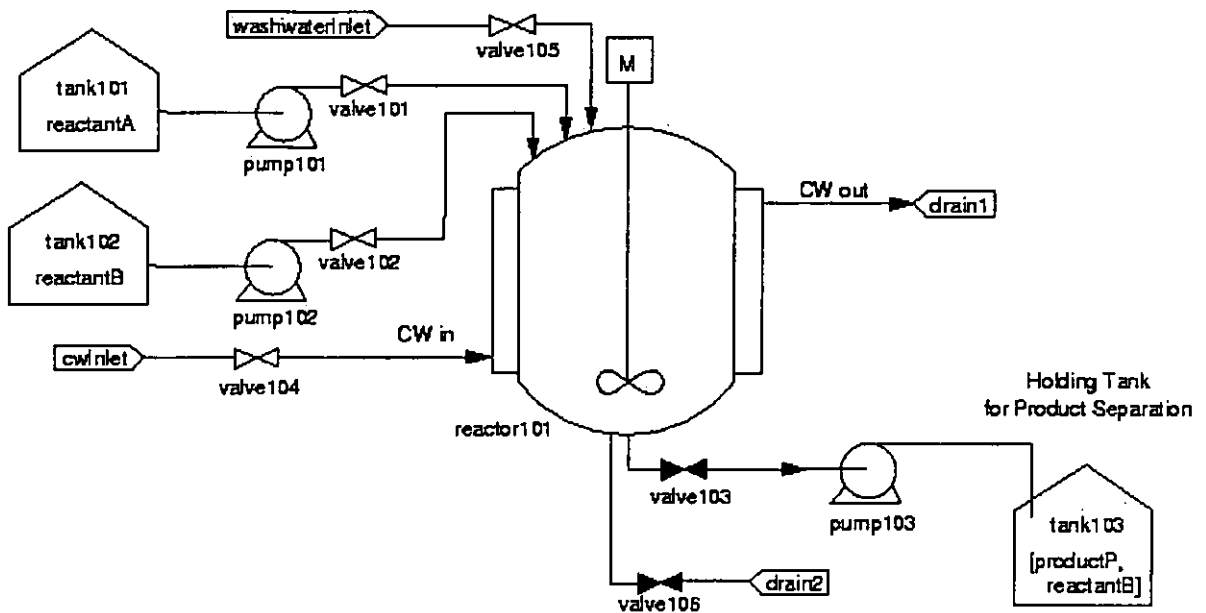


Figure 5.2 A Simple Batch Plant

Formal plant item declaration	Explanation
<pre>instance(tank101 isa tank, [content info [reactantA], outports info [out is [pump101,in]]]).</pre>	<p>Tank101 is a tank</p> <p>The content of the tank is reactantA</p> <p>The outlet of the tank is connected to the inlet of pump101</p>
<pre>instance(pump101 isa pump, [status is offline, outports info [out is [valve101,in]]]).</pre>	<p>Pump101 is a pump</p> <p>The status of the pump is off-line</p> <p>The outlet of the pump is connected to the inlet of valve101</p>
<pre>instance(valve101 isa valve, [status is closed, outports info [out is [reactor101, in2]]]).</pre>	<p>Valve101 is a valve</p> <p>The status of the valve is closed</p> <p>The outlet of the valve is connected to inlet 2 of reactor101</p>
<pre>instance(reactor101 isa stirred_tank_reactor,</pre>	<p>Reactor101 is a stirred-tank-reactor</p>

<pre>[outports info [out1 is [valve103,in], out2 is [valve106,in]], heatSink info [hout is [jacket101,hin]], reaction info [reaction_ab_p]].</pre>	<p>The outlet 1 of the reactor is connected to the inlet of valve 101 and outlet 2 is connected to valve 106</p> <p>The heat of the reactor is transferred to jacket 101</p> <p>The intended reaction is called reaction_ab_p</p>
--	---

Table 5.1 Explanation of Simple Plant Description

5.4 Operating Procedure Description

In order for the CHECKOP system to analyse an operating procedure, the instructions have to be written following the templates. In general, instructions that are written in natural language style are difficult for the computer to understand and their meaning may also be ambiguous. Therefore, to avoid natural language processing, an operating procedure written as input to CHECKOP strictly follows the templates below:

Template 1: *Item Action*

Example: valve101 open

Template 2: *Item Action until Condition*

Example: mixer on until elapsed-time 20 minutes

Template 3: *Item1 Action Item2 Filler-word Fluid until Condition*

Example: reactor101 fill-from tank101 with reactantA until volume 30 percent

The duration for some other actions could be very critical, for example, react the content in reactor 101 for 35 minutes. If the action is executed less than the

specified time then the reaction may not be have completed and therefore the operator may not get the right material and the right quantity of the final product. If the action is executed longer than the specified time then the product may also be ruined or the plant may reach a dangerous state. For these kinds of critical instruction, either template 2 or template 3 is used to represent the condition. For example, the action “react the content in reactor 101 for 35 minutes” would be represented using template 2 as “reactor101 react_content until elapsed_time 35 minutes” since it involves only one plant item.

Given the plant shown in figure 5.2, the instructions for charging reactor101 with reactantA can be expressed as:

```
valve101 open
pump101 start
reactor101 fill-from tank101 with reactantA
until volume 30 percent
pump101 stop
valve101 close
```

The file containing the operating instructions for operating the plant is read in by CHECKOP and translated into its internal form.

5.5 The Action Model Library

Associated with each plant item type there is an action model in the *Action Model Library*. The model specifies the operations that can be carried out on that type of plant item.

An action model first translates all templates (described in section 5.3) into a common format:

Action (Item1, Modifier)

where

- *Action* refers to action keywords such as *fill-from*, *open*, *close*
- *Item1* refers to units which will perform the action, e.g., reactor 101
- The fields *Action* and *Item1* are compulsory
- *Modifier* can be *null* or not *null*.

When the field *Modifier* is *null*, the template for such action model becomes

Action (Item1)

Otherwise

Modifier will be further extended into:

[Item2, Fluid, Condition]

where

Item2 refers to the units where the action performs on, e.g., tank101

Fluid refers to the material performed by the action, e.g., *reactantA*

Condition refers to the termination condition of the action. It can be a time termination or a post-condition as the result of the action, an example is *elapsed-time 20 minutes*, and another example is *volume 30 percent*.

For each action the pre-conditions that must be true before the action and the post-conditions after the action are stated. For example, the actions for a valve can be *open* or *close*. In general, there is no pre-condition for opening or closing a valve. However, the post-condition for opening a valve is that a flow path exists between the upstream unit and the downstream unit. The post-condition for closing a valve is that the flow path between the upstream unit and the downstream unit no longer exists.

The actions for a pump can be *start* or *stop*. To start a pump, the pre-conditions are that there must be a flow path between the source of a fluid and the pump and there must be a flow path between the pump and the sink. If the pre-conditions are not met then *start* operation will generate a warning message. The post-condition of starting a pump is that there is a flow between the source and the sink. On the other hand, there is no pre-condition for stopping a pump and the post-condition of stopping a pump is that there is no flow between the source and the sink.

5.6 The Deviation Generator

The Deviation Generator applies the guide words *no*, *early* and *late* systematically to the operating instructions to generate different versions of the operating procedure. This allows CHECKOP to explore the consequences of different scenarios that could result from operator human errors.

By applying the guideword *no* to the following example procedure:

- (1) valve101 open
- (2) pump101 start
- (3) reactor101 fill-from tank101 with reactantA until volume 30 percent
- (4) pump101 stop
- (5) valve101 close

the Deviation Generator will remove systematically one instruction at a time from the procedure, which will result in five different procedures. Each representing an error of omission, i.e. an operator failed, or forgot, to carry out a specified instruction.

For example, procedure with instruction 1 omitted:

- (2) pump101 start
- (3) reactor101 fill-from tank101 with reactantA until volume 30 percent
- (4) pump101 stop
- (5) valve101 close

Procedure with instruction 2 omitted:

- (1) valve101 open
- (3) reactor101 fill-from tank101 with reactantA until volume 30 percent
- (4) pump101 stop

(5) valve101 close

When the guide word *early* is applied to the procedure, instructions are moved earlier in the procedure. For example, moving the instruction “reactor101 fill-from tank101 with reactantA until volume 30 percent” two steps forward will result in the procedure:

(3) reactor101 fill-from tank101 with reactantA until
volume 30 percent

(1) valve101 open

(2) pump101 start

(4) pump101 stop

(5) valve101 close

All the different procedures generated by the Deviation Generator are passed to the Simulation Engine for analysis to identify operability problems and potential hazardous situations.

5.7 Simulation Engine

The heart of the CHECKOP system is the simulation engine. Given an operating procedure, it applies the instructions one at a time and simulates its effect by changing the state of the plant. Therefore, the plant moves from one state to another until all the instructions are completed. However, the execution of a procedure may not always reach its end. This is because when the simulation engine detects an operability problem or hazardous situation it will report to the user. Consider the procedure given earlier in this section where the instruction “valve101 open” was missing, the following warning will be generated: there is no flow path between tank101 and reactor101 for filling. The simulation engine identifies that the fill action cannot be completed (consequence) because no flow path exists between the source and the sink (due to the instruction “valve101 open” is missing).

The simulation engine will work systematically through all the procedures generated by the deviation generator. It can 'propagate' the causal relationship within the system in the sense that, for example, when no flow path between tank101 and reactor101 causes reactor101 not to be filled with the intended material from tank101, the simulation engine will then detect it is impossible to execute the action 'reactor101 react_content until elapsed_time 35 minutes' because not all the required materials are present in the reactor. Therefore, an effect of a previous step is "propagated" and considered in future steps.

The simulation engine is used together within an action model. Consider its usage in the fill_from action model, i.e. *Item1.fill_from(item2, material, intention)*:

If a flowpath does not exist between item1 and item2

then report consequence.

If the input volume plus the current volume in item1 is greater than the capacity of item1

then report consequence.

If the material in item2 is not the intended material

then report consequence.

If there is already material in item1

Then look up reaction model to check if the existing material and the input material will create a hazardous reaction

If yes then report hazard.

.....

The above algorithm shows how the simulation engine is integrated with the different models such as reaction model and flow path model to identify hazards. It applies the rules from the different models whenever applicable. These rules may or may not make changes to the states of the equipment units. The simulation engine can detect hazards as the direct result of an action or the result of a sequence of actions because of the state changes in the plant.

5.8 Flow Path Analysis

CHECKOP solves one of the typical problems encountered in this and another related area – Operating Procedure Synthesis (OPS). The problem is that of how to determine the effect on flow due to actions involving opening or closing a valve. For example, if two valves are present in sequence in the same line, then opening one of them will not produce a flow through it if the other valve is closed. Similarly, if the two valves are in parallel sections, then closing one will not necessarily prevent fluid from flowing. This flow modeling problem cannot be solved ahead of time and must be found during a run-time simulation of the system. This means that simple STRIPS-style operators (with associated lists of preconditions and effects) are inadequate for modeling the effects of actions in this domain, if the effects to be modeled include the facts of flow existing at different places in the plant.

In its use of state-based simulation and run-time search for flow path connections, CHECKOP uses the “action synergy” approach to flow modeling and generating the effects of valve operations, as also explored in the work of Soutter (1997). In OPS systems, the action synergy approach is used to find safe sequences of valve operations to achieve planning goals, given that the operations will have overlapping and perhaps conflicting effects on the flows in the plant. In CHECKOP, the aim is to simulate accurately when flows are possible and when they are not possible.

The developed flow path analysis model detects whether flow paths exist between two plant items. If yes, flow will propagate through the paths; otherwise, it will stop at where the paths are blocked. The algorithm for *flowpath_exist (equipment1, equipment2)* is:

All immediate upstream items of equipment2 are placed in a set X.

If X is empty

then no existing path

else Remove items that are in the set X which are blocked

if X is empty

then no existing path

else If any of the remaining items in set X is equipment 1

*Then a flow path exists between equipment1 and
equipment 2*

*Else for each item Y in the set X call
flowpath_exist(equipment1, Y).*

5.9 Reaction Model

In a batch process, a reaction could lead to a change of state such as the amount or heat content of the materials in an equipment unit. For example, in an action 'reactor101 react_content until elapsed_time 35 minutes', the reaction could cause the amount of both reactant A and reactant B decrease and of the final product increase. It may also generate heat and increase the temperature. When the reaction is complete the heat generation process is stopped. As mentioned above, different material will create different reactions. Therefore, reaction should be modelled individually due to its unique characteristics so that the simulation engine could identify the consequence if there is an abnormal reaction or when a reaction is terminated earlier or later in terms of clock time.

5.10 Overall Algorithm

Given the system architecture and the component description, the overall system algorithm is described below:

Load library information

Load plant input file

Load operating procedure and translate each instruction into the form

Action(object,predicate,modifier)

Enumerate and choose guidewords

Apply each guideword one by one

For each action

Choose the appropriate template

Execute action

If it is a flow related action

Then check whether the flow path exists

check whether the propagation rule is applicable

update the state of the object(s)

If it is reaction related action

Then check the reaction rule

update the state of the object

Else

update the state of the object

5.11 Summary

To automate the batch hazaop analysis, the *Parser* is used to load the input files prepared by the user and converts the plant information into an internal format for processing. The plant information includes what the units, their current states before the instructions are executed, their subunits, and their environmental units are, the plant layout, and the current states after the propagation rule is applied. The *Deviation Generator* then systematically applies the deviation guidewords such as no, early and late to the instructions

which are also parsed into an internal format to be match predefined templates which are designed in the action model. The action model can be reused for other plants or added depending on how complex the instructions are. The *Simulation Engine*, which is based on 'common sense' rule, then infers what will be the consequence if a certain instruction in the procedure is not executed, or the instruction is carried out too early or too late.

Chapter 6

Examples and Discussion

This chapter uses three examples to illustrate the HAZOP capability of CHECKOP and for the purpose of identifying the limitations of the current prototype. The first example is a tea making process which is a more complicated version than the coffee making process as discussed in Chapter 3. Two simple batch reactor plants are also used. The two batch plants, though simple, are typical and representative of many batch plants in used in industry.

The next section will describe the first example. It will begin with a brief description of the plant followed by an explanation of a procedure for achieving a particular purpose. The result of the HAZOP from CHECKOP is then presented and commented upon.

Sections 6.2 and 6.3 follow the same format for the other two examples. The chapter ends with a summary of the strengths and limitations of CHECKOP.

6.1 Tea-Making Example

6.1.1 Plant Description

The tea making example consists of the following plant items: kettle, kitchen tap, kettle base, power supply socket, tea bag tin, cup, milk bottle and spoon.

6.1.2 Procedure Description

Given the plant items, the procedure for input into CHECKOP for making a cup of tea has the following steps:

1. kettle move_under kitchentap
2. kettle open_lid
3. kitchentap turn_on
4. kettle fill_from kitchentap with water absolute amount 1 liter
5. kitchentap turn_off
6. kettle close_lid
7. kettle move_to kettlebase
8. kettle switch_on
9. kettlebase plug_to supplysocket
10. kettle switch_on
11. kettle heat_content until temperature 100 C
12. kettle switch_off
13. kettlebase plug_off supplysocket
14. tea_bag_tin move_to cup
15. tea_bag_tin open_lid
16. cup fill_from tea_bag_tin with teabag absolute amount 1 piece
17. tea_bag_tin close_lid
18. kettle move_to cup
19. cup fill_from kettle with water until level 0.2 liter
20. cup react_content until elapsed_time 3 second
21. milkbottle move_to cup
22. milkbottle open_lid
23. cup fill_from milkbottle with milk absolute amount 5 milliliter
24. milkbottle close_lid
25. spoon stir_content of cup until elapsed_time 3 seconds
26. cup react_content until elapsed_time 5 second
27. cup remove tea_bag

6.1.3 Result Analysis

Deviation: No Action	Consequence
kettle move_under kitchentap (step 1)	<ul style="list-style-type: none"> • Fill action (step 4) cannot be completed because there is no flow path between kettle and kitchentap. • Heat action (step 11) may cause over heating because there is no content in the kettle. • Fill action (step 19) cannot be completed because there is no content in the kettle. • React action (step 20) cannot be completed because current content is only tea_bag. • React action (step 26) cannot be completed because the final product is not tea.
Deviation: Early Action	Consequence
kettle close_lid (step 6) executed three steps early	<ul style="list-style-type: none"> • Fill action (step 4) cannot be completed because there is no flow path between kettle and kitchentap. • Heat action (step 11) may cause over heating because there is no content in the kettle. • Fill action (step 19) cannot be completed because there is no content in the kettle. • React action (step 20) cannot be completed because current

	<p>content is only tea_bag.</p> <ul style="list-style-type: none"> • React action (step 26) cannot be completed because the final product is not tea.
kettlebase plug_off supplysocket (step 13) executed one step early	<ul style="list-style-type: none"> • Plug_off action (step 13) should not be done when the electricity is still on.
Deviation: Late Action	Consequence
kettle open_lid (step 2) executed 3 steps late	<ul style="list-style-type: none"> • Fill action (step 4) cannot be completed because there is no flow path between kettle and kitchentap. • Heat action (step 11) may cause over heating because there is no content in the kettle. • Fill action (step 19) cannot be completed because there is no content in the kettle. • React action (step 20) cannot be completed because current content is only tea_bag. • React action (step 26) cannot be completed because the final product is not tea.

Table 6.1 Sample Output of CHECKOP for Tea Making Example

It is interesting to observe that some of the deviations generate exactly the same consequences.

6.2 Batch Reactor Example 1

6.2.1 Plant Description

This plant consists of a reactor that takes a solvent and another chemical as input. The solvent and the chemical will react together to create a desirable product. For the purpose of this example, only the feed from the solvent storage tank is included; the feed from the chemical storage tank is left out to keep the example simple. The scrubber is for cleaning out any toxic fume before releasing it to atmosphere. The plant also includes a line for transferring material to the sample point and a line for transferring material to the next vessel.

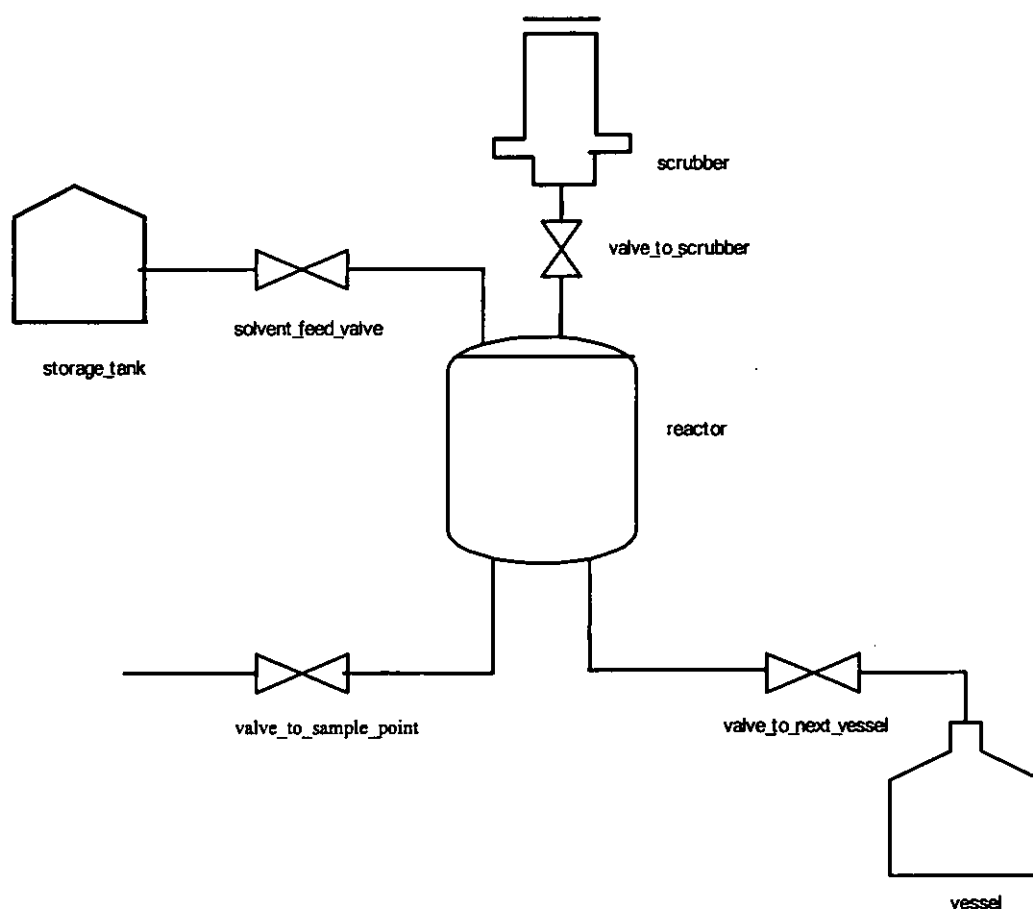


Figure 6.1 Plant Diagram for Batch Reactor Example 1

6.2.2 Procedure Description

The procedure under consideration is the one for charging the reactor with the solvent. The charging of the reactor with the solvent may create high pressure within the reactor. Therefore, it is important that the valve to the scrubber is open so that any excess pressure is vented through the scrubber. Otherwise, the build up of pressure may rupture the shell of the reactor. The solvent is a toxic material. As mentioned before, the scrubber will clean out any toxic fume before releasing it into the atmosphere.

During charging it is also important that the valves for transferring material to the sample point and to the next vessel are closed so that material will not be released unintentionally.

Given the plant and the precautions, the procedure for input into CHECKOP for charging the reactor is:

1. valve_to_scrubber open
2. valve_to_next_vessel check_closed
3. valve_to_sample_point check_closed
4. solvent_feed_valve open
5. reactor fill from storage_tank with solvent until level reaches 12 tonnes
6. solvent_feed_valve close
7. valve_to_scrubber close

6.2.3 Result Analysis

The scenarios of particular interest are the ones when any of the precautionary steps is missed out. This is done by applying the guideword “no” to the procedure so that the consequence of not carrying out any of the instructions is identified. The result from CHECKOP by applying the guideword “no” to the operating procedure is shown in table 6.2.

Deviation: No action	Consequence
valve_to_scrubber open (step 1)	<ul style="list-style-type: none"> • Fill operation (step 5) may cause over pressure in the reactor
valve_to_next_vessel check_closed (step 2)	<ul style="list-style-type: none"> • Fill operation (step 5) may not be completed because there is a flow-path out
valve_to_sample_point check_closed (step 3)	<ul style="list-style-type: none"> • Fill operation (step 5) may not be completed because there is a flow-path out
solvent_feed_valve open (step 4)	<ul style="list-style-type: none"> • Fill operation (step 5) can not be completed because there is no flow-path in
solvent_feed_valve close (step 6)	<ul style="list-style-type: none"> • Fill operation (step 5) may lead to overflow because flow-path in is not closed

Table 6.2 Sample Output of CHECKOP for Batch Reactor 1

6.3 Batch Reactor Example 2

6.3.1 Plant Description

The plant shown in figure 6.2 is a typical batch reactor. It consists of three feed lines – one for reactant A, one for reactant B and one for water. There is also a feed for cooling water for cooling down the reactor. Different valves are used to control these feeds from different tanks. The product can be transferred from the reactor to another storage tank or a drain.

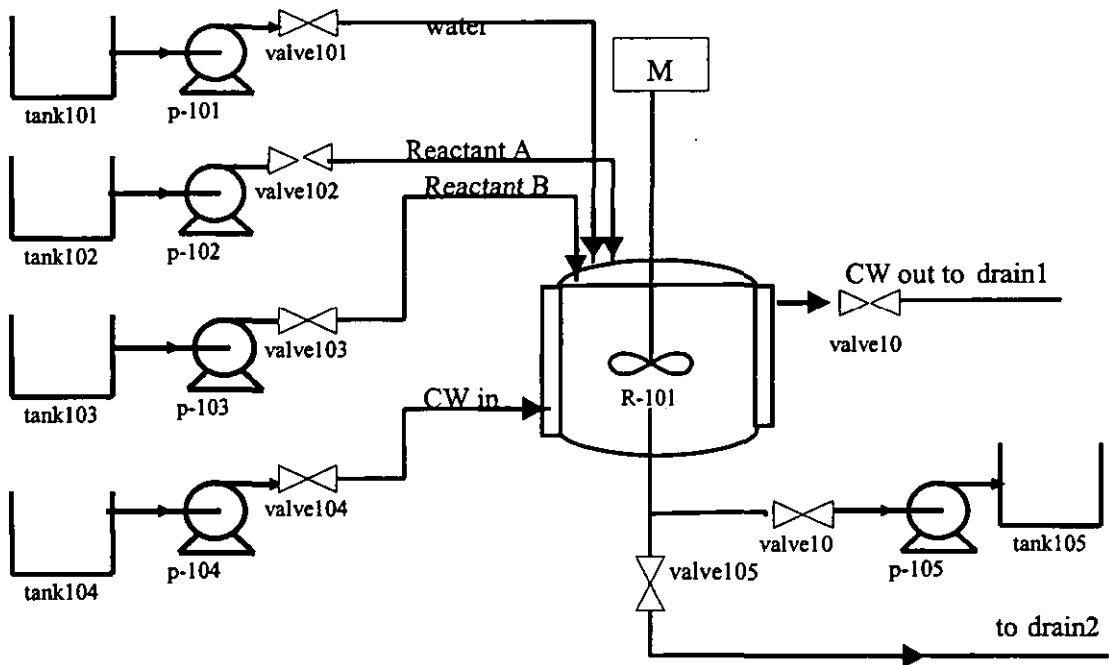


Figure 6.2 Plant Diagram for Batch Reactor Example 2

6.3.2 Procedure Description

The purpose for this procedure is to generate a product by mixing Reactant A and Reactant B. Reactor A is charged from tank 102. Reactant B is charged from tank 103.

When both materials are being mixed, there will be a reaction between them. The reaction will be accelerated by switching on the agitator. While the reaction is going on, the heat generated will be taken away by the cooling jacket. After the reaction, the final product will be charged into the storage tank. The reactor will be washed and ready for the next recipe. The procedure for input into CHECKOP is given below:

1. Valve102 open
2. Pump102 start
3. Reactor101 fill from tank102 with ReactantA until reactor101 volume 50 percent
4. Pump102 stop
5. Valve102 close
6. Agitator turn_on
7. Valve104 open
8. Pump104 start
9. Tank104 flow coolingwater to coolingjacket
10. Reactor101 cool_content until temperature 25 degree
11. Valve103 open
12. Punp103 start
13. Reactor101 fill from tank103 with reactantB until volume 60 percent
14. Pump103 stop
15. Valve103 close
16. Reactor101 react_content until 35 minutes
17. Agitator turn_off
18. Valve106 open
19. Pump105 start
20. Tank105 fill from reactor101 with productAB until reactor101 empty
21. Pump105 stop
22. Valve106 close
23. Pump104 stop
24. Valve104 close
25. Valve107 open
26. Coolingjacket flow coolingwater to drain1 until coolingjacket empty

27. Valve107 close
28. Valve101 open
29. Pump101 start
30. Reactor101 wash with washwater
31. Pump101 stop
32. Valve101 close
33. Valve105 open
34. Reactor101 flow washwater to drain2 until reactor101 empty
35. Valve105 close

6.3.3 Result Analysis

A variety of scenarios generated by are highlighted below to illustrates the capability of CHECKOP.

Deviation: No action	Consequence
valve102 open(step 1)	<ul style="list-style-type: none"> • Switching pump102 on (step 2) while valve102 is closed will result in pump102 being damaged. • Reactor101 cannot be filled from tank102 (step 3) because there is no flow path between tank102 and reactor101. • Reactor101 cannot be cooled (step 10) as it has no content. • ProductAB cannot be emptied from reactor101 (step 20) because reactor101's content is not productAB
pump104 start(step 8)	<ul style="list-style-type: none"> • Consequence1: Step (9) cooling water can't flow through cooling jacket from tank104 because there is no flow path between coolingjacket and tank104

	<ul style="list-style-type: none"> • Consequence2: Step (13) Since cooling water is not running through cooling jacket while there is a reaction between ReactantA and ReactantB, the reaction will lead to overheating in reactor101
agitator turn_on(step 6)	<ul style="list-style-type: none"> • Consequence: Since the agitator is not running while there is a reaction in step (13) between ReactantA and ReactantB, it will lead to overheating in reactor101 • Consequence: Overheating in step (16) because the content of the reactor is just ReactantB
Deviation: Early Action	Consequence
reactor101 fill_from tank102 with ReactantA until volume 50 percent(step 3)	<ul style="list-style-type: none"> • Step (3) cannot be completed because pump102 is off. • Reactor101 content cannot be cooled (step 10) as there is no content. • ProductAB cannot be emptied from reactor101 (step 20) because reactor101's content is not productAB
valve102 close(step 5)	<ul style="list-style-type: none"> • Reactor101 cannot be filled from tank102 (step 3) because there is no flow path between tank102 and reactor101. • Reactor101 content cannot be cooled (step 10) as there is no content. • ProductAB cannot be emptied from reactor101 (step 20) because reactor101's content is not productAB

Figure 6.3 Sample Output of CHECKOP for Batch Reactor 2

6.4 Summary and Discussion

The examples considered are typical in that:

- 1) Flow path due to physical connection between units is addressed in the system design because it is common in batch process. In the tea example, the connection is built up by physically moving a unit to another. In other examples, this connection is built up by the preconfigured plant layout as described in the plant description where a unit can have one or more than one incoming and/or outgoing flow path. Thus, the usage of this feature could be extensible in other batch plants.
- 2) *Action Model Library* is an extensible template for batch processes. It provides a generic but efficient means to deal with actions in different types of batch plants. For example, the action 'fill_from' template is used in all the batch examples to accomplish different intentions. More action words could be attended when more cases are studied.
- 3) Simulation engine proved to be efficient in representing the causal relationships between the state variables. It can identify the deviations due to change of state variables, which might be caused by equipment malfunction or human error, and propagate such deviations through the plant or subsequent actions.

CHECKOP has successfully identified hazards associated with operation problems. It can systematically explore the effects caused by operator human errors and automate the process of Hazop. The important keywords 'No', 'Early', and 'Late' are addressed in the CHECKOP system. They correspond to common human errors such as missing an operation, or executing an action earlier or later than expected in a sequence of actions. However, it is still limited in the following ways:

- 1) The *Parser* for inputting plant description file is still at the coding level and is very primitive. The plan to load plant description files from external sources and integrate the parser into the system.
- 2) At the moment, only three keywords are considered. The *Deviation Generator* need to be extended to address more guidewords.

- 3) The current *reaction module* is very simple and contains very little information. A more comprehensive module should be developed to deal with this important feature.
- 4) The *Action Model* needs to be extended to deal with more actions related to different types of equipment.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this chapter, there are mainly two parts of work the conclusions can be drawn from, i.e., literature review and proposed framework. The former built up the background about what previous work has been done, where the current research is, and what tools we could use to build up an adequate model to describe batch processing plants. The latter suggests a possible solution to the current problems and is expected to overcome the shortcomings of other researches in this field.

In view of the tools investigated, Petri nets help provide a way of modeling and understanding discrete systems. Petri nets can be used as a modeling tool to model actions and operations. However, Petri nets have problems in terms of handling complexity when modeling real systems. Furthermore, it cannot model the causal relationship required to do Hazop. These shortcomings limit the use of Petri nets to automated batch Hazop. Frame based system, however, is well able to carry information for reasoning. It can describe actions and support inheritance and reuse. It is also expressive and concise since each frame carries only necessary information for reasoning purpose. AI planning, on the other hand, is a tool to coordinate the sequence to reach a goal state from an initial state. In batch processing system, sequences of the actions and operations are very important since they suggest the system behavior or states in the implied temporal order. With AI planning, we can figure out a reasonable sequence for a procedure. Finally, in the literature review of the qualitative reasoning part, the concept of quantity space and landmark may be used to define the scale of the state values. Also, since QR tend to suffer from the problem of ambiguity, some

semi-quantitative simulator may be used as a filter to reduce this problem in future work.

The proposed framework provides a system architecture, a way of representing knowledge and a reasoning strategy for the automation of hazard identification of batch chemical plants. Although the chosen approach is a simple representation, it is sufficient to allow a wide range of phenomena to be modeled. Three simple examples are used to point out where we are trying to make improvement. Some suggestions have been made to address how to tackle the problems and develop an efficient modeling language for the chemical process plant. The test result shows its execution is efficient and expressive, nevertheless, the proposed framework has only been applied to these small examples. The next phase of research is to apply the created modeling language to different cases to test further its expressiveness and powerfulness.

In terms of the original objectives as described in the chapter 1 section 1.2, all of them are addressed. The knowledge required by the hazaop analysis are integrated within a novel framework with extensibility, reusability, and expressiveness. A prototype was developed and tested using three case studies. Compared with the outcome of previous work, the outcomes from this approach show that the proposed framework can be used to simulate the qualitative behaviour of batch process plants and can bring forward the opportunities of applying this approach for tackling processes with discrete behaviours.

7.2 Future Work

The work described in this thesis has demonstrated the feasibility of using the state-based simulation approach to automate batch Hazop. Future work should focus on improving the design and implementation of the prototype then testing it using more and larger batch chemical processing plants. Some areas that require attention are:

- 1) In the structural representation, a definition of how equipment items are further decomposed should be described. It is suggested that the principle of such decomposition should be based on the device ontology by de Kleer and Brown. It interprets physical systems as networks of devices whose interactions are through a fixed set of units.
- 2) Chapter 3 only discussed the effect of the actions on the system behaviour. It did not discuss what happens if there is a deviation of process variables, for example, the temperature of the water in a tank is becoming too hot. In the equipment item modelling, a slot can be added to represent the default value so that any deviation can be detected. It is suggested that a slot such as content-temperature (default cold) could be added. Once such a slot is added then any deviation from the default value can be detected and the cause identified.
- 3) Currently, the values that a state variable can take on are not clearly defined. This may cause ambiguity and the system behaviour becomes inaccurate. To tackle this problem, Forbus' quantity space may be adopted to support qualitative reasoning. Kuipers' idea about landmark values are useful to identify where the qualitative value of the variable changes. A finite set of labels such as very hot, hot, normal, cold, very cold can be used to signify and compare the variable changes. Besides, quantitative numerical information can be integrated with qualitative simulation, for example, Kuipers' semi-quantitative simulators as filters, to resolve such ambiguities in system behaviour.
- 4) Some templates for actions have been defined. The set of templates may be extended when actions from more case studies are considered. This is

necessary to allow a wide range of instructions to be expressed naturally and unambiguously.

- 5) The current reaction model is very limited and contains very little information. A more extensive reaction model will need to be developed to deal with a wide range of chemicals.

References

Abbott, K., Schutte, P., Palmer, M., and Ricks, W., 1987, *Faultfinder: A diagnostic expert system with graceful degradation for onboard aircraft application*, In: 14th International Symposium on Aircraft Integrated Monitoring Systems.

Abbott, K., 1988, *Robust Operative Diagnosis as Problem Solving in a Hypothesis Space*, Proceedings of AAAI-88, 369-374.

Andreu, D., Pascal, J.C. Pingaud, H. and Valette, R., 1994, *Batch process modelling using Petri nets*, in: Proceedings of the 1994 IEEE International Conference on Systems, Man and Cybernetics, Vol. 1, San Antonio, TX, pp.314-319.

Andreu, D., Pascal, J.C. and Valette, R., 1995, *Interaction of discrete and continuous parts of a batch process control system*, in: Proceedings of the Workshop on Analysis and Design of Event-Driven Operations in Process Systems (ADEDOPS), Imperial College, London, pp. 123-128.

Andreu, D., Pascal, J.C. and Valette, R., 1996, *Events as a key of a batch process control system*, in: Proceedings of the CESA'96 IMACS Multiconference, Symposium on Discrete Events and Manufacturing Systems, Lille, France, pp. 297-302.

Atallah, S., 1980, *Assessing and Managing Industrial Risk*, Chemical Engineering,

Aylett, R.S., Petley, G.J., Chung, P.W.H., Chen, B. and Edwards, D.W., 2000, *AI planning: solutions for real world problems*, Knowledge based systems, 13, pp61-69.

Aylett, R.S., Soutter, J., Petley, G.J., Chung, P.W.H. and Edwards, D.W., 2001, *Planning plant operating procedures for chemical plant*, Engineering Applications of Artificial Intelligence, 14, pp341-356.

Bartolozzi, V., Castiglione, L., Picciotto, A. and Galluzzo, M., 2000, *Qualitative models of equipment units and their use in automatic Hazop analysis*, Reliability Engineering and System Safety, 70, pp49-57.

Battelle, C., 1985, *Guidelines for Hazard Evaluation Procedures*, AIChE Press.

Boissel, O.R. and Kantor, J.C., 1993, *Optimal control for discrete-event systems using simulated annealing*, in: Proceedings of the 1993 American Control Conference, Vol. 3, USA, pp2533-2537.

Boissel, O.R. and Kantor, J.C., 1995, *Optimal feedback control design for discrete-event systems using simulated annealing*, Computers & Chemical Engineering 19, pp253-266.

Bowen, J.P., 2005, <http://www.sbu.ac.uk/~abdallae/units/sse/statebased.pdf>, lecture notes, accessed on 7th October, 2005

Bowen, J.P., 2005, <ftp://ftp.comlab.ox.ac.uk/pub/Zforum/faq.txt>, accessed on 9th October, 2005

Boykin, R.F. and Kazarians, M., 1987, *Quantitative Risk Assessment for Chemical Operations*. International Symposium on Preventing Major Chemical Accidents, Washington D.C., February, AIChE, p.187.

Brannegan, D.P., 1985, *Hazards Evaluation in Process Development*, Chemical Process Hazard Review, American Chemical Society, p.18.

Brauer, W., 1979, *Net theory and applications*, Hamburg, Springer-Verlag.

Bylander, T., Smith, J.W., and Svirbely, J., 1988, *Qualitative Representation of Behaviour in the Medical Domain*, Computers and Biomedical Research 21, 367-380.

Catino, C., 1993, *Automated Modelling of Chemical Plants with Application to Hazard and Operability Studies*. PhD dissertation, University of Pennsylvania: Chemical Engineering.

Catino, C.A., Grantham, S.D. and Ungar, L.H., 1991, *Automatic generation of qualitative models of chemical process units*, Computers and Chemical Engineering, 15(8), pp583-599.

Catino, C., and Ungar, L.H., 1995, *Model based approach to automated hazard identification of chemical plants*, American Institute of Chemical Engineering Journal, 41(1), 97-109.

Chae, H., Yoon, Y.H. and Yoon, E.S., 1994, *Safety analysis using an expert system in chemical processes*, Korean Journal of Chemical Engineering, 11(3), pp153-161

Chapman, D., 1987, *Planning for conjunctive goals*, Artificial Intelligence 32(3).

Chen, H. and Hanisch, H.M., 1998, *Integration of batch plant scheduling and control by means of hybrid net condition/event system model*, in: Proceedings of the 37th IEEE Conference on Decision and Control, Vol. 3, Tampa, USA, pp3218-3223.

Chung, P.W.H., 1993, *Qualitative analysis of process plant behaviour*, 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Gordon and Breach, pp277-283.

Cox, R.A., 1987, *An Overview of Hazard Analysis*, International Symposium on Preventing Major Chemical Accidents, Washington D.C., February, AIChE, p.1.37.

Culbertson T.L. and A. H. Searson, 1983, *Exxon Facility Design Assessment and Control of Hazards*, internal publication, September.

Dague, P., Raiman, O. and Deves, P., 1987, *Troubleshooting: When Modelling is the Trouble*, Proceedings of AAAI-87, pp590-595.

David, R., 1997, *Modelling of hybrid systems using continuous and hybrid Petri nets*, in: Proceedings of International Conference on Petri nets and Performances Evaluation, Saint Malo, France, pp47-58.

David, R. and Alla, H., 1994, *Petri nets for modelling of dynamic systems: a survey*, Automatica, 30, pp175-202.

De Coste, D., 1994, *Goal directed Qualitative Reasoning with Partial States* (Tech. Rep. No. 57), Evanston: Northwestern University, Institute for the Learning Sciences.

De Kleer, J., 1977, *Multiple Representations of Knowledge in a Mechanics Problem Solver*, Proceedings of IJCAI-77, pp299-304.

De Kleer, J., 1984, *How Circuits Work*, Artificial Intelligence 24, pp205-280.

De Kleer, J., and Brown, J., 1984, *A Qualitative Physics Based on Confluences*, *Artificial Intelligence*, 24, pp7-83.

Dennis, J., June 1970, *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, New York: AMC.

Dijkstra, E.W., 1976, *Discipline of programming*, prentice hall, Englewood cliffs, NJ.

Fikes, R.E. and Nilsson, N.J., 1971, STRIPS: *A new approach to the application of theorem proving to problem solving*, *Artificial Intelligence*, 2(3-4), pp189-208.

Dow Chemical, 1976, *Fire and Explosion Index*, Corporate Safety and Loss Prevention, Dow Chemical Company.

Drabble, B., 1993, *Excalibur: A Program for Planning and Reasoning with Processes*, *Artificial Intelligence*, 62(1), pp1-40.

Dvorak, D., and Kuipers, B., 1989, *Model-based Monitoring of Dynamic Systems*, Proceedings of AAAI-89.

Falkenhainer, B., 1990, *A Unified Approach to Explanation and Theory Formation*. Shrager & Langley (Eds.), Computational models of scientific discovery and theory formation, San Mateo, CA: Morgan Kaufmann Publishers.

Faltings, B., & Struss, P.(Eds.), 1992, *Recent Advances in Qualitative Physics*, Cambridge, Mass: MIT Press.

Forbus, K., 1984, *Qualitative process theory*, Artificial Intelligence, 24, pp85-168.

Forbus, K., 1989, *Introducing Actions into Qualitative Simulation*, Proceedings of IJCAI-89, pp1273-1278.

Forbus, K., 1996, *Qualitative Reasoning*, CRC Handbook of Computer Science.

Forbus, K., and Falkenhainer, B., 1990, *Self-Explanatory Simulations: An integration of qualitative and quantitative knowledge*, Proceedings of AAAI-90, pp380-387.

Genrich, H.J., 1987, *Predicate/Transition Nets, Lecture notes in Computer Science*, Vol.254, Springer, Berlin, pp207-247.

Green C., 1969, *Application of theorem proving to problem solving*, In Proceedings of IJCAI-69.

Guerrin, F., 1991, *Qualitative reasoning about an ecological process: Interpretation in hydroecology*, Ecological Modelling, 59, pp165-201.

Hanisch, H.M., 1992, *Coordination control modelling in batch production systems by means of Petri net.*, Computers & Chemical Engineering, 16, pp1-10.

Hanisch, H.M., 1994, *Discrete models of the dynamic behaviour of batch plants*, Computers & Chemical Engineering, 18, ppS403-S407.

Hayes, P., 1979, *The naïve physics manifesto*, In D. Michie(Ed.) Expert Systems in the Micro-Electronic Age, Edinburgh University Press.

Hayes, P., 1985, *Naïve Physics 1: Ontology for liquids*. In Hobbs, R., & Moore, R. (Eds.), *Formal Theories of the Commonsense World*. Norwood, NJ: Ablex Publishing Corporation.

Hoffmann, J.M., 1985, *Chemical Process Hazard Review*, Chemical Process Hazard Review, American Chemical Society, p1.

Hogge, J.C., 1987, *Compiling Plan Operators from Domains Expressed in Qualitative Process Theory*, Proceedings of IJCAI-87, pp229-233.

Holt, A.W., Saint, H., Shapiro, R. and Warshall, S., 1968, *Final Report of the Information Systems Theory Project*, Technical Report RADC-TR-68-305, Rome Air Development Centre, Griffis Air Force Base, New York (Sept. 1968), 352 pages. Distributed by Clearing house for Federal Scientific and Technical Information, US Department of Commerce.

Hunt, A., Kelly, B.E., Mullhi, J.S., Lees, F.P. and Rushton, A.G., 1993, *The propagation of faults in process plants: 6, Overview of, and modelling for, fault tree synthesis*, Reliability Engineering and System Safety, 39, 173-194.

ISA S88.01, 1995, *Batch Control, Part 1: Models and Terminology*, ISBN: 1-55617-562-0.

Iwasaki, Y., Tessler, S., & Law, K., 1995, *Qualitative Structural Analysis Through Mixed Diagrammatic and Symbolic Reasoning*, In J. Glasgow, B. Karan, & N. Narayanan (Eds.), *Diagrammatic Reasoning*. Cambridge, Mass: AAAI Press/The MIT Press, pp711-729.

Jefferson, M., Chung, P.W.H. and Rushton, A.G., 1995, *Automated hazard identification by emulation of hazard and operability studies*, Proceedings of the

8th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Forsyth, G.F. and Ali, M.(eds), Gordon and Breach, Melbourne, Australia, pp765-770.

Johnsson, C., 1999, *A Graphical Language for Batch Control*, PhD Thesis, Lund University, Chapter 4, p.50.

Kamps, J., & Peli, G., 1995, *Qualitative Reasoning beyond the Physics Domain: The Density Dependence Theory of Organizational Ecology*, Proceedings of QR-95, pp114-121.

Kay, H., & Kuipers, B., 1993, *Numerical Behaviour Envelopes for Qualitative Models*, Proceedings of AAAI-93, pp606-613.

Kim, H., 1992, *Qualitative Kinematics of Linkages*, In B.Faltings&P.Struss(Eds.). *Recent Advances in Qualitative Physics*, Cambridge, Mass.: The MIT Press.

Kimble, D.L., 1997, <http://taylor.ces.clemson.edu/ie340/files/340-16.htm>, IE340 Lecture Notes#16, Petri Nets, accessed on May 3, 2002.

Kitajima, T. et al., 2000, *A coloured Petri Net Model for analysing Batch Sequential Control System* Hibarigaoka, Tenpaku-cho, Toyohashi, Aichi, 441-8580, Japan.

Kletz, T., 1999, *Hazop and Hazan*, 4th edition, Institute of Chemical engineers

Knoblock, C.A., 1995, *Planning, Executing, sensing, and replanning for information gathering*, In Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995.

Kuipers, B., 1986, *Qualitative Simulation*, Artificial Intelligence, 29, pp289-338.

Kuipers, B., 1994, *Qualitative Reasoning: Modelling and simulation with incomplete knowledge*, Cambridge, Mass.: MIT Press.

Kuipers, B. & Berleant, D., 1988, *Using Incomplete Quantitative Knowledge in Qualitative Reasoning*, Proceedings of AAAI-88, pp324-329.

Lang, K.R. Moore, J.C., and Whinston, A.B., 1995. *Computational systems for qualitative economics*, Computational Economics, 8, 1-26.

Larkin, F.D., Rushton, A.R., Chung, P.W.H., Lees, F.P., McCoy, S.A., & Wakeman S.J., 1997, *Computer-aided hazard identification: methodology and system architecture*, IchemE Symposium Series No.141, Hazards XIII Process Safety – The Future, pp337-348.

Lawley, H.G., 1974, *Operability studies and hazard analysis*. Chemical Engineering Progress, 70, pp105-116.

Lawley, H.G., 1976, *Size up plant hazards this way*. Hydrocarbon Processing, 55. pp247-261.

Le Clair, S., Abrams, F., & Matejka, R., 1989, *Qualitative Process Automation: Self Directed Manufacture of Composite Materials*, AI EDAM, 3(2), pp125-136.

Lees, F.P., 1996, *Loss prevention in the process industries: hazard identification, assessment, and control, vol.2 (2nd ed.)*. London: Butterworth-Heinemann.

Leone, H., 1996, *A knowledge-based system for Hazop studies*, Computers and Chemical Engineering, 20(Suppl.), S369-S374.

Lewis, D.J., 1979, *The Mond Fire Explosion and Toxicity Index – A Development of the Dow Index*, AIChE Loss Prevention Symposium, Houston, Texas.

Lloyd, S. and Salleh, Y.M., 1991, *Modelling and control design of batch process plant by timed Petri net*, in: Proceedings of the 30th IEEE Conference on Decision and Control, Vol.2, pp1531-1536.

Lowe, D.R., and Solomon, C.H., 1983, *Hazard Identification Procedures*, IChemE Symposium Series, No. 80, p. G8.

Mau, K.Y., Nolan, P.F. and Steele, C.H., 1996, *The Development of a Real-Time Emergency Advisory System for Batch Reactors*, Computers in Chemical Engineering, 20(Suppl.), ppS593-S598.

McAllester D. and Rosenblitt, D., 1991, *Systematic nonlinear planning*, In: Proceedings AAAI-91, Anaheim, CA, 634-639.

McCarthy, J., 1960, *Recursive functions of symbolic expressions and their computations by machine*, Communications of the AACM, 7, pp184-195.

McCoy, S.A., 1999, *Combining Qualitative and Quantitative Reasoning to support Hazard Identification by Computer*, PhD Thesis.

McCoy, S.A., Crawley, F. and Chung, P.W.H., 2004, *An Expert User's Trail of the Hazid System for Computer-Aided HAZOP Emulation*, Proceedings of HAZARDS XVIII Symposium , IChemE , Manchester, pp790-805.

McCoy, S.A., Wakeman, S.J., Larkin, F.D., Jefferson, M.L., Chung, P.W.H., Rushton, A.G., Lees, F.P. and Heino, P.M., 1999, *HAZID, A Computer Aid For*

Hazard Identification 1. The STOPHAZ Package and the HAZID Code: An overview, the Issues and the Structure, Transactions of the Institute of Chemical Engineers, Part B (Process Safety and Environmental Protection), 77, pp317-327.

McCoy, S.A., Wakeman, S.J., Larkin, F.D., Jefferson, M.L., Chung, P.W.H., Rushton, A.G., Lees, F.P. and Heino, P.M., 1999, *HAZID, A Computer Aid For Hazard Identification 2. Unit Model System*, Transactions of the Institute of Chemical Engineers, Part B (Process Safety and Environmental Protection), 77, pp328-334.

McCoy, S.A., Wakeman, S.J., Larkin, F.D., Jefferson, M.L., Chung, P.W.H., Rushton, A.G., Lees, F.P. and Heino, P.M., 1999, *HAZID, A Computer Aid For Hazard Identification 3. The Fluid Model and Consequence Evaluation Systems*, Transactions of the Institute of Chemical Engineers, Part B, (Process Safety and Environmental Protection), 77, pp335-353.

McCoy, S.A., Wakeman, S.J., Larkin, F.D., Jefferson, M.L., Chung, P.W.H., Rushton, A.G. and Lees, F.P., 2000, *HAZID, A Computer Aid For Hazard Identification 4. Learning Set, Main Study System, Output Quality and Validation Trials*, Transactions of the Institute of Chemical Engineers, Part B, (Process Safety and Environmental Protection), 78, pp91-119.

McCoy, S.A., Wakeman, S.J., Larkin, F.D., Jefferson, M.L., Chung, P.W.H., Rushton, A.G. and Lees, F.P., 2000, *HAZID, A Computer Aid For Hazard Identification 5. Future Development Topics and Conclusions*, Transactions of the Institute of Chemical Engineers, Part B, (Process Safety and Environmental Protection), 78, pp120-142.

Milne, R., & Trave-Massuyes, L., 1993, *Application oriented Qualitative Reasoning*, The 7th International Workshop on Qualitative Reasoning about Physical Systems, pp145-156.

Murata, T., 1989, *Petri Nets: Properties, Analysis and Applications*, Proceedings of the IEEE, 77(4), pp541-580.

Mushtaq, F. and Chung, P.W.H., 2000, *A systematic Hazop procedure for batch processes, and its application to pipeless plants*, Journal of Loss Prevention in the Process Industries, 13, pp41-48.

Nam, D.S., Jeong, C.W., Choe, Y.J. and Yoon, E.S., 1996, *Operation aided system for fault diagnosis of continuous and semi-continuous processes*, Computers in Chemical Engineering, 20(6/7), pp793-803.

Newell, A. and Simon, H.A., 1963, *GPS, a program that simulates human thought*, In *Computers and Thought*, ed. E.A. Feigenbaum and J. Feldman. New York: McGraw-Hill.

Ozog, H., 1987, *Hazard Identification Analysis and Control*, Chemical Engineering, February 18, p161.

Ozog, H. and Bendixen, L.M., 1987, *Hazard Identification and Quantification*, Chemical Engineering Progress, p55.

Parmar, J.C., & Lees, F.P., 1987a, *The propagation of faults in process plants: hazard*, Reliability Engineering, 17, pp277-302

Parmar, J.C., & Lees, F.P., 1987b, *The propagation of faults in process plants: hazard identification for a water separator system*, Reliability Engineering, 17, pp303-314.

Peterson, J.L., 1981, *Petri net Theory and the Modelling of Systems*, Prentice Hall, Englewood Cliffs.

Pednault, E.P.D., 1989, ADL: *exploring the middle ground between STRIPS and the situation calculus*, in: Proceedings First International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Ont., pp324-332.

Peterson J. L., 1981, *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, Englewood Cliffs, New Jersey.

Petri C.A., 1962, *Kommunikation mit Automaten*. Ph.D Dissertation, University of Bonn.

Proth, J., and Xie, X., 1996, *Petri nets A Tool for Design and Management of Manufacturing Systems*, John Wiley & Sons.

Reece, S., & Durrant-Whyte, H., 1995, *A Qualitative Approach to Sensor Data Fusion for Mobile Robot Navigation*, Proceedings of IJCAI-95, 36-41.

Reisig, W., 1985, *Petri Nets*. Springer-Verlag, New York.

Rich, E., 1991, *Artificial Intelligence*, 2nd edition, McGraw-Hill, Inc.

Russell S., and Norvig, P., 1995, *Artificial Intelligence: A Modern Approach*. Prentice Hall Inc.

Sacerdoti, E.D., 1975, *The nonlinear nature of plans*, In Proceedings of IJCAI-75.

Selic, B., Gullekson, G. and Ward, P.T., 1994, *Real-Time Object-Oriented Modelling*, John Wiley & Sons, Inc.

Shimada, Y., Suzuki, K. and Sayama, H., 1996, *Computer-aided operability study*, Comp Chem Eng, 20(6/7), pp905-913.

Shimada, Y., Yang, Z.X., Song, J.W., Suzuki, K. and Sayama, H., 1995, *Computer-aided Operability Study for Batch Plants*, Loss Prevention and Safety Promotion in the Process Industries, II, pp587-598

Silva, M., Teruel, E., Valette, R. and Pingaud, H., 1998, *Petri nets and production systems*, Lectures on Petri nets II: Applications, Lecture notes in Computer Science, Vol. 1492, Springer, Berlin, pp85-124.

Slater, Corran, and Pitblado (eds.), 1987, *Major Industrial Hazards Project Report*, The Warren Centre for Advanced Engineering, University of Sidney.

Soutter, J., 1997, *An Integrated Architecture for Operating Procedure Synthesis*, PhD Thesis, Loughborough University.

Srinivasan, R. and Venkatasubramanian, V., 1996, *Petri net digraph models for automating HAZOP analysis of batch process plants*, *Computers in Chemical Engineering*, 20 (Suppl.), ppS719-S725

Srinivasan, R., and Venkatasubramanian, V., 1998, *Automating Hazop analysis of batch chemical plants: Part I. The knowledge representation framework*, 20(9), pp1345-1355.

Srinivasan, R., & Venkatasubramanian, V., 1998, *Automating Hazop analysis of batch chemical plants: part II algorithms and application*. *Computers & Chemical Engineering*, 22, pp1357-1370.

Sussman, G.J., 1975, *A computer model of skill acquisition*, Cambridge, MA:MIT Press.

Suzuki, K., Shimada, Y., Song, T., Sayama H. and Nojiri, I., 1998, *An Object-oriented Approach for Computer-aided HAZOP for Batch Plants*, 1998, The 9th

International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Vol.2, pp493-508.

Tate, A., 1977, *Generating project networks*, In Proceedings IJCAI-77.

Tittus, M., and Akesson, K., 1998, *Modular supervisors for deadlock avoidance in batch processes*, in: Proceedings of the 1998 IEEE International Conference on Systems, Man and Cybernetics, Vol.1, San Diego, USA, pp764-769.

Tittus, M. and Akesson, K., 1999, *Petri net models in batch control*. Mathematical and computer modelling of dynamics systems, 5, pp113-132.

Tittus, M., Fabian, M. and Lennartson, B., 1995, *Controlling and co-ordinating recipes in batch applications*, in: Proceedings of the 34th IEEE Conference on Decision and Control, Vol.2, New Orleans, USA, pp2484-2489.

Vaidhyanathan, R. and Venkatasubramanian, V., 1995, *Digraph-based models for automated HAZOP analysis*, Reliability and System Safety, 50, pp33-49

Vaidhyanathan, R. and Venkatasubramanian, V., 1996, *Experience with an Expert system for automated Hazop analysis*, 20(suppl.). pp S1589-1594.

Valette, R., 1995, *Petri nets for control and monitoring: specification, verification and implementation*, in: Proceedings of the Workshop on Analysis and Design of Event-Driven Operations in Process Systems, Imperial College, London, pp213-228.

Venkatasubramanian, V., & Vaidhyanathan, R., 1994, *A Knowledge-based framework for automating HAZOP analysis*, AIChE Journal, 40(3), pp496-505

Venkatasubramanian, V., Zhao, J., and Viswanathan, S., 2000, *Intelligent systems for Hazop analysis of complex process plants*, Computers and Chemical Engineering, 24, pp2291-2302.

Viswanathan, S., Johnsson, C., Srinivasan, R., Venkatasubramanian, V., and Arzen, K., 1998a, *Automating operating procedure synthesis for batch processes – part I knowledge representation and planning framework*, Computers & Chemical Engineering, 22, pp1673-1685.

Viswanathan, S., Johnsson, C., Srinivasan, R., Venkatasubramanian, V., & Arzen, K., 1998b, *Automating operating procedure synthesis for batch processes – part II implementation and application*, Computers & Chemical Engineering, 22, pp1687-1698.

Viswanathan, A., Zhao, J., Venkatasubramanian, V., Mockus, L., Vinson, J., Noren, A. and Basu, P.K., 1999a, *Integrating Operating Procedure Synthesis and Hazards Analysis Automation Tools for Batch Processes*, Computers and Chemical Engineering, Supplement, ppS747-S750.

Viswanathan, S., Zhao, J., & Venkatasubramanian, V., 1999b, *Integrating operating procedure synthesis and hazards analysis automation tools for batch processes*, Computers & Chemical Engineering, 23, ppS747-S750.

Wakeman, S.J., Chung, P.W.H., Rushton, A.R., Lees, F.P., Larkin, F.D., & McCoy, S.A., 1997, *Computer-aided hazard identification: fault propagation and fault-consequence scenario filtering*, IChemE Symposium Series No.141, Hazards XIII Process Safety – The Future, pp305-316.

Waldinger, R., *Achieving several goals simultaneously*, in: E. Elock and D. Michie, eds., Machine Intelligence 8(Ellis Horwood, Chichester, 1977) 94-136; also in N.J.Nilsson and B. Webber, eds., Readings in Artificial Intelligence (Tioga, Palo Alto, CA, 1981).

Warren, D.H.D., Warplan, 1974, *a system for generating plans*, Technical report 76, University of Edinburgh, Department of Computational Logic.

Weld, D.S., 1994, *An Introduction to least commitment planning*, AI Magazine, 15(4), pp27-61.

Weld, D.S. and de Kleer J., 1990, *Qualitative reasoning about physical systems, Introduction*. Morgan Kaufmann Publishers, San Mateo, Cal., pp1-11.

Weld, D., and de Kleer, J.(Eds.), 1990, *Readings in Qualitative Reasoning about Physical Systems*, Los Altos, CA: Morgan Kaufmann.

Werthner, H., 1994, *Qualitative Reasoning: Modelling and the Generation of Behaviour*, Springer-Verlag Wien, New York.

Williams B.C. and de Kleer J., 1991, *Qualitative reasoning about physical systems, a return to roots*, Artificial Intelligence 51, pp1-9.

Winston, P.H., 1992, *Artificial Intelligence*, 3rd Ed, Addison-Wesley.

Yamalidou, E.C., and Kantor, J.C., 1991, *Modelling and optimal control of discrete-event chemical processes using Petri nets*. Computers & Chemical Engineering 15, pp.503-519

Yamalidou, E.C., Patsidou, E.P. and Kantor, J.C., 1990, *Modelling discrete-event dynamical systems for chemical process control: a survey of several new techniques*, Computers & Chemical Engineering 14, pp28-299.

Yip, K., 1995, *Reasoning about Fluid Motion I: Finding Structures*, Proceedings of IJCAI-95, pp1782-1788.

Zerkani, H. and Rushton, A.G., 1992, *Computer Emulation of Hazard Identification*, International Federation of Automatic Control Workshop, pp221-226.

Zerkani, H. and Rushton, A.G., 1993, *Computer Aid for Hazard Identification*, 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence, Gordon and Breach, pp102-109.

Zhao, J., Viswanathan, S., Zhao C., Mu, F., Venkatasubramanian, V., 2000, *Computer integrated tools for batch process development*, Computers and Chemical Engineering 24, pp1529-1533.

Zhou, M.C. and Venkatesh, K., 1998, *Modelling, Simulation, and Control of Flexible Manufacturing Systems: A Petri net Approach*, World Scientific, Singapore.

Appendix A

Batch Reactor Example 1 – CHECKOP Output

Operation Instruction:

Step (1) valve_to_scrubber open
Step (2) valve_to_next_vessel checkclosed
Step (3) valve_to_sample_point checkclosed
Step (4) solvent_feed_valve open
Step (5) reactor fill_from_storage_tank with solvent until level 12 Tonnes
Step (6) solvent_feed_valve close
Step (7) valve_to_scrubber close

Report for Hazop Result:

Deviation: NO Action -- Step (1) valve_to_scrubber open

Consequence: (5) cause overpressure in the reactor because valve_to_scrubber is off

Deviation: NO Action -- Step (2) valve_to_next_vessel checkclosed

Consequence: (5) can't be completed because valve_to_next_vessel is on

Deviation: NO Action -- Step (3) valve_to_sample_point checkclosed

Consequence: (5) can't be completed because valve_to_sample_point is on

Deviation: NO Action -- Step (4) solvent_feed_valve open

Consequence: (5) reactor can't be filled from storage_tank because there is no flow path between storage_tank and reactor

Deviation: NO Action -- Step (5) reactor fill_from_storage_tank with solvent until level 12 Tonnes

Deviation: NO Action -- Step (6) solvent_feed_valve close

Consequence: Step(5) is not able to be completed as solvent_feed_valve aperture is still on
There could have been more flow than expected.

Deviation: NO Action -- Step (7) valve_to_scrubber close

Appendix B

Batch Reactor Example 2 – CHECKOP Output

Report for Hazop Result

Deviation: NO Action -- Step (1) valve102 open

Consequence: (2) valve102 could be damaged as valve102 is still off whilst pump102's state is on

Consequence: (3) reactor101 can't be filled from tank102 because there is no flow path between tank102 and reactor101

Consequence: (6) There is nothing in reactor101 at the moment, there could be a risk of mechanical damage.

Consequence: (10) reactor101 can't be cooled as there is no content.

Consequence (16) No reaction can occur because the content of the reactor is just ReactantB

Deviation: NO Action -- Step (2) pump102 start

Consequence: (3) the task can't be completed because pump102 is off

Consequence: (6) There is nothing in reactor101 at the moment, there could be a risk of mechanical damage.

Consequence: (10) reactor101 can't be cooled as there is no content.

Consequence (16) No reaction can occur because the content of the reactor is just ReactantB

Deviation: NO Action -- Step (3) reactor101 fill from tank102 with ReactantA until volume 50 percent

Consequence: (6) There is nothing in reactor101 at the moment, there could be a risk of mechanical damage.

Consequence: (10) reactor101 can't be cooled as there is no content.

Consequence (16) No reaction can occur because the content of the reactor is just ReactantB

Deviation: NO Action -- Step (4) pump102 stop

Consequence: (5) valve102 could be damaged as pump102 is still on whilst valve102's aperture is off

Deviation: NO Action -- Step (5) valve102 close

Deviation: NO Action -- Step (6) agitator turn_on

Consequence: (13) agitator is not running while there is a reaction between ReactantA and ReactantB leading to overheating in reactor101

Deviation: NO Action -- Step (7) valve104 open

Consequence: (8) valve104 could be damaged as valve104 is still off whilst pump104's state is on

Consequence: (9) coolingjacket can't flow_from tank104 with coolingwater because there is no flow path between coolingjacket and tank104

Consequence: (13) coolingjacket is not running while there is a reaction between ReactantA and ReactantB leading to overheating in reactor101

Deviation: NO Action -- Step (8) pump104 start

Consequence: (9) coolingjacket can't flow_from tank104 with coolingwater because there is no flow path between coolingjacket and tank104

Consequence: (13) coolingjacket is not running while there is a reaction between ReactantA and ReactantB leading to overheating in reactor101

Deviation: NO Action -- Step (9) coolingjacket flow_from tank104 with coolingwater

Consequence: (13) coolingjacket is not running while there is a reaction between ReactantA and ReactantB leading to overheating in reactor101

Deviation: NO Action -- Step (10) reactor101 cool_content until temperature 25 degree

Deviation: NO Action -- Step (11) valve103 open

Consequence: (12) valve103 could be damaged as valve103 is still off whilst pump103's state is on

Consequence: (13) reactor101 can't be filled from tank103 because there is no flow path between tank103 and reactor101

Consequence (16) No reaction can occur because the content of the reactor is just ReactantA

Deviation: NO Action -- Step (12) pump103 start

Consequence: (13) the task can't be completed because pump103 is off

Consequence (16) No reaction can occur because the content of the reactor is just ReactantA

Deviation: NO Action -- Step (13) reactor101 fill_from tank103 with ReactantB until volume 60 percent

Consequence (16) No reaction can occur because the content of the reactor is just ReactantA

Deviation: NO Action -- Step (14) pump103 stop

Consequence: (15) valve103 could be damaged as pump103 is still on whilst valve103's aperture is off

Deviation: NO Action -- Step (15) valve103 close

Report for Hazop Result

Deviation: Early Action -- Step (2) pump102 start (-1)

Consequence: (2) valve102 could be damaged as valve102 is still off whilst pump102's state is on

Deviation: Early Action -- Step (3) reactor101 fill from tank102 with ReactantA until volume 50 percent (-1)

Consequence: (3) the task can't be completed because pump102 is off

Consequence: (7) There is nothing in reactor101 at the moment, there could be a risk of mechanical damage.

Consequence: (10) reactor101 can't be cooled as there is no content.

Consequence (16) No reaction can occur because the content of the reactor is just ReactantB.

Deviation: Early Action -- Step (4) pump102 stop (-1)

Consequence: (3) the task can't be completed because pump102 is off

Consequence: (6) There is nothing in reactor101 at the moment, there could be a risk of mechanical damage.

Consequence: (10) reactor101 can't be cooled as there is no content.

Consequence (16) No reaction can occur because the content of the reactor is just ReactantB.

Deviation: Early Action -- Step (5) valve102 close (-1)

Consequence: (5) valve102 could be damaged as pump102 is still on whilst valve102's aperture is off

Deviation: Early Action -- Step (6) agitator turn on (-1)

Deviation: Early Action -- Step (7) valve104 open (-1)

Deviation: Early Action -- Step (8) pump104 start (-1)

Consequence: (7) valve104 could be damaged as valve104 is still off whilst pump104's state is on

Deviation: Early Action -- Step (9) coolingjacket flow from tank104 with coolingwater (-1)

Consequence: (9) coolingjacket can't flow from tank104 with coolingwater because there is no flow path between coolingjacket and tank104

Deviation: Early Action -- Step (10) reactor101 cool content until temperature 25 degree (-1)

Deviation: Early Action -- Step (5) valve102 close (-2)
 Consequence: (2) valve102 could be damaged as pump102 is still on whilst valve102's aperture is off
 Consequence: (3) reactor101 can't be filled from tank102 because there is no flow path between tank102 and reactor101
 Consequence: (6) There is nothing in reactor101 at the moment, there could be a risk of mechanical damage.
 Consequence: (10) reactor101 can't be cooled as there is no content.
 Consequence (16) No reaction can occur because the content of the reactor is just ReactantB

Deviation: Early Action -- Step (6) agitator turn on (-2)
 Consequence: Step(3) There could be liquid spillage as the filling process is going on

Deviation: Early Action -- Step (7) valve104 open (-2)

Deviation: Early Action -- Step (8) pump104 start (-2)
 Consequence: (7) valve104 could be damaged as valve104 is still off whilst pump104's state is on

Deviation: Early Action -- Step (9) coolingjacket flow from tank104 with coolingwater (-2)

Deviation: Early Action -- Step (10) reactor101 cool content until temperature 25 degree (-2)

Deviation: Early Action -- Step (11) valve103 open (-2)

Deviation: Early Action -- Step (12) pump103 start (-2)

Consequence: (11) valve103 could be damaged as valve103 is still off whilst pump103's state is on

Deviation: Early Action -- Step (13) reactor101 fill from tank103 with ReactantB until volume 60 percent (-2)

Consequence: (13) reactor101 can't be filled from tank103 because there is no flow path between tank103 and reactor101
 Consequence (16) No reaction can occur because the content of the reactor is just ReactantA

Deviation: Early Action -- Step (14) pump103 stop (-2)
 Consequence: (15) valve103 could be damaged as pump103 is still on whilst valve103's aperture is off

Deviation: Early Action -- Step (15) valve103 close (-2)

Consequence: (14) valve103 could be damaged as pump103 is still on whilst valve103's aperture is off
 Consequence: (13) reactor101 can't be filled from tank103 because there is no flow path between tank103 and reactor101
 Consequence (16) No reaction can occur because the content of the reactor is just ReactantA

Appendix C

List of Publications

McCoy, S.A., Zhou, D. and Chung, P.W.H., June 2003, *'State-Based Modelling in Hazard Identification*, Proceedings of 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Chung, P.W.H., Hinde, C.J. and Ali, M. (eds), Springer-Verlag, pp. 244-253.

Chung, P.W.H., McCoy, S.A. and Zhou, D.F., October 2003, *Computer-aided Hazard Identification*, Proceedings of Mary Kay O'Connor Process Safety Center Annual Symposium, Texas A&M University, Dr M.S. Mannan (ed.), Omnipress, pp 409-417.

McCoy, S.A., Chung, P.W.H. and Zhou, D.F., May 2004, *Computer-Aided HAZOP of Batch Processes*, Proceedings of European Symposium on Computer-Aided Process Engineering (ESCAPE-14), Barbosa-Povoa, A. and Matos, H. (eds.), Lisbon, published by Elsevier (Amsterdam), ISBN: 0-444-51694-8 (CD Version).

Palmer, C., Chung, P.W.H., Zhou, D.F. and McCoy, S.A., 2005, *Automated Identification of Hazardous Scenarios due to Human Errors in Batch Plant Operation*, Proceedings of the 7th World Congress of Chemical Engineering (CD-ROM), Glasgow, UK, IChemE, Rugby.

McCoy, S.A., Zhou, D.F. and Chung, P.W.H., "State-based modelling in hazard identification", *Applied Intelligence*, accepted for publication.

