Perttu Heino

Fluid property reasoning in knowledge-based hazard identification

TECHNICAL RESEARCH CENTRE OF FINLAND ESPOO 1999

VTT PUBLICATIONS 393

Fluid property reasoning in knowledge-based hazard identification

Perttu Heino

VTT Automation

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of PhD
of Loughborough University
September 30, 1998

TECHNICAL RESEARCH CENTRE OF FINLAND
ESPOO 1999

ISBN 951–38–5395–0–6 (soft back ed.)
ISSN 1235–0621 (soft back ed.)

ISBN 951–38–5396–9 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0849 (URL: http://www.inf.vtt.fi/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

Valtion teknillinen tutkimuskeskus (VTT), Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

Statens tekniska forskningscentral (VTT), Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

Technical Research Centre of Finland (VTT), Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Automaatio, Riskienhallinta, Tekniikankatu 1, PL 1306, 33101 TAMPERE
puh. vaihde (03) 316 3111, faksi (03) 316 3282

VTT Automation, Riskhantering, Tekniikankatu 1, PB 1306, 33101 TAMMERFORS
tel. växel (03) 316 3111, fax (03) 316 3282

VTT Automation, Risk Management, Tekniikankatu 1, P.O.Box 1306, FIN–33101 TAMPERE, Finland
phone internat. + 358 3 316 3111, fax + 358 3 316 3282

Technical editing Kerttu Tirronen

Libella Painopalvelu Oy, Espoo 1999

Heino, Perttu. Fluid property reasoning in knowledge-based hazard identification. Espoo 1999. Technical Research Centre of Finland, VTT Publications 393. 167 p. + app. 53 p.

Keywords safety, hazard identification, HAZOP, computer-assisted hazard identification, physical and chemical properties, knowledge-based systems, process industry

Abstract

The study of serious accidents, which have occurred in the chemical process industry in recent times, highlights the need to understand fluid property related phenomena and the interactions between chemicals under abnormal process conditions or with abnormal fluid compositions. Consideration of these issues should be common practice in professional safety analysis work, and computer programs designed to support this work have to be able to deal with them.

The purpose of Hazard and Operability (HAZOP) study is to identify all possible deviations from the way a plant design is intended to be operated and all hazards associated with these deviations. Due to its systematic nature, the method is a good candidate for automation. Several research groups have developed embryonic knowledge-based HAZOP systems. However, no automated hazard identification features are included in current commercial software packages supporting HAZOP. The main problem of knowledge-based HAZOP systems is their poor performance in relation to the correctness and completeness of the resulting HAZOP study.

This thesis describes a novel methodology for fluid property reasoning in connection to knowledge-based HAZOP. Building on the earlier achievements of Loughborough University (LU) and Technical Research Centre of Finland (VTT) researchers, the methodology enables knowledge-based hazard identification programs to make a more intelligent assessment of the potential hazards and their causes.

In the first phase of the study, a rule-based fluid property and reaction property reasoning system was created for use in the HAZOPTOOL program. In the second phase, the LU fault propagation reasoning methodology implemented in

the AutoHAZID HAZOP emulation program was extended with fluid property reasoning capabilities.

AutoHAZID was subjected to extensive evaluation which consisted of an evaluation workshop, a fluid model oriented study of the workshop results, and comparative testing based on a set of test cases. It was shown that it is possible and beneficial to extend knowledge-based HAZOP with a capability to reason about fluid properties and interactions. A framework for such a system is presented in this thesis together with some ideas for future work. Based on the results of the work reported here, it is recommended that fluid property reasoning is taken into use in any application of knowledge-based hazard identification.

Preface

This work was done at VTT Manufacturing Technology, Safety Engineering in Tampere, Finland. This thesis suggests a knowledge-based method for fluid property considerations during the computerised identification of process hazards.

I wish to express my gratitude to my thesis supervisors, Professor Frank P. Lees and Dr. Andrew G. Rushton, and to my superior, Professor Jouko Suokas, whose encouragement, support and advice made this work possible.

I am grateful to all my colleagues in the Industrial Risk Management group for their collaborative and supportive attitude towards me and my research projects. In particular, I wish to express my warmest thanks to Mr. Erkki Kotikunnas and Dr. Raija Koivisto who were heavily involved in the projects which this work is based on. I also wish to thank Dr. Felim Larkin, Mr. Steve McCoy and Mr. Steve Wakeman for the fruitful discussions during their involvement in the STOPHAZ project as part of the Loughborough University project group. In addition, all the other partners of the STOPHAZ project gave their valuable contribution to making the circumstances favourable for my work.

The project funding provided by the Technology Development Centre of Finland (TEKES) and CTCI Corporation, Taipei, Taiwan, R.O.C. is gratefully acknowledged, as well as the complementary project funding provided by VTT and the financial support provided by The Finnish Work Environmental Fund for the preparation of this thesis.

Finally, I wish to express my loving thanks to my wife Päivi and my children Helinä, Ilmari and Emilia. They have always done their best to show me that I do not have to worry about them when my work forces me to be away from them.

Tampere, September 1998

Perttu Heino

Contents

1. Introduction

1.1 Computer software in the safety analysis of chemical processes

Safety analysis is the systematic examination of the structure and functions of a system. Its aim is to identify all factors having the potential to contribute to accidents, to evaluate the risk induced by them and to find ways to minimise these risks (Rouhiainen 1990). Both qualitative and quantitative studies are included in a typical safety analysis of a chemical process system.

Computerised tools have been available from the 1970's for the quantitative estimation of reliability, availability, and accident frequency based on a model of the accident in the form of fault trees. At the same time, calculation programs based on physical models have become available for the estimation of accident consequences in relation to gas releases and dispersions. In their modernised versions, these programs are widely used today. Well-known examples are CARA (SINTEF 1995) and STARS (Poucet 1990) for fault tree quantification, and SAFETI (Technica 1994).and RISKWIT (Kakko 1991) for gas dispersion calculations.

Following the evolution of user interaction methods for text and graphics processing in the personal computer (PC) environment, programs became available for the efficient preparation and management of safety analysis documentation. The qualitative study of the system during the identification of potential hazards can benefit from the use of such tools which sometimes also offer support for the steering of the analysis process. Naturally, also the documentation of fault tree based quantitative studies, as well as the documentation of risk reduction recommendations, can be made more efficient by the use of appropriate tools of this kind. Widely used software packages for the documentation of identification phase Hazard and Operability (HAZOP) studies are PHAWORKS (Primatech 1996), HAZOPtimizer (ADL 1997), PHA-Pro (DYADEM 1996) and HAZSEC (Technica 1991).

The first attempts to develop more advanced software for safety analysts dealt with the automated construction of fault trees. It had been recognised that the

modelling of fault propagation event chains is strongly based on a systematic analysis of system structure in terms of equipment and their characteristics and the connectivity between equipment. One problem of the early systems of this type was the insufficient quality and coverage of the model library for the description and analysis of real processes (Suokas and Karvonen 1985). The large system description effort and the need to examine the end results very closely in order to interpret them correctly were other main obstacles for the widespread use of these advanced techniques. Although there has been a lot of progress since the early days of automated fault tree construction, such software is still not used widely. An example of a modern fault tree workstation software package with good automated fault tree construction capabilities is STARS (Poucet 1990).

The importance of hazard identification techniques, HAZOP in particular, has increased during the recent years due to new legislative requirements in Europe (EEC 1982) and USA (OSHA 1992). This has turned the researchers' attention to automating the HAZOP method. Several research groups work actively on this topic (Rushton 1997). The algorithms for automated HAZOP correspond closely to the ones developed earlier for automated fault tree construction. The problems are also similar. The characteristics and problems of automated HAZOP form the starting point of this work and a detailed discussion of the topic will follow.

There is a variety of software commercially available for safety analyses and related tasks. In a recent catalogue (CEP 1997), 77 packages were listed under the category "Reliability, Failure analysis, Risk analysis". This includes HAZOP software from seven software provider companies. The most advanced features of these packages include knowledge libraries, intelligent checklists and analysis process guidance. No automated HAZOP features, such as discussed above, are included in any of these packages. This is a clear indication of the fact that automated HAZOP still has such problems which make it commercially uninteresting.

1.2 Need for the study

Significant benefits could potentially be gained by the use of automated HAZOP. Benefits are expected in the following areas:

Less human effort is required to perform a HAZOP study

- Generating a HAZOP study using a knowledge-based program before the groupwork HAZOP sessions would take care of listing the most self-evident event chains which can potentially lead to hazards. The HAZOP group could then quickly check the computer-generated results and concentrate on more complicated considerations which deal with issues specific to the studied process. The total amount of required manpower could be decreased in this way at the same time as the quality of the results is improved.

- The use of automated HAZOP earlier than HAZOP is conventionally performed would lead to the identification and elimination of part of the problems before the groupwork HAZOP. During the group sessions, fewer problems would need to be identified which would make it possible for the group to complete the HAZOP in less time.

The HAZOP process becomes standardised

- HAZOP studies produced using a knowledge-based computer program are similar in quality and style regardless of who is using the program. If the use of such a program were to become the normal industrial practice, HAZOP studies produced by different companies and different people would in practice follow a standard imposed by this program. A standard minimum level of quality would be guaranteed in case the HAZOP study is carried out by people with insufficient expertise. Experienced HAZOP analysts would normally exceed this minimum level of quality but the style of their reports could still follow the same standard.

Alternative solutions can be assessed quickly with respect to hazards

- Design alternatives or proposed modifications can be examined by comparing the HAZOP studies of the alternatives. It is a minor task to

modify the plant description which is given to the HAZOP program as an input to correspond to alternative designs or proposed modifications.

HAZOP-type quick studies can be carried out earlier during design

- The designer can use the knowledge-based HAZOP program to check the impact of the latest progress of the design on safety. This can be repeated as often as appropriate. For such use, a direct link from the design drawing system to the HAZOP program is required.

HAZOP results are stored in a data base and are easier to update

- The knowledge-based HAZOP program stores the resulting HAZOP information in a data base. When updating the HAZOP study becomes necessary, the old HAZOP study of the system can be reviewed with the support of the program, and new computer-generated results can be used as the starting point of a revised HAZOP report together with the old results which incorporate the contribution of the HAZOP group.

Corporate memory on process hazards is created

- In order to maintain the best possible quality of the knowledge-based HAZOP, its generic models of process equipment behaviour should be kept up-to-date by the user organisation. The model library will then serve as the corporate memory of equipment failures and the associated hazards.

In spite of their evident potential benefits, knowledge-based HAZOP systems have not been taken into industrial use. Although the laborious nature of the data input stage is one deficiency of these systems, the main problem is their poor performance in relation to the correctness and completeness of the resulting HAZOP study. This is due to the fact that the current systems are based on highly generalised knowledge of the behaviour of process equipment and can not perform advanced reasoning on the special characteristics of the chemical process.

An obvious example of special characteristics beyond the scope of most current knowledge-based HAZOP systems is the impact of the properties of the

processed chemicals on the identified hazards. Work needs to be done in this area in order to improve the performance of automated HAZOP up to an industrially acceptable level.

1.3 Scope and objective of the study

The performance of automated HAZOP has to be improved before industrial acceptance of the methodology can be expected. Current systems have major deficiencies in the area of fluid property reasoning. It has already been suggested by recent limited developments that significant performance improvement could be achieved by adding fluid property reasoning capabilities to knowledge-based HAZOP. The full-scale development of these techniques in connection to a state-of-the-art HAZOP system is needed.

It is hypothesised as the basis of this work that an appropriate fluid property reasoning methodology leads to a significant improvement when applied in a knowledge-based HAZOP framework.

The work involves theoretical study, methodology development, testing and evaluation in relation to the intelligent use of the following types of data:

- properties of chemicals
- compatibility (chemical/chemical, chemical/contaminant, chemical/material)
- properties of chemical reactions.

The objective of this work is to improve the performance of knowledge-based hazard identification methods. The intention is to enable the HAZOP program to make a more intelligent judgment of the potential hazards and their causes based on the properties of the processed fluid. This should not only be done based on the intended composition of the fluid, but also taking abnormal fluid compositions into consideration. Possible reactions between the normal fluid components and potential contaminants should also be considered.

The scope is limited to the domain of knowledge-based hazard identification methods. Discussion of other improvements to automated HAZOP than those related to fluid property reasoning has been excluded.

1.4 Structure of the thesis

After the introductory discussions and the definition of scope in chapters 1-3, the theoretical framework of the work is described in chapter 4. Then, the first development stage of the fluid property reasoning methodology is discussed in chapter 5. This part of the work is based on the results of the HAZOPTOOL project which was carried out in 1992-1994.

Chapter 6 includes the discussion of the second development stage in 1994-1997 in connection to the STOPHAZ project. The final part of the work, chapters 7-9, contain a discussion of the study outcomes, the final conclusions and recommendations for further work.

2. Knowledge-based identification of process hazards

Knowledge-based programs for process hazard identification take as input the description of the process to be studied. As a result of the knowledge-based processing, a report of causal chains of events associated with potential hazards is generated.

In his recent survey of knowledge-based HAZOP systems, Rushton (1997) illustrates the architecture of a stereotype knowledge-based HAZOP system as shown in Fig. 1.

Separation of process-specific knowledge from process-independent knowledge is a key feature of these knowledge-based systems. The process-specific part of the knowledge consists of a definition of process equipment and their properties, the connectivity between process equipment, and a description of the process streams in terms of fluid composition and process conditions. This part of the knowledge is often called the plant description.

The process-independent part of the knowledge contains generic models of the behaviour of process equipment in abnormal conditions. Because various equipment types have been considered in the models in isolation from any process environment, the models are universally applicable. Naturally, the model library needs to contain models of such a range of equipment types that the actual process to be studied can be correctly described.

The inference engine of the knowledge-based program applies the process-specific information to the appropriate generic models from the library. In other words, the plant description is used for guiding the selection of information from the generic causal knowledge. The selected information is then reported in a process-specific way.

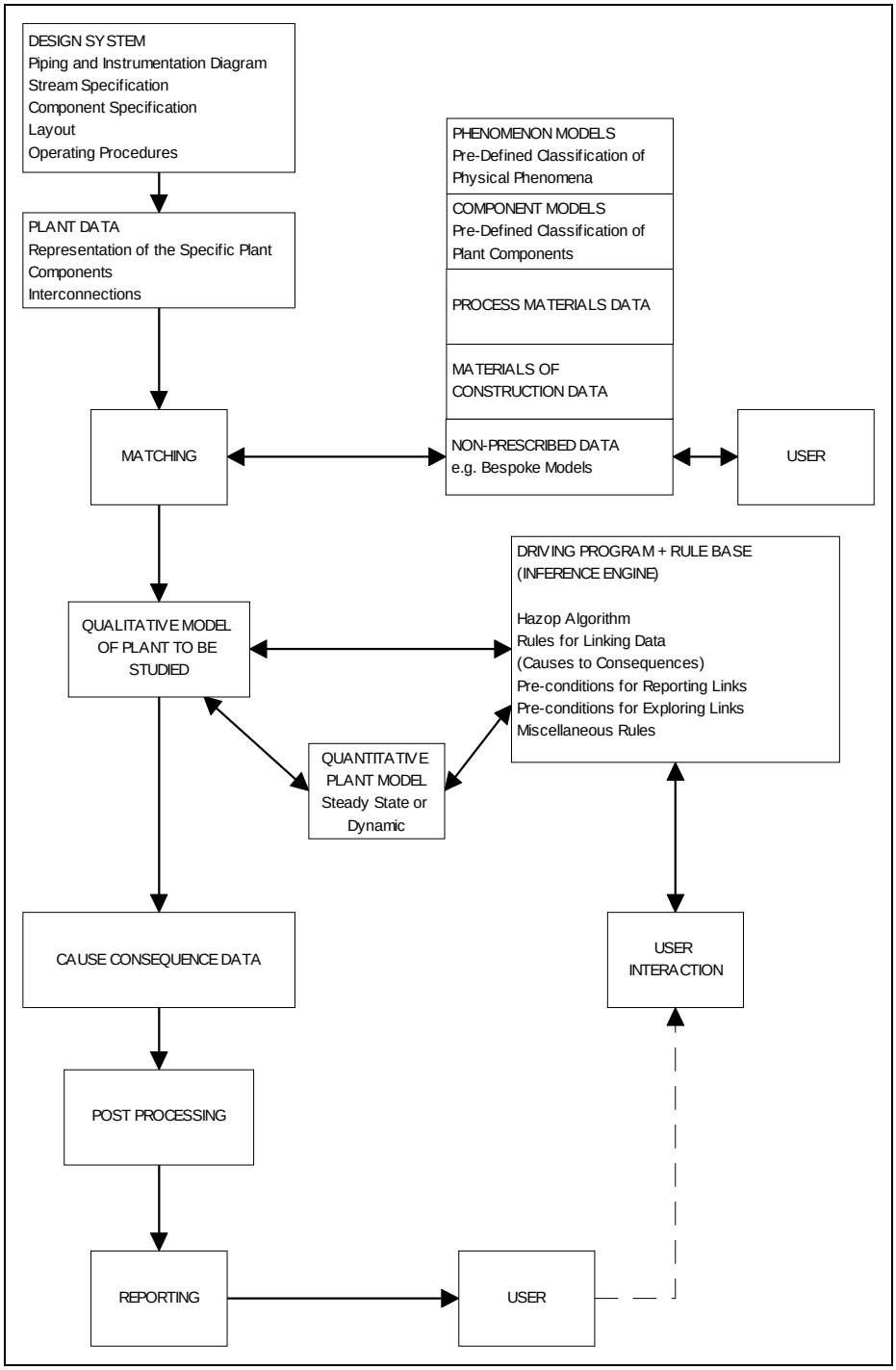


Fig. 1. A stereotype knowledge-based HAZOP system (Rushton 1997).

It is very difficult to reach a sufficient level of detail with this type of systems because the models would have to be made highly conditional on process-specific issues to achieve that. Adding a lot of conditions related to process-specific issues would significantly complicate the modelling task, and it would also lead to an increase of required data input effort from the user. However, it is necessary to add this kind of filtering capabilities to the programs. Otherwise, the results will remain on a too general level to be industrially acceptable.

3. The role of fluid property reasoning

Chemicals as accident contributors in reported accident histories

Knowledge-based HAZOP should emulate the information processing of conventional group work HAZOP as closely as possible. Since the objective of HAZOP is to identify all potential event chains which could lead to a major accident it is important to know what the role of chemical phenomena and interactions between chemicals is in the occurrence of incidents at chemical process plants.

In the following, examples of severe chemical accidents will be discussed. The intention is to show that, in order to identify the existence of fluid related hazards and prevent accidents such as the ones discussed below, the properties of the processed fluids and their intended and unintended interactions have to be taken into account.

By examining reports on major chemical accidents one can see that almost all of them have been related to the hazardous characteristics of the processed chemical in a way or another. Lees (1996) provides extensive descriptions of the well-known chemical accidents of Flixborough, Seveso, Mexico City, Bhopal and Pasadena. The Bhopal and Seveso accidents are discussed here in order to illustrate the role of the processed chemicals in the evolution of a major accident.

Lees (1996) also gives a series of chemical accident case histories chosen according to one or more of the following criteria:

1. the accident is well-known, generally by name,
2. it involved major loss of life and/or property, and
3. the physical events and the escalation are of interest.

Selected examples of accidents belonging to this series of 139 accident histories are discussed below, together with the Seveso and Bhopal accidents. This set of accident examples will be revisited in Chapter 7 when the utility and limitations of the proposed methodology is discussed. In addition to a brief description of the role of chemicals in the evolution of the accident, comments are made below

on what would be required from a hazard identification methodology for it to identify the potential for this type of accident.

Ashton, Manchester, England, UK, 1917: A nitrator in an explosives works went out of control and hot acid was released onto the wooden staging. This started a fire which resulted in a large TNT explosion. This kind of a major hazard could have been identified by a comprehensive study of fire sources and the impact of various types of fire incidents to the chemicals and equipment present at the plant.

Texas City, Texas, USA, 1969: The butadiene flow to a distillation column was lost due to a leak from the inlet pipeline, and a build-up of vinyl acetylene occurred. On some trays of the column vinyl acetylene reached the molar concentration of 50-60%, and thermal decomposition of vinyl acetylene and ethyl acetylenes occurred resulting in a violent explosion of the column. This potential hazard could have been identified by a comprehensive study of how the process conditions and concentrations change inside the column in the case where the incoming flow of butadiene is lost.

Baton Rouge, Louisiana, USA, 1976: Natural gas used for excluding air from the hydrogen in hydrogen compressors entered a half full chlorine tank through a common nitrogen system. The resulting internal explosion at the other end of the horizontal bullet-type chlorine vessel caused the vessel to fall off its supports so that it was pierced by a metal upstand on the ground. This led to a massive release of chlorine to the environment. A thorough study of unintended interactions of the chemicals present at the plant could have drawn attention to the potential for this kind of a severe accident.

Breahead, Renfrew, Scotland, UK, 1977: A chemicals warehouse caught fire and, as a result, 1774 steel vessels containing sodium chlorate overheated and exploded. This example illustrates the need to consider the behaviour of the involved chemicals in case of heating by external fire.

Bhopal, India, 1984: Water entered a methyl isocyanate (MIC) tank in an unknown manner. The exothermic reaction between MIC and water led to

an increase of temperature and pressure. Highly toxic MIC was released to the environment through the relief valve. A thorough study of the hazards associated with potential runaway reactions in the storage of toxic materials at the Bhopal site could have relieved the severity of such an accident in relation to MIC storage and led to the use of appropriate preventive measures. In fact, awareness of the potential major hazard should have led to the decision not to have a MIC storage facility at the site at all.

Seveso, Italy, 1986: Due to unintended reactor temperature increase, a remarkable quantity of very toxic 2,3,7,8-tetrachlorodibenzo-p-dioxin (TCDD) was formed and released to the environment subsequent to pressure increase and bursting disc rupture. Significant amounts of TCDD are not formed under the normal process conditions. Originally, the process was operated in atmospheric conditions and the formation of TCDD was not possible. However, when the process was modified to be operated under pressurised conditions, the major hazard associated with the potential formation and release of TCDD could have been identified.

Jonova, Lithuania, 1989: Warm ammonia at 10ºC entered a tank containing ammonia at -33ºC. It first formed a layer on the bottom of the tank but then rose suddenly to the surface. The higher vapour pressure of the warmer liquid caused a sudden rise in pressure in the tank, and a tank rupture and a release of ammonia followed. This kind of a hazard could be identified by a study of vapour pressure changes in case of temperature increase.

Stanlow, Cheshire, England, UK, 1990: Water ingress through a leaking valve in a late batch distillation stage of a batch production of 2,4-difluoronitrobenzene (DFNB) led to the formation of acetic acid during water removal by prolonged azeotropic distillation. After this, the subsequent use of recycled N,N-dimethylacetamide (DMAC) for a new batch led to reaction runaway and explosion due to the presence of acetic acid. Although the potential for this kind of an incident involving complex chemistry is not easy to identify in advance, maximal attention should be paid to potential hazards associated with abnormal fluid compositions and process conditions.

The presented examples highlight the need to understand fluid property related phenomena and the interactions between chemicals under abnormal process conditions and fluid compositions. Typically, an unwanted chain of events is triggered by an unintended increase of temperature, often coupled with an abnormal composition of the fluid either as a cause or a consequence. Consideration of these issues should be common practice in professional safety analysis work, and computer programs designed to support this work have to be able to deal with them, preferably in an intelligent manner.

Fluid property reasoning in knowledge-based hazard identification

Modern approaches to automated hazard identification are based on composing a process specific qualitative model out of a library of generic unit models. These generic models deal with the potential disturbances of process conditions: how the disturbance is initiated, how it propagates through the unit, and what potentially hazardous consequences can be triggered. The basic methodology excludes the consideration of phenomena which are dependent on the chemical composition of the processed fluid and its properties

The most advanced knowledge based HAZOP systems include simple features for taking into account the characteristics of the processed chemicals during HAZOP reasoning. An early attempt to deal with fluid considerations was made in the STARS project (Christensen et al. 1991). Recently, Vaidhyanathan and Venkatasubramanian (1996) have proposed a semi-quantitative approach which deals with a number of fluid dependent issues. In connection to the STOPHAZ project and the development of a fully automated HAZOP tool as part of the project (Preston 1995), the generic methodology for fluid considerations discussed in this work was developed. The recent achievements of six research groups on this topic are discussed in Chapter 4.

Fluid properties like toxicity can be checked easily and used for consequence identification. If a reliable estimate of the fluid composition and process conditions to be considered can be made, then state of matter and other actual fluid properties can be calculated, and the relevance of the identified event chains can be evaluated. However, reasoning about the order of magnitude

associated with a process deviation caused by a given fault is a complicated issue.

Extending knowledge-based HAZOP to handle the considerations related to chemical phenomena and interactions between chemicals on the level of a human expert would have a significant impact on its performance. The achievements of the various research groups who have studied this topic indicate that it should be possible to reach a completeness and coverage of the results which is close to the output of conventional group work HAZOP carried out by a group of human experts.

4. Theoretical framework

4.1 Introduction

In this chapter, the theoretical framework for this work is discussed. Key concepts are defined in the beginning. Next, an overview of hazard identification is presented. In particular, the Hazard and Operability Study (HAZOP) method and its strengths and weaknesses concerning the computerisation of the method are discussed. After that, the results of six research groups in the area of knowledge-based hazard identification are reviewed. This review of the state-of-the-art aims to define the starting point for this work. The final part of this chapter adds detail to the review by taking a closer look at the methods developed by the six groups. The focus is on issues related to the processing of fluid property information.

4.2 Definition of concepts

Fluid — The processed chemical substance or mixture of chemical substances, including gases, liquids and relevant types of solids (e.g. powders).

Fluid composition — Information on what chemical substances the fluid contains and what are their concentrations.

Fluid property — A numeric or symbolic value which characterises the fluid in terms of a distinct aspect of behaviour.

Hazard — A condition that is prerequisite to a mishap. Prevention of accidents is done by eliminating or minimising hazards.

Hazard identification — An activity which aims to detect aspects in the behaviour of a technical system which can lead to a hazard.

Knowledge Declarative statements which describe factual information.

Reasoning Information processing based on the use of knowledge.

Knowledge-based hazard identification

Computerised hazard identification by computer programs which are capable of reasoning with knowledge about process hazards.

Process parameter A variable used for describing a distinct aspect of the behaviour of a chemical process.

Process conditions The values of process parameters at a specific moment of time.

4.3 Overview of hazard identification

The importance of systematic studies in assuring the safety of a system has been widely recognised. Safety analysis techniques serve as tools for finding the possible links between causes and consequences, in order to devise ways in which the activation of these links can be avoided (Hollnagel and Cacciabue 1992). The purpose of Hazard and Operability (HAZOP) study is to identify all possible deviations from the way the design is expected to operate and all hazards associated with these deviations (Anon. 1977). HAZOP has established itself as the common practice in chemical process safety. Discussion on HAZOP and other hazard identification methods can be found in Anon. (1985), Rouhiainen (1990) and Lees (1996).

HAZOP is based on the assumption that a system is safe when all the process parameters are in their normal or another accepted state. Component failures, human errors and threats from the operating environment are reflected as changes of these parameters. A description of the system, typically in the form of flowsheets, line diagrams, and pipe and instrumentation diagrams (P&ID), is

used as the basis for the HAZOP study. HAZOP key words corresponding to all conceivable process parameter deviations are applied to process units such as pipelines, tanks and reactors. The aim is to identify hazardous scenarios where conceivable causes of the deviation and potentially hazardous consequences exist. More complete descriptions of HAZOP can be found in CIA (1997), Kletz (1983), CCPS (1985) and Lees (1996). An extract from the results of a typical HAZOP study is presented in Fig. 2 (Suokas 1985).

Due to its systematic nature, the HAZOP method is a good candidate for automation. Venkatasubramanian and Vaidhyanathan (1994) note that HAZOP is a systematic and logical procedure and, therefore, lends itself to the possibility of automation through the use of the knowledge-based systems approach. As a justification for emulating the HAZOP method in their system, Parmar and Lees (1987) state that completeness in the identification of hazards is the crucial characteristic of an efficient method. HAZOP generates an open set of causes and consequences based on a closed set of deviation events and, therefore, is preferred over top-down and bottom-up methods which always start with a fixed list of causes or consequences.

It is characteristic of the original HAZOP technique, due to its purely qualitative nature, that it does not provide the quantitative ranking of the identified problems. The result of a HAZOP study is a set of identified hazards with their potential causes. The most severe problems are well identified but, at the same time, a number of less relevant event chains are suggested. Some of these findings may even prove to be incorrect or clearly irrelevant when they are examined more closely in a quantitative manner.

It is not possible to completely overcome this problem of the HAZOP method because its roots are in the inherent characteristics of qualitative methods such as HAZOP. However, the human experts who lead HAZOP teams can normally avoid producing uninteresting results. An experienced HAZOP chairman can steer the group to consider relevant issues, and the members of the group can rely on their experience and common sense.

OCCUPATIONAL SAFETY ENGINEERING LABORATORY		SYSTEM:	BLACK LIQUOR RECOVERY BOILER PLANT	PAGE:	8
HAZARD AND OPERABILITY STUDY		SUBSYSTEM:	EVAPORATION PLANT 1	DATE:	18.2.1983
		UNIT:	EVAPORATION UNIT 1 – STRONG LIQUOR TANK	DRAFTED BY:	HAZOP TEAM
DEVIATION	POTENTIAL CAUSES		CONSEQUENCES	ACTION REQUIRED	
NO FLOW	LINE BLOCKAGE, VALVE ERRONEOUSLY CLOSED, PUMP FAILURE, FAILURE IN ELECTRICITY SUPPLY	15	OVERFILLING OF THE EVAPORATION UNIT 1		
HIGH FLOW	LEAKAGE OF THE PUMP WASH WATER LINE VALVE OR THE VALVE IS ERRONEOUSLY OPEN	16	DILUTION OF THE STRONG LIQUOR – ENTRY OF WATER INTO THE STRONG LIQUOR LINE	USING AS NEEDED MOUNTED HOSES INSTEAD OF FIXED WATER PIPE LINES	
	VALVE V1 ERRONEOUSLY OPEN	17	DILUTION OF THE STRONG LIQUOR – ENTRY OF WEAK LIQUOR INTO THE STRONG LIQUOR LINE	REMOVING THE VALVE V1 AND INSERTING BLANKS	
	LEAKAGE OF THE VALVES HV1 AND HV3	18	DILUTION OF THE STRONG LIQUOR – ENTRY OF INTERMEDIATE OR STRONG INTERMEDIATE LIQUOR INTO THE STRONG LIQUOR LINE	CHANGING THE VALVE TYPE OF HV1, HV3 AND HV4 INTO BALL VALVES	
	LEAKAGE OF THE VALVE HV4 WHEN WASHING THE STRONG LIQUOR LINE				
	FALSE OPERATION OF THE INTERDEPENDENT AUTOMATIC VALVES HV1/HV4 AND HV3/HV4 WHEN WASHING THE STRONG LIQUOR LINE			ALARM ON THE FALSE OPERATION OF THE VALVES	

Fig. 2. An extract from the results of a typical HAZOP study (Suokas 1985).

Computerised HAZOP systems have more difficulty in dealing with this problem. There are two ways to make progress:

- Human interaction for the evaluation and acceptance of computer generated findings

- Intelligent operations within the computer program to check the relevance of candidate event chains.

In connection to the development of the HAZOPEX system (Karvonen et al. 1990), the roles of the computer and the human analyst in a computer-aided HAZOP study were considered. The diagram in Fig. 3 shows the flow of the HAZOP analysis process and suggests a division of responsibility between the computer and the user.

There are variations from the course of HAZOP analysis presented in Fig. 3. For instance, some groups may prefer to consider consequences before causes. However, a HAZOP study is always a systematic ’walkthrough‘ of the system line by line or unit by unit.

In computerised HAZOP, the aim is to automate the identification of causes and consequences. As was shown in Fig.1 in Chapter 2, the plant is modelled in qualitative terms and rule-type knowledge is processed in order to generate the HAZOP study report. Different knowledge-based HAZOP systems have different objectives and, therefore, human-computer interaction plays different roles in these systems. Of the systems discussed in the methodology development part of this work, HAZOPTOOL is a browsing tool which considers only one deviation in a single process unit at one time and offers its user the possibility to evaluate the generated candidate event chains after each step of this kind. Therefore, the user has a major influence on which deviations and process units are studied more thoroughly and which of the considered event chains are stored as part of the final HAZOP report.

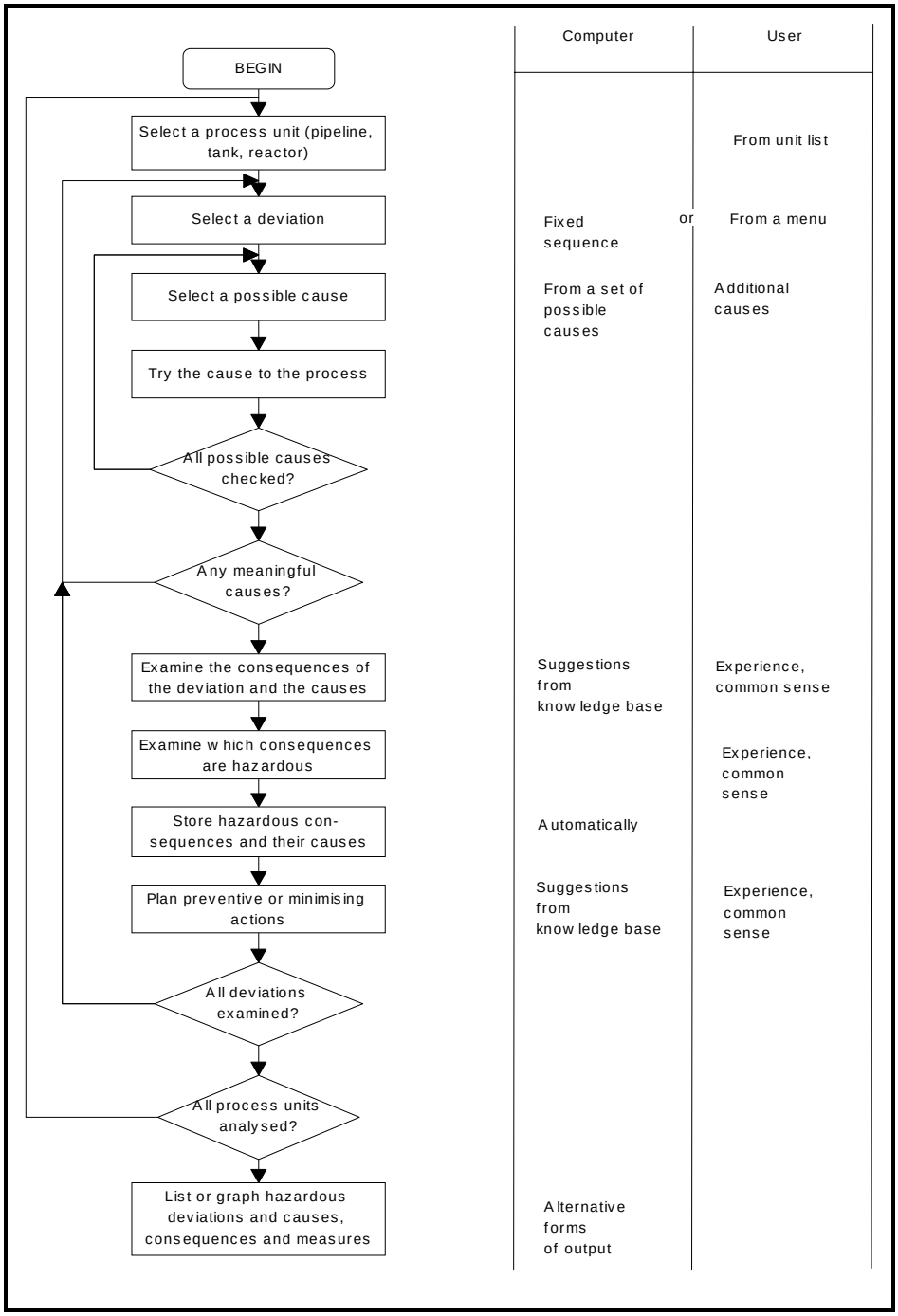


Fig. 3. The course of HAZOP analysis and the tasks of computer and the user (Karvonen et al. 1990).

On the other hand, AutoHAZID aims to fully automate the generation of a HAZOP study report. User interaction can only take place before starting the HAZOP reasoning procedure and after it has been completed. This means that the user can not contribute to the selection of interesting and relevant problems during the ’walkthrough‘ of the process. Therefore, intelligent computerised methods to evaluate the validity and relevance of candidate event chains are needed in order to reach the desired quality of results. Fluid property reasoning has been proposed as one method of that kind.

Even if the user is given the responsibility to evaluate the relevance of the candidate findings, the quality of these computer generated results must be high enough to keep the user interested in them and to keep the required effort acceptable. Therefore, any knowledge-based HAZOP system needs to incorporate some features for screening out irrelevant findings automatically. Because such operations must be based on knowledge of the properties of the process and the fluids, the user is required to input process specific information in addition to describing the process equipment and their connections. In an interactive system, input can be given either in advance or during the HAZOP reasoning. If the HAZOP reasoning stage is implemented as a non-interactive procedure, the information has to be provided in advance, possibly with the support of a set of forms to fill in. Some possibilities also exist for the post-processing of the computer generated results to screen out irrelevant items.

In principle, the quality of the computer generated results gets better when more process specific information is given. At first, the improvement can be very significant but at some point, when the most important characteristics of the process and fluids have already been taken into account, the improvement from providing more information will become too small compared to the effort needed to acquire and to input the information.

4.4 Knowledge-based computerised methods

General structure

The methods developed by different research groups have many similarities. Fig. 1 in Chapter 2 (Rushton 1997) presents the components of a stereotype

knowledge-based HAZOP system. To support the following study and comparison of different methods, the simplified description of knowledge-based HAZOP presented in Fig. 4 will be used.

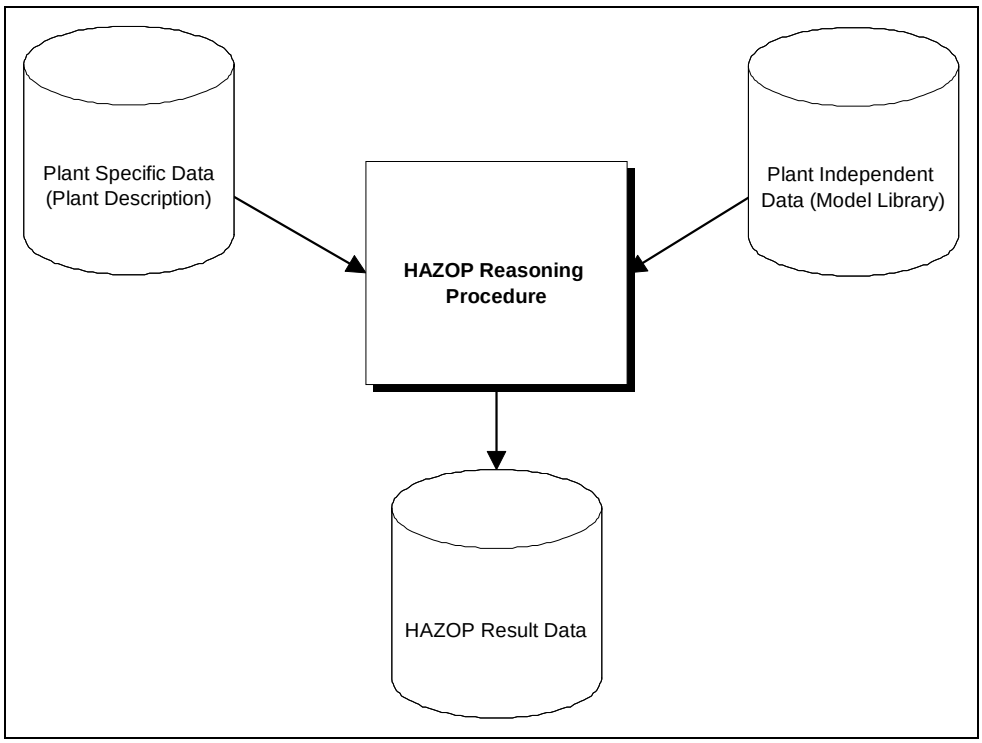


Fig. 4. General structure of knowledge-based HAZOP.

Qualitative models of equipment faults and their propagation form the basis of knowledge-based HAZOP. These plant independent models are stored in a model library which is completely separated from the HAZOP reasoning procedure and the plant dependent data.

The HAZOP reasoning procedure is a computer program which automates the HAZOP study method and uses the model library to get information about potential faults and potential hazards associated with different types of equipment.

In order to apply the generic models to the HAZOP study of a particular plant, a description of the plant has to be given. This plant dependent data states what

equipment exists in the plant and how these equipment are connected together. Based on this information, the relevant models can be picked from the model library and processed by the HAZOP reasoning procedure. The generated HAZOP result data can take various forms but conventional HAZOP tables and fault trees are the most common.

In the following, the achievements of six research groups in the area of knowledge-based HAZOP are discussed. The research groups' work follows the general structure of knowledge-based HAZOP presented above.

Loughborough University of Technology (LUT), now Loughborough University

Researchers led by Prof. F. P. Lees in Loughborough University of Technology are well-known for their comprehensive research work on the computer-aided analysis of fault propagation in chemical process plants. Their early work was published in 1986 in a series of four papers (Kelly and Lees 1986a, b, c and d) dealing with the modelling of fault propagation, fault tree synthesis, the computer program FAULTFINDER, and an example of a pump changeover system. Development of a methodology for modelling fault propagation in a process-independent manner formed a central part of this work.

This fault propagation modelling methodology was the starting point of the subsequent research work on the automation of HAZOP. Parmar and Lees (1987) describe this work which resulted in the Prolog-based hazard identification program IDENTIFIER and a number of supporting FORTRAN programs. The emphasis was still on fault propagation modelling. Propagation equations were used for describing the propagation of faults through a healthy unit, and event statements were introduced in order to model the initiation of a fault in an unhealthy unit or the termination of a fault in a unit which is thereby rendered unhealthy. Rules were chosen as the representation methodology.

The further development of the ideas on knowledge-based computerised HAZOP led to their new implementation in the HAZID program (Zerkani and Rushton 1992). The STOPHAZ project took HAZID as the starting point for the development of industrial quality computerised HAZOP.

The early Prolog-based version of HAZID already contained a system knowledge base defined as a hierarchy of frames, although this model library was still rather incomplete. P.W.H. Chung (1993) took the signed directed graph (SDG) formalism into use in his QUEEN (Qualitative Effects Engine) system which provided a set of common procedures for knowledge-based programs such as HAZID which operate on qualitative models. The HAZOP reasoning procedure and filtering of results in HAZID were also already present in embryonic form and were then developed further in the STOPHAZ project. The features of HAZID as the result of this further development are discussed in Chapter 6.

Technical Research Centre of Finland (VTT)

VTT started research work on knowledge-based methods for computerised HAZOP in 1986 when the HAZOPEX project was launched (Karvonen et al. 1990). The interest in computerised HAZOP was based on VTT's role as a recognised developer of process safety analysis methodologies.

HAZOPEX was a highly interactive expert system-type program which was able to suggest potential causes for process variable deviations. The suggested causes were presented on a HAZOP form which could then be completed by the user. The reasoning was based on rules on the occurrence of process deviations under different circumstances and different process configurations. The topology of process equipment was used for the evaluation of the rules but the structure of the rule base was based on process variables, not on a taxonomy of equipment types as in most other systems. HAZOPEX was implemented using the KEE expert system shell.

As a result of joint preparative work, the STARS project was launched in 1989 by the EC Joint Research Centre (JRC), The Danish Risoe National Laboratory, and VTT. The aim was to develop a new generation fault tree synthesis tool based on the earlier work of JRC (Poucet 1983) and Risoe (Taylor 1982), and a knowledge-based HAZOP tool as part of the same integrated toolkit. This work was completed in 1992 (Heino et al. 1992), resulting in an advanced environment for fault tree studies, a simple chemical hazard expert system, and a prototype tool for knowledge-based HAZOP-type qualitative studies (QUAL).

QUAL was based on the representation of generic knowledge in a frame-based equipment type hierarchy. It was implemented in a Unix workstation environment using C and Xwindows.

The work done in the STARS project on the development of QUAL formed the basis for the development of the HAZOPTOOL program (Heino et al. 1995). The process-independent unit model library with a specific rule syntax and the related inference engine of HAZOPTOOL were the results of the further development of the QUAL ideas. The user interface was quite different, aiming to meet the requirements of the user company which was a Taiwanese engineering company. HAZOPTOOL was implemented using C and Microsoft Windows. It formed the framework for the first phase of the fluid property reasoning methodology development discussed in this thesis. HAZOPTOOL and this development work are discussed in more detail in Chapter 5.

Purdue University

In the early 1990's, researchers in Purdue University, West Lafayette, Indiana, USA, led by Prof. Venkat Venkatasubramanian, started the development of model-based software for the automation of HAZOP studies. This can be seen as a continuation of the earlier Purdue research on integrating compiled and deep-level knowledge for process diagnosis described in Venkatasubramanian and Rich (1988). The Purdue HAZOP system development work has been based on a modelling technique which they call digraphs. It equals to the Signed Directed Graph (SDG) technique used by the LUT researchers. The Purdue researchers have used the G2 software package as the development environment.

The HAZOPExpert system (Venkatasubramanian and Vaidhyanathan 1994, Vaidhyanathan and Venkatasubramanian 1995) is, according to the developers' own statement, the first intelligent (HAZOP) system, and to date the only system, in the published literature that has been successfully evaluated on various industrial-scale case studies (Vaidhyanathan and Venkatasubramanian 1996a). It is beyond the scope of this work to compare the results of the different research groups in that respect.

As in the LUT and VTT systems described earlier, a key feature of HAZOPExpert is the separation of process-independent and process-specific knowledge and the use of these two types of knowledge during reasoning in an integrated manner. Qualitative causal propagation techniques are applied through the use of digraph models with specific nodes for representing abnormal causes and adverse consequences.

The presence of ambiguities in qualitative reasoning on the behaviour of a chemical process has been recognised. In principle, the Purdue researchers have taken a conservative worst-case approach towards ambiguities and order-of-magnitude issues. However, they have successfully extended their system with a semi-quantitative reasoning methodology which integrates the additional quantitative information with the qualitative HAZOP digraph models. This methodology is used for filtering and ranking the adverse consequences found by HAZOPExpert. They also note that the alternative approach of studying exhaustively all possible process behaviours in a qualitative simulation manner would result in combinatorial explosion (Vaidhyanathan and Venkatasubramanian 1996b).

The filtering of adverse consequences in HAZOPExpert is partly based on the use of process material property information and partly on the definition of thresholds for the variables representing process conditions. The approach taken concerning process material property information will be discussed later. The process variable thresholds are adjustable and default as follows:

- design pressure threshold: design pressure > 1.1 * operating pressure (psig)
- design temperature threshold: design temperature > 50 + operating temperature (F).

When the threshold mechanism is applied, the adverse consequences which are directly related to high pressure or high temperature are only reported if the given design pressure or temperature is high enough to violate the corresponding threshold.

Other advanced features of the HAZOPExpert system are:

- four level severity ranking of identified hazardous consequences
- control-loop-cause generic procedure which is triggered whenever a controlled variable is encountered during reasoning
- adequacy of protective devices is checked against the unit design pressures
- ambiguities related to deviations at the inlets of units with multiple inlets are solved by an order-of-magnitude comparison of the inlet flows.

Recently, Srinivasan and Venkatasubramanian (1996) have proposed an extension to HAZOPExpert which combines Petri net representation with the digraph based method. Their aim is to model the recipes and operating sequences of batch processes and thereby to study maloperation related hazards in addition to process variable deviations. In this extended version, process variable deviation reasoning is done inside subtask digraphs, and propagation is carried out by propagating mass-in and heat-in (or mass-out and heat-out) values between the subtask digraphs.

Okayama University

Recent work by Dr. Suzuki and his co-workers in Okayama University, Japan (Shimada, Yang, Song, Suzuki and Sayama 1995; Shimada, Suzuki and Sayama 1996), has also been based on similar qualitative causal propagation techniques. Fault propagation types are classified by considering the input-output relationships among main plant units. This process-independent knowledge, represented in the form of decision tables, covers the propagation of process variable deviations, the impacts of process equipment failures, related potential hazards, and proposed mitigation actions.

Suzuki and co-workers pay a lot of attention to batch processes and chemical reactions. They look at the charge, reaction and discharge stages of a batch process separately. The impacts of abnormal behaviour of reactor support systems, such as ‘less cooling’ and ‘cooling started late’, are considered. The Okayama chemical reasoning approach will be discussed in more detail later.

Seoul National University

In Seoul National Unversity, Republic of Korea, automated hazard analysis of chemical plants has also recently been a topic of extensive studies. Suh, Lee and Yoon (1997a and 1997b) have developed a system which automatically identifies hazardous event chains based on the representation of process knowledge according to a model of the accident mechanism.

In their system, the reasoning is based on three complementary models of process equipment:

Unit function model: Malfunctions and their causes in relation to variable deviations at the inlets or outlets or inside the unit.

Unit behaviour model: Causal relationships between inlet, internal and outlet variables.

Organisational knowledge base: Process streams, side streams and control streams, including process materials and the connectivity between units.

The unit function and behaviour model seem to correspond to the process-independent knowledge base of the earlier described approaches, whereas the organisational knowledge base contains the process-specific structural description.

In the Seoul system, the hazard identification reasoning involves three algorithms:

- Deviation analysis algorithm
- Malfunction analysis algorithm
- Accident analysis algorithm.

The deviation analysis algorithm starts from a process variable deviation and carries out fault propagation to find the causes and consequences. The malfunction analysis algorithm does the same starting from a unit malfunction. The accident analysis algorithm discovers the potential accidents from the

results of the other two algorithms. The three algorithms form a reasoning method which is very similar to the other approaches presented in this work.

As an example, the following accident analysis inference is presented in Suh, Lee and Yoon (1997b):

Level High (deviation) + Open Tank (process unit characteristic)
- ➔ Overflow + Toxic Material ($N_h > 2$)
- ➔ Toxic Material Release + Human
- ➔ Personnel Injury due to Toxic Material.

Santa Fe, Argentina

Leone (1996) and Vecchetti and Leone (1996) present another qualitative modelling approach based on object-oriented methods. The unit class model is the central entity in their approach. In addition, the hazard knowledge concerning the chemicals used or produced is encapsulated within the class Compound. For automated HAZOP, object message sequences covering the following tasks have been defined:

- Causes identification
- Looking for deviation safeguards
- Fault propagation - Consequences identification.

The reasoning in Leone's SERO system results in object diagrams which present the identified hazardous event chains in relation to the process equipment.

4.5 Approaches to fluid property reasoning

Loughborough University of Technology (LUT),
now Loughborough University

In their numerous publications on fault tree synthesis (e.g. Kelly and Lees 1986a, b, c and d), the LUT researchers do not mention the need to consider fluid properties. However, when the same fault propagation methodology was

applied to hazard identification in a parallel development (Parmar and Lees 1987), this issue was taken under consideration. They state that some aspects of modelling are more conveniently handled in terms of the materials used. As examples of this, leak and blockage are mentioned. These generic faults can occur in virtually any unit, but it is only credible specific realisations of these which are of interest.

The concept of a process material model was taken into use. In the modelling related to the development of the Prolog-based IDENTIFIER program, a process material model was created for each process material. This model defined the characteristics of the material in terms of its properties and susceptibilities.

A property is defined by Parmar and Lees (1987) as a characteristic which may lead unconditionally to a consequence without having to be activated by a process variable deviation. For example, a fluid may contain gunk, which results in the unconditional consequence blockage.

A susceptibility is defined as a characteristic which may lead to a consequence, but only if it is activated by a process variable deviation. For example, a fluid may be susceptible if the temperature is high to polymerisation, which thus results in the conditional consequence blockage.

Parmar and Lees (1987) also specify two other types of process material property. One of these is a property which, in combination with a materials of construction susceptibility, results in a specific realisation of a leak. For example, mild steel may be susceptible to an acid impurity. The other type of property is a noxious property relevant to escapes. For example, the fluid may be flammable.

The ideas presented above were not developed further in connection to the early versions of HAZID (Zerkani and Rushton 1992). The reported ideas of Parmar and Lees (1987) mainly concentrated on the consideration of leak, blockage, materials of construction susceptibility, and hazards related to a chemical release to atmosphere. Together with the VTT approach described next they formed the basis for the fluid property reasoning methodology research reported in this thesis. The second phase of the fluid property reasoning methodology research done as part of the STOPHAZ project is discussed in Chapter 6.

Technical Research Centre of Finland (VTT)

The need for reasoning on fluid properties was recognised in the HAZOPEX project (Karvonen et al. 1990). As an example, high temperature suppressing the correct function of an ammonia condenser was mentioned. No implementation of fluid property reasoning was done in connection to HAZOPEX.

In the STARS project (Heino et al. 1992), reasoning methods related to the properties of chemical substances and reactions were considered as a separate activity (Christensen et al. 1991). Three knowledge bases and associated interactive tools were developed:

- Chemical substance knowledge base
- Chemical reaction knowledge base
- Unwanted chemical reactions knowledge base.

The foundations for handling these three types of knowledge were defined in the STARS project but the development of methods for using this knowledge in knowledge-based hazard identification and fault tree synthesis remained at idea stage.

A set of properties which are of interest in the consideration of hazards was defined. In addition, examples of rules connecting the physical, chemical and toxicological properties of chemical substances with a generic list of end consequence-type hazards were developed. The hazard characteristics of different types of reactions and their susceptibility to process variable deviations were also considered. This covered the consideration of intended reactions. Unintended reactions and the associated consequences were to be handled with a general purpose reaction matrix, similar to the one reported by Hatayama (1980).

The above described methodologies of the STARS project were developed further in the HAZOPTOOL project (Heino et al. 1995). This first phase of the fluid property reasoning methodology development work is discussed in Chapter 5 of this thesis.

Purdue University

The HAZOPExpert system of Purdue University uses information on the properties of process materials for consequence filtering purposes. An extensive discussion of this aspect of HAZOPExpert can be found in Vaidhyanathan and Venkatasubramanian (1996b).

In HAZOPExpert, process materials and their properties form a part of the process-specific knowledge. The process-independent knowledge on adverse consequences makes use of this information when required. Generic procedures process-material-cause and process-material-consequence have been defined for the practical implementation of this feature. These generic procedures can be attached to the abnormal cause and adverse consequence nodes of the digraph models.

For instance, one node of the digraph model for a vessel may represent the pressure inside the vessel. The cause-high attribute of the attached abnormal cause node may then list one direct cause ”temperature increases due to insufficient cooling”. In addition, the process-material-cause procedure is automatically attached to it which is expected to capture process material dependent causes, such as ”vessel outlet blocked due to frozen fluid”.

Process material properties are also used for semi-quantitative filtering in HAZOPExpert. An adjustable fire hazard threshold has been defined as follows:

- fire hazard threshold: auto-ignition temperature of process material < 1.5 * operating temperature (F).

Fire or explosion hazard due to flammable material at high temperature is only reported when this threshold has been violated. Additionally, fire or explosion hazard due to the release of process material above its flash point is only reported when operating temperature is greater than the flash point of the process material.

In the case of loss of containment, the health hazards of the process materials and the occupational safety violations due to their release are determined from the process material properties.

In the version of HAZOPExpert extended with Petri nets, Batch HAZOPExpert (Srinivasan and Venkatasubramanian 1996), the study of process maloperation also intends to cover chemical related issues such as the addition of wrong process material and the occurrence of a wrong reaction. To consider the hazards related to the addition of wrong material, the Petri net description of the recipe has to be modified accordingly. Currently, Batch HAZOPExpert cannot conduct analysis of the occurrence of wrong reaction maloperation.

Okayama University

The process-specific knowledge representation of the Okayama University system (Shimada, Suzuki and Sayama 1996) includes a description of the intended reactions. For example, the plant information related to an ammoniation column can be expressed as follows:

column(ammoniation,[‘R1’],[‘NH3’],[‘NaCl_sol’,’NH4OH_sol’,’H2O’]).
reaction(‘R1’,[‘NaCl_sol’,’H2O’,’NH3’],[‘NaCl_sol’,’NH4OH_sol’], exo, con).

The first statement names the reaction which takes place and the gases and liquids present in the column. The second statement lists the reactants and products of the reaction, and defines the reaction as exothermic and condensative.

In their reported work, Suzuki and co-workers have not concentrated on the development of fluid property reasoning techniques which would make use of this process-specific knowledge. The reported usage of this knowledge does not seem to fully exploit the potential of their knowledge representation methodology.

Seoul National University

The Seoul National University researchers recognise the importance of fluid property reasoning in automated hazard analysis (Suh, Lee and Yoon 1997a). They have adopted the NFPA (US National Fire Protection Association) code in their work. Their material knowledge base consists of hazard indices and

reaction matrix data. The hazard indices represent flammability (N_f), health hazard (N_h) and reactivity (N_r) according to the NFPA code. The reaction matrix describes the possible unwanted reactions and their effects.

The reported work on the Seoul system does not discuss the usage of the hazard index and reaction matrix data in detail. It seems that the data is used by the accident analysis algorithm to select the accident scenarios which are relevant under the studied circumstances.

Santa Fe, Argentina

Leone (1996) presents, in association with his object oriented system for hazard identification, an object class definition for representing chemical knowledge for automated hazard identification. In this class, called Compound, attributes related to toxicity, flammability and corrosion are included. No detailed description is given on the use of this knowledge in Leone's object-oriented framework.

4.6 Summary of the state-of-the-art

The main emphasis in all the published research on knowledge-based hazard identification appears to have been on the successuful implementation of the HAZOP procedure with intelligent features for the walkthrough of the process topology and on the identification of the potential faults and their propagation. This is quite natural because the first step in the development of such knowledge.based programs has to be the proper implementation of the basic features such as a library of causal fault propagation models and the reasoning mechanism for its application.

Although a number of research groups have identified the need for fluid property reasoning and other additional features which would improve the quality of the resulting HAZOP study, little effort has been put in the realisation of the recorded ideas. It is concluded that the reported early developments discussed in the previous section are useful primarily as background information for this work and other new developments.

5. Methodology development, phase 1

5.1 Introduction

In this chapter, the first development phase of the proposed fluid property reasoning methodology is discussed. This part of the research was done in a contract research project funded by an engineering company. First, an overview of the project and the HAZOPTOOL program is presented. Next, a method for reasoning on fluid and reaction properties in that framework is described. Separate knowledge bases for chemical substances, chemical reactions and unwanted chemical reactions form the basis for the proposed reasoning techniques. Finally, issues related to the evaluation of the results obtained in this phase of development are discussed.

5.2 Overview of HAZOPTOOL development

5.2.1 Background

HAZOPTOOL was developed at VTT in a contract research project funded by the Taiwanese engineering company China Technical Consultants Inc. (CTCI) Corporation and the local Industrial Development Bureau (IDB). The objective was to develop an intelligent program which gives support to safety analysts for the HAZOP studies of chemical and petrochemical plants.

The expert system functions of HAZOPTOOL were designed to use process information (process structure, chemicals, reactions, etc.) to generate high quality information about the causes and consequences of process deviations, including instructive information about suitable recommendations. If process information is not supplied to the program, then the program was supposed to give efficient support for the documentation of manual HAZOP studies. The HAZOP analyses were to be stored in a structured form to encourage their further utilisation.

The structure of HAZOPTOOL is presented in Fig. 5. The functional modules and their interaction is illustrated in the diagram.

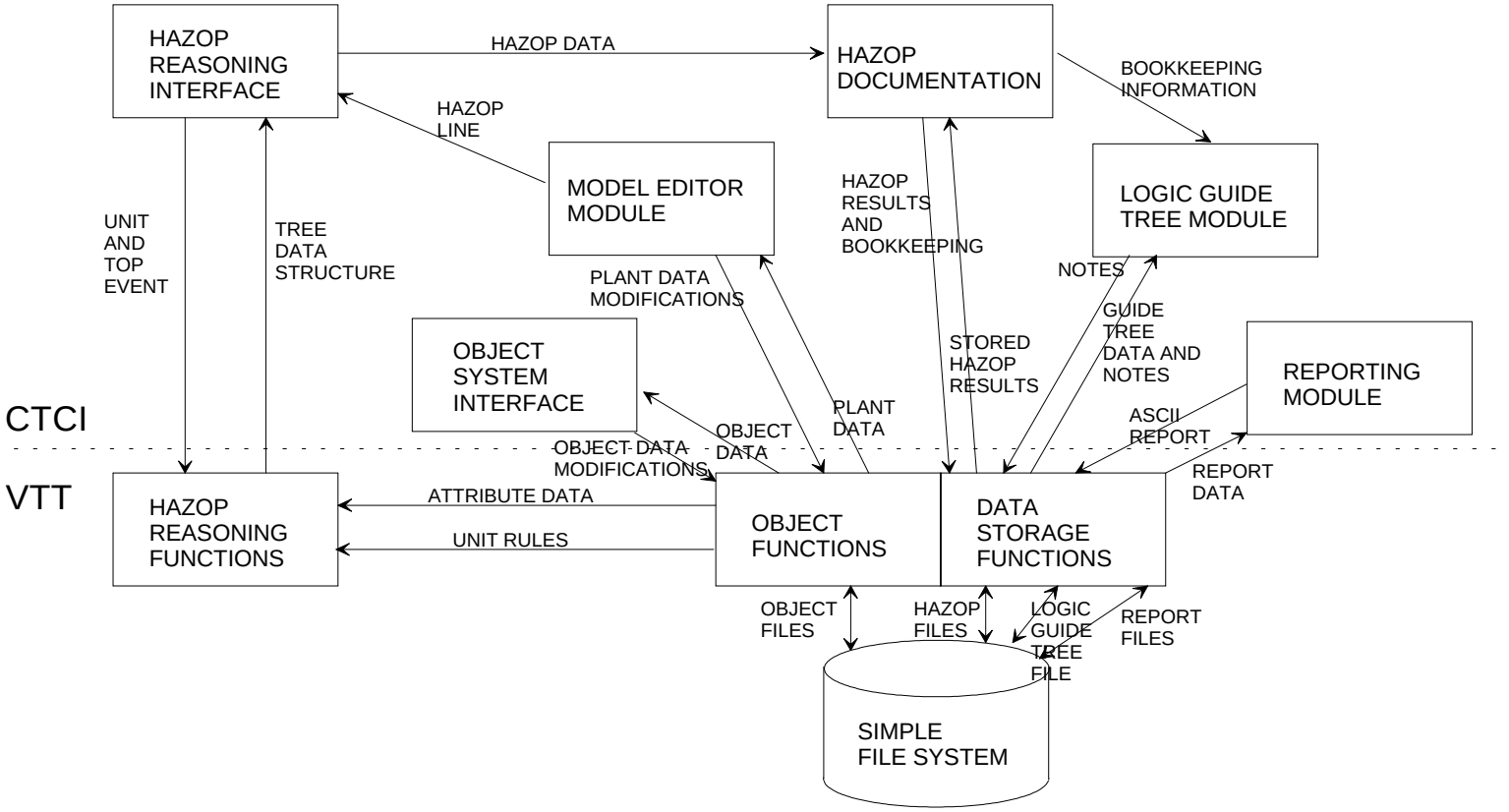


Fig. 5. The structure of HAZOPTOOL.

The model editor module is able to read in the graphics of a process diagram from a CAD system. When the graphics have been loaded, it is possible to define on top of the drawing the corresponding unit instances. The unit instances themselves should be defined using the object system functions, unless a connection exists for reading them in automatically from CAD files. After the unit instances have been defined on top of the drawing, it is possible to divide the process into multiple unit HAZOP lines for the analysis.

The HAZOP reasoning is based on a rule-based process hazard identification method. It is able to identify potentially hazardous event chains based on the plant description. It starts with a process deviation and finds out what are the possible causes and consequences within the same process unit. Some causes will be identified as process variable deviations at the boundary of the unit. Upon request, the causes can be propagated to the neighbouring unit in order to find out the initial causes. That is, the causes in the neighbouring unit which lead to the relevant process variable deviation at the boundary can be retreived. This process can be repeated as necessary.

The potential causes of the studied deviation are presented in tree form, and the potential end consequences of the deviation are listed in a separate consequence pane together with the additional end consequences which could potentially follow from any of the identified causes. The reasoning is done unit by unit so that propagation to the neighbouring unit must be started separately by selecting one of the input events at the bottom level of the tree. Any of the listed consequences can also be selected as the top event of a fault tree, called the consequence tree, which combines all the event chains which could potentially lead to the end consequence within the studied unit. Fig. 6 shows the HAZOP reasoning user interface in an example case.

In the example case, fluid properties were not provided to the program and, therefore, it lists a number of clearly incorrect consequences among the correct ones in the ”Consequence Pane” window. The HAZOP reasoning results including the results of one propagation step (high temperature from 101-R to 102-T) are presented in tree form. The complete text of the selected tree node (highlighted by a dashed surround) is shown in the information window in the upper right corner of the screen.

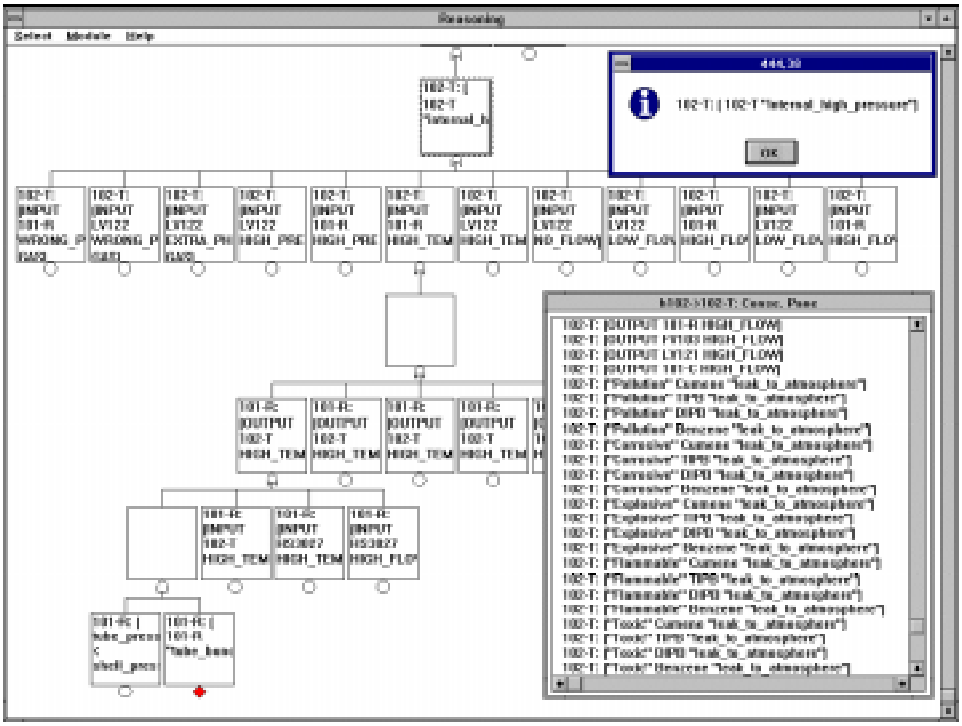


Fig. 6. The HAZOP reasoning interface of HAZOPTOOL.

When a deviation has been studied using the reasoning module, selected parts of the resulting information can be copied to the HAZOP documentation module. The analysis can be documented in two forms: HAZOP trees and HAZOP forms. The module includes a user friendly interface for modifying and extending the information copied from the reasoning module, or for typing in the complete HAZOP without using the reasoning functionality at all. The information recorded using the HAZOP documentation module can be stored in files. These files are then used as the source of information for printing out HAZOP reports.

An alternative view to the analysis process is provided by the logic guide tree module. It contains a predefined tree-formed checklist of the different aspects which should be considered during a safety study. When hazards are studied using the reasoning module and the analysis documentation module, any of the identified events can be linked to a node of the checklist tree. If that is done for the whole analysis, the logic guide tree can in the end be used for producing

summaries of the analysis concerning specific aspects of safety. The user interface of the logic guide tree module in an example case is shown in Fig. 7.

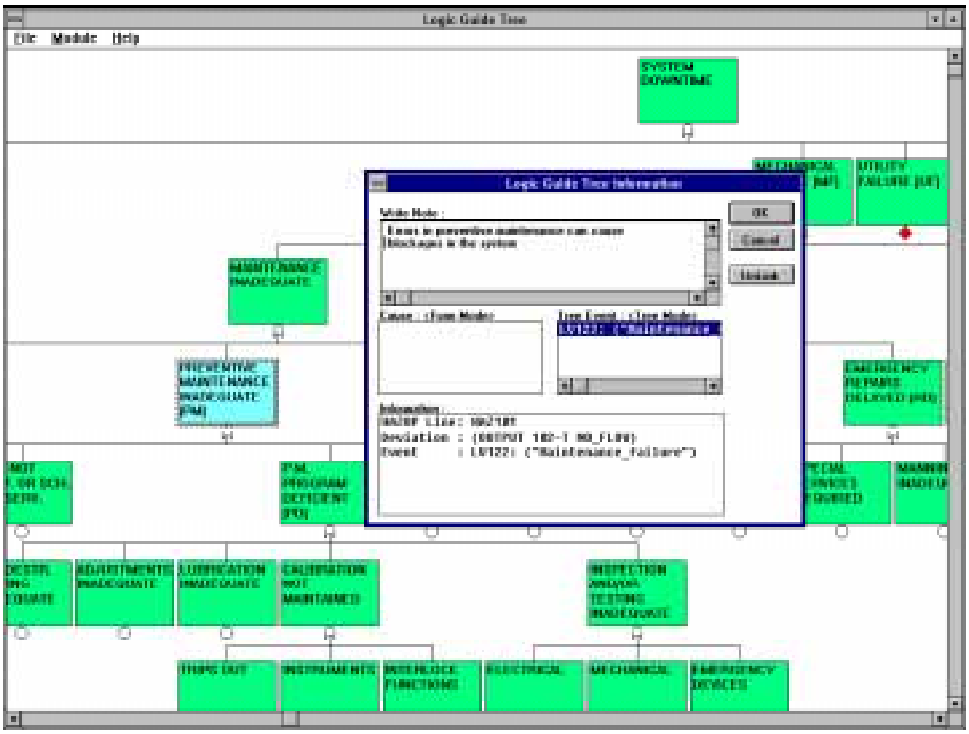


Fig. 7. The logic guide tree interface of HAZOPTOOL.

In this example case, a maintenance related event has been identified as part of the HAZOP reasoning results. This so called tree event has been linked to the appropriate node of the logic guide tree, in this case to the node ”PREVENTIVE MAINTENANCE INADEQUATE”. A note describing the problem has been written by the user in the ”Write Note” pane of the ”Logic Guide Tree Information” window.

5.2.2 Unit level HAZOP models

The heart of a knowledge based hazard identification system is the process-independent knowledge base. Its purpose is to allow the description of the

characteristics of process units in a generic way. Usually this is achieved by constructing a type hierarchy of process units and using it as a framework for describing the actual plant. The plant is described in terms of instances of the class hierarchy which correspond to the actual units of the plant. To describe an actual unit of the process to be studied, an appropriate parent class is selected and the necessary attribute values for the particular unit are filled in.

A class describes a process unit type using attributes and rules. An example of a class in the HAZOPTOOL unit knowledge base is "CONTROL_VALVES". Attributes are used for describing the properties of a unit, for example "normal_flow_rate".

Rules are used to specify the behaviour. Their role will be explained in the next section. The classes can be connected together by assigning a superclass to each new class. As a result, a class hierarchy is generated.

In order to define a class attribute, an attribute definition is added to the class. After this the class and all its subclasses have that attribute. This means that a subclass of a class has all the attributes of its superclass. In addition, the subclass can have attributes of its own. All classes having a certain attribute can be given a default value for it.

The value of an attribute depends on the locally assigned attribute value, default values and inheritance mode. Two inheritance modes are available in HAZOPTOOL. In override mode, the instance is first examined to find a locally assigned value. If no value is found, the search goes on upwards in the class hierarchy until a default value for the attribute is met. The other inheritance mode is union. When union mode is used, all the attribute default values appearing at any level in the hierarchy from the attribute definition class to the parent class of the instance are collected together. In addition to these defaults, the locally assigned attribute value is added to accomplish the final list of values. Rules are defined in a special multi valued attribute for which union inheritance is used.

5.2.3 Representation of hazardous event chains with rules

Knowledge acquisition has crucial importance in the construction of a knowledge based hazard identification system. Knowledge based reasoning can only produce as good results as the quality of the knowledge base permits. To construct a high quality knowledge base of generic knowledge describing the behaviour of process units in abnormal process conditions, a systematic unit HAZOP method was used. The process units were considered separately from any process installation. Fig. 8 shows the unit HAZOP view of a process unit.

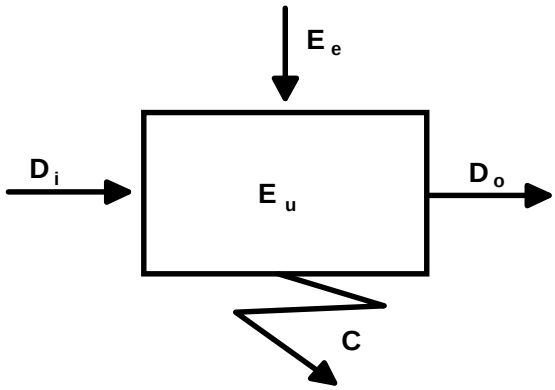


Fig. 8. The unit HAZOP view of a process unit.

D_o is the process deviation at the output of the unit. This can be for example "HIGH FLOW" or "LOW TEMPERATURE". The output deviations can be caused by internal events E_u of the unit, or they can be caused by input deviations D_i or external events E_e. Input deviations and external events can lead to output deviations either directly or through a chain of internal events. Similarly, end consequences C can be caused by internal events, input deviations or external events.

As a result of the unit HAZOP sessions, a set of rules was defined for each unit type. Some of the rules apply to all or many unit classes and can, therefore, be located at a higher level of the unit class hierarchy. The upper level rules are then inherited to the lower levels and can be used together with the locally defined more specific rules.

As an example, some of the rules of the class "CONTROL_VALVES" are listed in Table 1.

Table 1. Example rules from the class "CONTROL_VALVES".

("Control_valve_fails_close") -> (OUTPUT process_flow NO FLOW)
("Inst7rument_air_failure")(type = "Fail_close") -> ("Control_valve_fails_close")
("Control_valve_damage") -> ("Control_valve_fails_close")
("Control_valve_damage") -> ("Leak")
("Leak")("Toxic" process_flow.subst) -> ("Toxic_release")
(INPUT process_flow NO FLOW) -> (OUTPUT process_flow NO FLOW)
("Control_valve_close") -> (OUTPUT process_flow NO FLOW)
(primary_instrument = "Level")(primary_instrument_position = "Upstream")(INPUT primary_instrument LOW SIGNAL) -> ("Control_valve_close")
("Cold_outside_temperature")("Freezing" process.flow.subst) -> (OUTPUT process_flow NO FLOW)

One rule can contain multiple premises, separated from each other with parentheses, but only one conclusion. When multiple premises exist, all of them need to be checked and found to be true before the conclusion will be regarded as true. Physical events such as (”Control_valve_close”), (”Leak”) and (”Cold_outside_temperature”) as well as acceptance conditions such as (type = ”Fail_close”) and (”Freezing” process_flow.subst) can be represented in the form of rule premises. The acceptance conditions tie the generic rules to the actual process information by the use of variables. Above, the variable ‘type’ is used for considering the behaviour of the control valve in detail. The variable ‘process_flow.subst’ is used for triggering fluid related considerations, in this

case in order to access the freezing conditions of the processed fluid and evaluate the need to consider freezing.

5.2.4 Reasoning

The reasoning mechanism of HAZOPTOOL is based on the following operations:

- Backward chaining for causes
- Forward chaining for consequences
- Propagation procedure.

In the backward chaining stage, the deviation expression serves as the starting point. The backward chaining procedure tries to match the deviation expression to the right sides of applicable rules. When there is a match, the process is continued in a depth-first manner, i.e. the backward chaining procedure follows the chain of rules until a statement which does not chain to any other rule is found. Then, it moves to the next unresolved branch until all branches have been studied. The backward chaining procedure has been implemented in a recursive manner. Loop prevention has been incorporated in the procedure.

The backward chaining procedure makes numerous calls to the forward chaining procedure. Consequences are identified in this way for each event of the causal tree generated by the backward chaining procedure. The event expression is matched to the conclusion statements on the right hand side of applicable rules. When there is a match, the process is also in this case continued in a depth-first manner. The events at the end of identified event chains are then collected to the consequence list. Also this procedure has been implemented in a recursive manner, with loop prevention incorporated.

The propagation procedure extends the unit specific backward chaining procedure with the possibility to cross unit boundaries by the further study of input and output events. An input event is propagated to the previous unit and an output event to the next unit. The rules appropriate for the study of this unit are loaded from the process-independent part of the unit knowledge base. Backward chaining is then continued as usual, except that different rule variable names

may be in use in different unit models. Therefore, variables present in the original input or output event may have to be associated with changed variable names. For instance, the flow through the valve may be called ‘process_flow’ in the process-independent rules which describe the valve. If the valve is attached to a line coming out of the hot side of a heat exchanger, the flow out may be called ‘hot_flow’. When the propagation procedure turns its attention from the valve to the heat exchanger, it has to start to apply the rules describing the behaviour of ‘hot_flow’ instead of ‘process_flow’.

5.2.5 Use of unit data in reasoning and presentation of results

The object oriented plant description method makes it possible to describe the characteristics of process units with object attributes. These attributes have a special relation to the rules. Whenever a rule includes a variable, the variable name is compared to the names of the attributes of the currently studied unit. If a matching attribute is found, the variable is replaced with the value of the attribute. If the attribute has many values, a separate event is created for each of them. Because of this variable substitution mechanism, rules can be written which include conditions which depend on the attribute values used for describing the characteristics of the actual process under study. These conditions are then evaluated during the reasoning and invalid event chains can be eliminated from the final results.

Connections between the units are described using the attributes "inputs" and "outputs". Their values combine the names of the neighbouring units and the flowing chemicals. This information is used during reasoning for propagating events correctly between units.

The analysis of deviations in control and support systems is also possible by defining the appropriate control or support inputs and outputs and specifying the flowing entity as "Signal" or "Steam" etc. It is very important to give values to all those attributes which are used within the unit rules to identify the flowing chemical or the chemical reaction taking place. Firstly, this makes the actual chemical and reaction names visible in the final results instead of the variable name. Secondly, the use of chemical and reaction knowledge depends on the availability of this information.

5.3 Method for reasoning on fluid and reaction properties

5.3.1 Use of fluid and reaction knowledge from separate knowledge bases

How events occur and follow each other in a chemical process does not only depend on the properties of process equipment. The properties of chemicals and reactions play a very significant role in the identification of hazardous event chains. The reasoning method described here uses separate knowledge bases for storing this information:

- chemical substance knowledge base (SKB)
- chemical reaction knowledge base (RKB)
- unwanted chemical reaction knowledge base (UWKB).

The chemical substance and chemical reaction knowledge base are used through a predefined set of keywords. For example, in the rule

```
("Leak")(SKB "Toxic" process_subst) -> ("Toxic_release")
```

the chemical substance knowledge base (SKB) is called with the keyword "Toxic" and the name of the processed chemical found from the unit attribute "process_subst". The substance knowledge base returns information about the toxicity of the chemical, including any special conditions which might turn a normally non-toxic chemical into a toxic state. The information is returned in tree form incorporating and-relations to represent the special conditions.

The unwanted reactions knowledge base takes two chemical names as input and returns a list of possible reactions between these chemicals. It is implemented using a reactivity group approach.

The grammar for HAZOP rules (Appendix 1) defines how the HAZOP reasoning procedure can interact with fluid and reaction knowledge. Fig. 9 presents the fluid and reaction reasoning procedure in the form of a flow chart. The information generated by the procedure is normally returned in tree form. However, it is also possible to request the value of an individual property attribute. In that case, this value is returned instead of a tree.

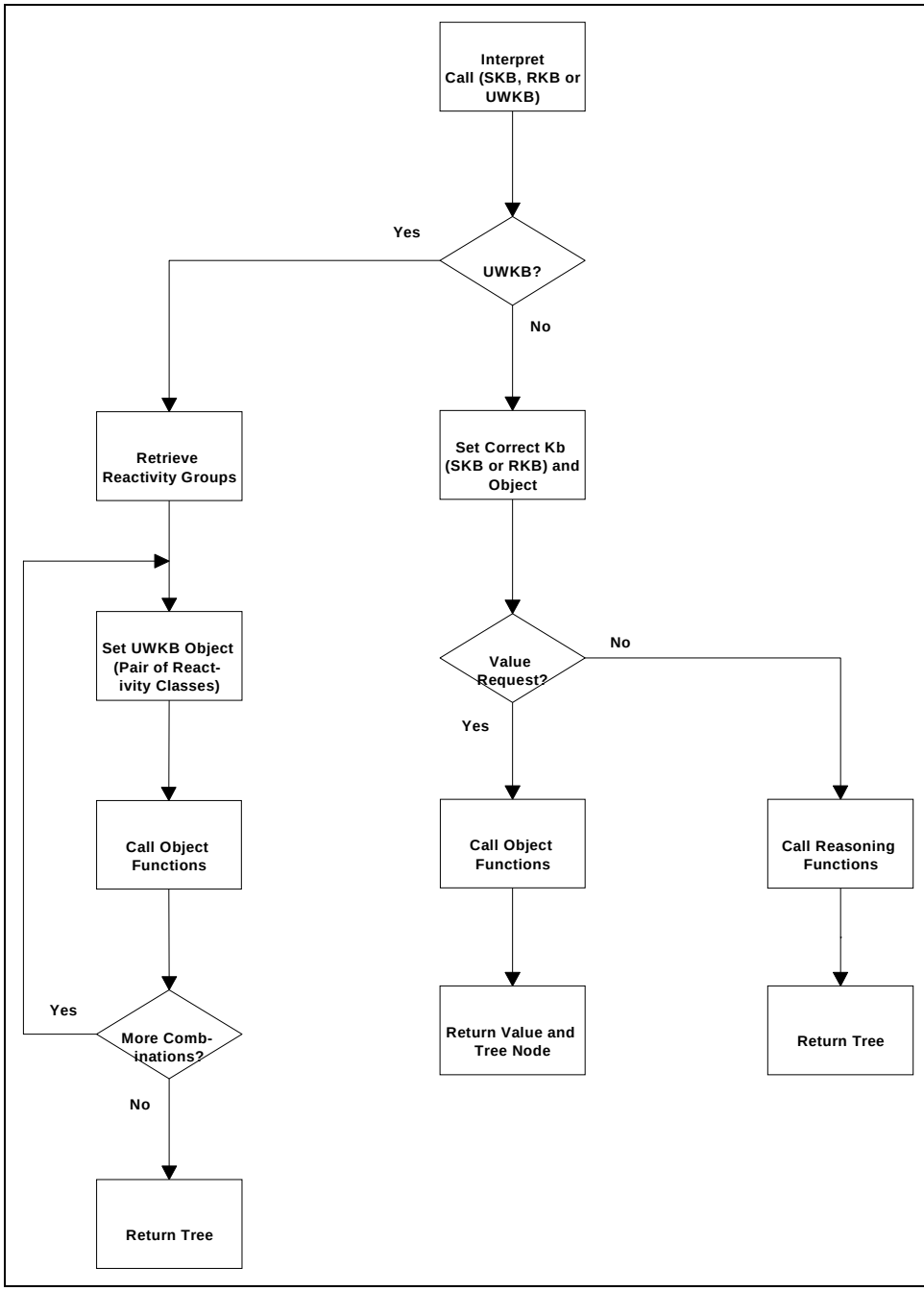


Fig. 9. The fluid and reaction property reasoning procedure of HAZOPTOOL.

The SKB, RKB and UWKB rules can include a number of variables which describe the chemical substances which are present in the unit and certain characteristics of the unit, such as maximum temperature etc. These variables can refer either to local attributes found in the called knowledge base (SKB, RKB or UWKB) or to the attributes of the unit under consideration. If a variable refers to a unit attribute, it is left unevaluated in the returned chemical cause tree and the variable substitution is completed by the reasoning machine after return. The variable substitution mechanism works in such a way that any attributes used as variables in the rules should be called exactly the same in the attribute list and in the rule, and they should be given a value before the reasoning is started. If this is not done, the rules referring to the corresponding variables will be left unevaluated and the generated event chains will be incomplete.

5.3.2 Properties of chemicals

Data and rules about chemical substances are stored in a separate chemical substance knowledge base (SKB). This knowledge base includes a simple reasoning system which is able to respond to certain enquiries by backward chaining the rules. Calls to the SKB can be included in the HAZOP rules by using the keyword ”SKB”, giving the substance name as a constant or variable, and including one of the hazard keywords or the name of an substance attribute.

If there is a need to consider the potential impacts of all substances present in the system, the special variable ‘all_substances’ can be used which has a list of these substances as its value. This can be useful, for example when the plant is screened for potential sources of leaks leading to external fire. This is necessary if the intention is to identify the complete event chain from the causes of the leak to the possibly plant-wide consequences of an external fire. In order to list relevant leak possibilities, the HAZOP reasoning procedure checks the flammability of each substance from the SKB.

Another typical way of using chemical property information in HAZOPTOOL is the use of keywords ”gas”, ”liquid” and ”solid” to identify the appearance of an abnormal phase. The conditions for phase change are formed by processing selected ruled from the SKB. They are then returned for further consideration of actual process conditions by the HAZOP reasoning procedure, and finally by the

user. Table 2 lists typical examples of calls from the unit rules to the chemical substance knowledge base (SKB).

Table 2. Typical examples of calls from the unit rules to chemical substance knowledge.

(input_flow.subst "leak_to_atmosphere")(SKB "Toxic" input_flow.subst) -> ("Toxic" input_flow.subst "leak_to_atmosphere")
(unit_no. "internal_high_temperature")(SKB "gas" input_flow.subst)(unit_no. "gas_outlet_blocked") -> (unit_no. "internal_high_pressure")
(INPUT tube_side LOW_TEMPERATURE)(SKB "solid" tube_side.subst) (tube_phase = liquid) -> (OUTPUT tube_side EXTRA_PHASE SOLID)
(SKB "Fire" all_substances)("Substance_leaks_close_to_unit") -> ("External_heat")
(OUTPUT output_flow LOW_FLOW)(SKB "viscous" output_flow.subst) -> ("Pump_overload")
(SKB "Corrosive" shell_side.subst) -> (unit_no. "shell_damage")

unit_no = unit number.

These rules should only be taken as examples of the use of SKB in reasoning. The keywords to be used for reasoning with the support of the SKB have to be predefined. Therefore, the list of keywords should be re-considered when new rules and attribute lists are designed. New rules and attributes should be added in such a way that SKB is able to give an answer to the request by backward chaining appropriate rules and referring to related attributes.

Although a fixed list of keywords is presented here, the system allows the use of any string of characters as a keyword, and new keywords can be introduced just by adding the corresponding rules.

In the basic version of HAZOPTOOL, the predefined chemical substance keywords are:

- Flammable
- Fire
- Explosion
- Unstable
- Toxic
- Toxic_gas_release
- Toxic_fire
- Vapor
- Gas
- Solid
- Liquid
- Corrosive
- Viscous
- Plugging
- Polymerizing
- Polluting.

In addition, the direct retrieval attribute ”Reactivity_groups” has been defined for representing reactivity group information.

Table 3 presents an extract from the existing set of general level chemical substance rules. The rules presented here should be taken as examples which can later be refined and extended to achieve better reasoning capability. Also substance specific or substance class specific rules can be added. It is possible to include alternative ways of evaluating the property in the rule set. For instance, in the category ”Flammable” rules exist for determining flammability either based on flammability range (flamm_range_%), predefined health hazard category (health_hazard) or a special type of thermal instability (thermal_stability = ”Releases_GF”).

Table 3. Example set of general level chemical substance rules in the SKB.

Flammable
("Gas")(amount = flamm_range_%) -> ("Flammable")
("Liquid")(temp > flash_point_C)(amount = flamm_range_%) -> ("Flammable")
(health_hazard = "Flammable") -> ("Flammable")
(health_hazard = "Highly_flammable") -> ("Flammable")
(thermal_stability = "Releases_GF")("Internal_high_temperature") -> ("Flammable")
Fire
("Flammable")("Unconfined") -> ("Fire")
(health_hazard = "Oxidizing")("Contact_with_combustible_material") -> ("Fire")
("Flammable")(thermal_stability = "Polymerizing")("Internal_high_temperature") -> ("Fire")
Explosion
("Flammable")("Confined") -> ("Explosion")
(thermal_stability = "Explosive")("Internal_high_temperature") -> ("Explosion")
(health_hazard = "Explosive") -> ("Explosion")
Toxic
(health_hazard = "Highly_toxic") -> ("Toxic")
(health_hazard = "Toxic") -> ("Toxic")
(health_hazard = "Corrosive") -> ("Toxic")
(health_hazard = "Noxious") -> ("Toxic")
(health_hazard = "Irritative") -> ("Toxic")
("Toxic_gas_release") -> ("Toxic")
(thermal_stability = "Releases_GT")("Internal_high_temperature") -> ("Toxic")
(thermal_stability = "Decomposes_>_GT")("Internal_high_temperature") -> ("Toxic")

”GF” = ’flammable gas’, ”GT” = ’toxic gas’.

5.3.3 Properties of reactions

Knowledge on typical chemical reactions is stored as a separate chemical reactions knowledge base. This knowledge base uses the same simple backward chaining reasoning mechanism as SKB. Calls to RKB can be included in the HAZOP rules by using the keyword ”RKB”, giving the reaction name as a

constant or variable, and including one of the hazard keywords or the name of an reaction attribute.

Example:

(RKB "Heat_generation" Reaction) -> ("Internal_high_temperature")

As in the SKB, a predefined set of keywords for the RKB has been defined. These keywords are listed below:

- Violent_reaction
- Explosion
- Fire
- Chemical_fire
- Toxic_gas_generation
- Toxic_fire
- Excessive_heat_generation
- Heat_consumption
- Unwanted_by_product
- Wrong_product
- Polymerization.

Examples of general level rules for chemical reactions are presented in Table 4.

Table 4. Examples of general level rules for reactions in the RKB.

(Temperature < Lower_temp_limit) -> (”Accumulation_of_reactants”)
(INPUT catalyst_flow LOW_FLOW) -> (”Accumulation_of_reactants”)
(”Accumulation_of_reactants”)(Heat_of_reaction = ”Exothermic”) -> (”Excessive_heat_generation”)
(Gas_evolution = ”GT”) -> (”Toxic_gas_generation”)
(”Wrong_conditions”)(Reaction_class = ”Nitration”) -> (”Toxic_gas_generation”)
(”Excessive_heat_generation”)(Reaction_phase = ”Liquid”) -> (”Vapour”)
(”Wrong_conditions”)(Decomposition = ”Possible”) -> (”Violent_reaction”)
(Reaction_class = ”Diazotation”)(”Wrong_pH”) -> (”Unwanted_by_product”)

5.3.4 Compatibility of chemicals

A reaction matrix is stored in the system as a separate unwanted chemical reactions knowledge base (UWKB). It is based on a publicly available general purpose reaction matrix (Hatayama 1980) which gives the possible unwanted reactions between members of a collection of reactivity groups. Individual chemical substances can belong to one or more reactivity groups.

HAZOPTOOL will automatically retrieve reactivity group information for the given substances from SKB. UWKB can be called using the keyword ”UWKB”. Table 5 lists typical examples of calls from the unit rules to retrieve compatibility knowledge from the UWKB.

Table 5. Typical examples of calls from the unit rules to retrieve compatibility knowledge from the UWKB.

("Internal_extra_substance" "water")(UWKB "reaction" process_flow.subst "water") -> ("Internal_extra_substance" new_subst)
(INPUT input_flow EXTRA_SUBSTANCE extra_subst)(UWKB "heat_generation" input_flow.subst extra_subst) -> ("Internal_high_temperature")
(INPUT input_flow EXTRA_SUBSTANCE extra_subst)(UWKB "polymerization" input_flow.subst extra_subst) -> (unit_no. "blocked")
(INPUT input_flow EXTRA_FLOW extra_subst)(UWKB "polymerization" input_flow.subst extra_subst) -> ("Internal_high_pressure")
(INPUT input_flow EXTRA_SUBSTANCE extra_subst)(UWKB "violent_reaction" input_flow.subst extra_substance) -> ("Explosion")

The UWKB keywords listed below correspond to the hazard information of this type contained in a reaction matrix:

- Reaction
- Violent_reaction
- Heat_generation

- Fire
- Gas_generation
- Toxic_gas_generation
- Flammable_gas_generation
- Explosion
- Polymerization
- Toxic_solubilization.

These keywords should be considered when new HAZOP rules and attribute lists are designed.

5.4 Evaluation of methodology development phase 1 results

No evaluation of the resulting software was carried out during the HAZOPTOOL project. The choice was made to use all the available resources in the development of more program features.

The only part of HAZOPTOOL which was used extensively during the HAZOPTOOL development project was the knowledge representation methodology and tools. CTCI Corporation generated, for their own use, experience-based rules for about forty process equipment types. However, this was an attempt to cover many equipment types quickly and, therefore, the quality of the knowledge remained far from sufficient.

CTCI Corporation carried out an internal follow-up project of one year which included a case study and subsequent evaluation of HAZOPTOOL. They have kept the results of this follow-up project confidential.

A small case study was carried out at VTT in connection to a European Space Agency (ESA) project. In this case study, unit rules were produced for the main equipment types used in a rocket propulsion system. The purpose of the case study was to demonstrate the application of the techniques of knowledge-based hazard analysis to the space systems engineering community in ESA and to evaluate the usefulness of these techniques for the hazard analysis of space systems.

A vulcain rocket engine developed as an ESA programme was selected as the target system for the case study. The motivation was that the vulcain engine is based on a chemical combustion process and is therefore suitable as the first application of the chemical process oriented knowledge-based methodology. The vulcain engine is described in an article by H. De Boisheraud and S. Eury (1991).

Most of the resources spent in this case study went to the development of the rule based models of the unit types ”Gas Tank”, ”Turbine Pump” and ”Combustion Chamber”. Only the first analysis experiments limited to the combustion chamber could be carried out. To demonstrate the methodology, the top event ”Decreased combustion” was considered. The event chains leading to this event were followed step by step. Fluid issues did not play a significant role in those considerations.

6. Methodology development, phase 2

6.1 Introduction

In this chapter, the second development phase of the proposed fluid property reasoning methodology is discussed. This part of the research was done in a collabortive research project partly funded by the European Commission. First, an overview of the project and the AutoHAZID program is presented. A complete technical description of AutoHAZID can be found in the Final Technical Report of the STOPHAZ project (STOPHAZ 1997). After the overview, a method for reasoning with chemical knowledge in that framework is described. The method is based on presenting the considerations related to fluid properties and interactions as a separate activity with well-defined links to the automated hazard identification process. Finally, the results of a two-stage exercise aimed at the evaluation of the proposed method are presented.

6.2 Overview of AutoHAZID development

6.2.1 Background

The objective of the STOPHAZ project was to promote the safer and more efficient design of process plants through the use of a new generation of quality assured tools. The cost of HAZOP studies and the subsequent redesign activities would decrease significantly if safety is better incorporated into designs at an early stage.

STOPHAZ was project number 8228 of the EU ESPRIT research programme. The tools developed by the project include an intelligent safe design hyperbook, an automated HAZOP tool, and an operating instructions helper. The automated HAZOP tool (AutoHAZID) formed the framework for the phase 2 of the methodology development part of this work.

AutoHAZID is the latest outcome of the research work done in Loughborough University by Prof. F. P. Lees, Dr. A. G. Rushton, Dr. P. W. H. Chung and their co-workers. Their earlier work has briefly been discussed in Chapter 4. The emulation of the HAZOP algorithm in AutoHAZID is based on a signed directed graph (SDG) representation of process equipment faults and their propagation.

A typical user of a tool like AutoHAZID is a design engineer who wishes to evaluate the plant design before commencing with a HAZOP study. Alternatively, it could be used at earlier stages of the design process, before detailed process and instrumentation diagrams (P&ID) are available, to screen for possible problems when the cost of design changes is not too high.

AutoHAZID uses process-independent unit models to create a signed directed graph (SDG) model of fault propagation based on the given process-specific plant description. This SDG model is then examined by applying the HAZOP algorithm. Special fluid model features are used to take into account the impacts of chemical properties. A simplified diagram of AutoHAZID is shown in Fig. 10.

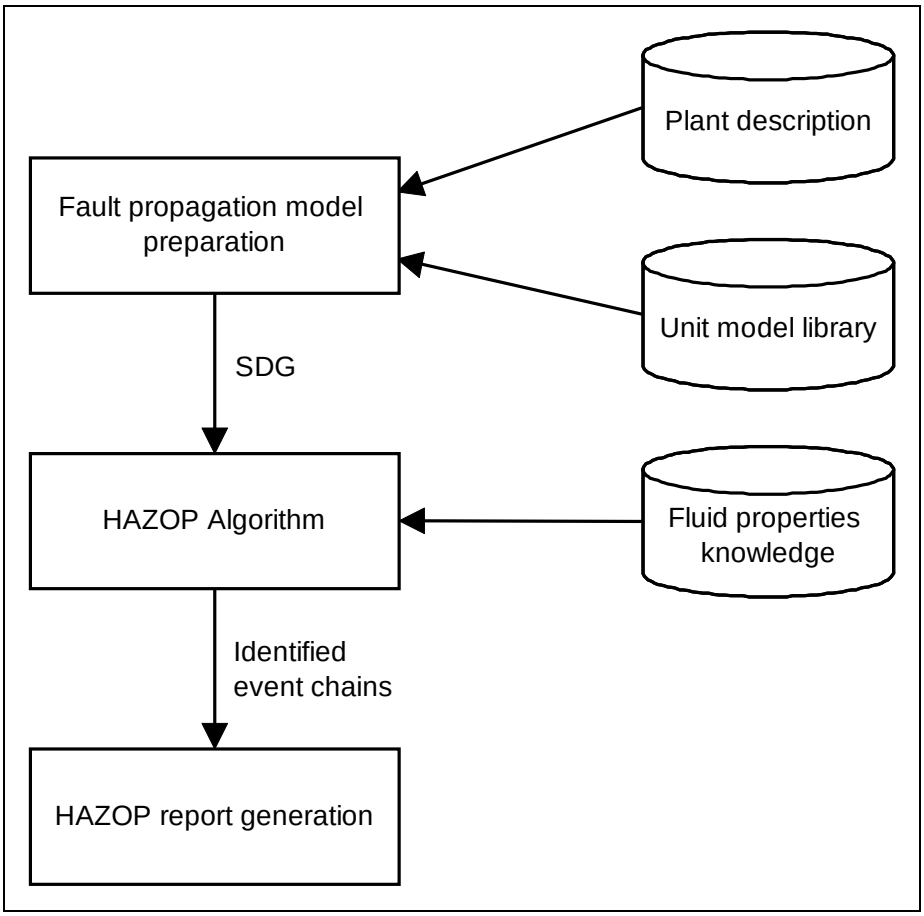


Fig. 10. Simplified diagram of AutoHAZID structure.

The unit modelling methodology, the fault propagation models, the HAZOP algorithm, its application and the subsequent filtering of results will be discussed next. Afterwards, the fluid model system which extends AutoHAZID with a capability for fluid property reasoning will be discussed.

6.2.2 Unit model library

The unit model library file contains qualitative models of each type of process unit which can be represented within AutoHAZID. The models exist within a hierarchy which supports inheritance of information between models.

The fault propagation inference mechanism of AutoHAZID is based on the use of the signed directed graph formalism. The concept of an ”arc” is used for representing a causal relationship. The equipment models of the AutoHAZID unit model library group together the arcs relevant to an equipment type in general.

The object-oriented nature of the frames used to write models for AutoHAZID provides an inheritance mechanism for deriving models in terms of other models. This means that lists of SDG arcs for a child model can be partly inherited from the parent model, and partly derived by differences stated in the child.

Another level of support is the facility available to the equipment modeller to specify that certain blocks of arcs are relevant to a particular equipment model. This is implemented by the use of templates and scripts.

In addition, it is possible to use conditional arcs in the equipment models. They can be used to make the application of a generic equipment model to an actual equipment instance vary depending on the value of an attribute.

The purpose of a unit model is primarily to define the qualitative relationships between deviations, faults and consequences within the unit. The models also fulfil a number of other functions, including:

- definition of the ports present on a unit
- definition of which ports must be connected for correct operation of the unit
- definition of default attribute values.

Table 6. Example of a unit model: reciprocating pump.

```
frame(reciprocatingPump isa pdPump,[
propLinks include [
  % LOSS OF DRIVE ARCS:
  arc([fault,'loss of drive'],1,[out,noFlow]), % (5)
  % CHECK VALVE ARCS:
  arc([fault,'check valve fails shut'],1,[out,noFlow]), % (6)
  arc([fault,'inlet check valve damaged or worn'],-1, [in,pressure]), % (7)
  arc([fault,'outlet check valve damaged or worn'],-1, [out,pressure]), % (8)
  arc([fault,'inlet check valve fails open'],1,[in,revFlow]), % (9)
  arc([fault,'outlet check valve fails open'],1,[out,revFlow]), % (10)
  arc([fault,'outlet check valve fitted in wrong direction'],1,
    [consequence,'pump failure']), % (11)
  % SPILL-BACK ARCS:
  arc([fault,'spill-back occurring'],-1,[in,flow]), % (16)
  arc([fault,'spill-back occurring'],-1,[out,flow]), % (24)
  arc([fault,'spill-back occurring'],1,[in,temp]), % (17)
  % MISCELLANEOUS
  arc([fault,'internals damaged'],-1,[out,pressure]), % (18)
  arc([fault,'internals damaged'],1,[out,solids]), % (23)
  arc([fault,'internals damaged'],1,
    [consequence,'particulates to downstream units']), % (20)
  arc([deviation,[morePressure,out]],1,[consequence,
    'casing or pipework overpressure if relief system fails']), % (21)
  arc([deviation,[noFlow,out]],1,[consequence,
    'casing or pipework overpressure if relief system fails']) % (22)
  ],
  % Conditional section.
  conditionLinks info [
   [ driver_type is variable, [
      propLinks include [
       arc([fault,'high stroke speed'],1,[out,pressure]), % (12)
       arc([fault,'high stroke speed'],-1,[in,pressure]), % (13)
       arc([fault,'low stroke speed'],-1,[out,pressure]) % (14)
      ]]],
   [ status is spare, [
       script info [ leak_out(in,out),
             leak_vacuum(in,out)],
                                    propLinks info [
        arc([fault,'spare unit turned on'],1,[out,flow]), % (30)
        arc([fault,'spare unit turned on'],1,[out,pressure]), % (31)
        arc([fault,'spare unit turned on'],-1,[in,pressure]) % (32)
     ]]]
  ]
 ]
).
```

The version of the unit model library which was released in connection to the final prototype of STOPHAZ includes the models of over 50 detailed types of equipment. Examples of equipment types at this detailed level are distillation column, three phase gas-liquid-liquid separator, reciprocating pump, and signal splitter. These detailed models are grouped in a hierarchy under nine top level equipment types, such as vessel, pressure raiser and instrument. The reciprocating pump model is presented as an example in Table 6.

6.2.3 Fault propagation models

The qualitative propagation information recorded in the unit models is the core of AutoHAZID's analysis technique. Unit models define qualitatively:

- how deviations in process variables propagate through a unit
- the faults which can occur within a unit and their effect on process variables
- the adverse consequences of deviations and faults within a unit.

There are four types of process arcs within the unit model library. These represent different types of propagation links as illustrated in Table 7. In an arc, one cause and one effect are linked together by a positive or negative influence. This is represented using a simple syntax. For example, arc(X, 1, Y) means that the occurrence or increase of X leads to the increase of Y, and arc(A, -1, B) means that the occurrence or increase of A leads to the decrease of B.

Table 7. AutoHAZID propagation link types.

	1. variable ➔ variable Example:
arc([out, resistance],-1,[out,flow]) • Increased (decreased) resistance at outlet leads to decreased (increased) flow	
	2. fault ➔ variable Example:
arc([fault,'high pressure upstream'],1,[out,flow]) • High pressure upstream leads to increased flow at outlet	
	3. deviation ➔ consequence
Example: arc([deviation,[moreTemp,out]],1,[consequence,'pump casing overtemperature',[haz_op,2,[ed,lc]]]) • High temperature leads to pump casing overtemperature (hazard and operability problem with default severity 2 which may lead to equipment damage and loss of containment)	
	4. fault ➔ consequence.
arc([fault,'leak to environment'],1,[consequence,['fire/explosion risk',flammable(out)],[haz,2,[lc]]]). • Leak to environment, in case the fluid is flammable, leads to fire/explosion risk (hazard problem with default severity 2 which may lead to loss of containment).	

6.2.4 HAZOP algorithm

When a HAZOP study is carried out in a conventional manner, a systematic procedure is followed. The process is examined unit by unit or line by line, and all possible process variable deviations are considered. The intention is to identify:

- possible faults which could cause the process variable deviation

- potentially hazardous consequences of the deviation and its causes.

In principle, AutoHAZID intends to emulate the conventional HAZOP procedure, in a similar manner as has been illustrated in Fig. 3 in Chapter 4. However, it is computationally unnecessarily expensive to search for the desired information in exactly that way.

During an exhaustive HAZOP style graph search, all the paths which can cause a change are found first, and those which do not have the appropriate influence are then discarded. Effort is wasted in repeated graph search when the opposite deviations of the same process variable in the same unit or line are considered. Additionally, process variable deviations sometimes chain to each other which also causes similar unnecessary repetition of the graph search.

The AutoHAZID HAZOP algorithm tries to reduce the amount of wasted search effort by considering all deviations at once, instead of one at a time in isolation. This is done by operating with the whole list of deviations during graph search and partitioning the results for individual deviations afterwards. The algorithm is illustrated in Fig. 11.

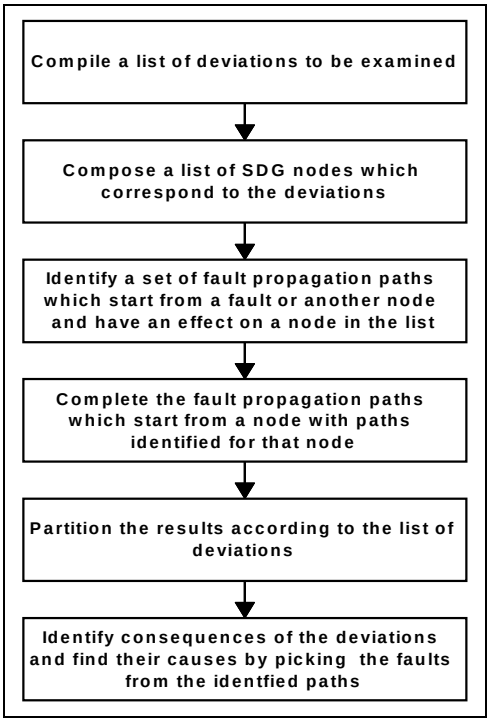


Fig. 11. AutoHAZID HAZOP algorithm.

6.2.5 Filtering of results

It is possible to consider the validity or relevance of a specific fault propagation path either during the execution of the HAZOP algorithm or afterwards. Considerations related to removing some entries from the initial results produced by the HAZOP algorithm are called filtering.

AutoHAZID contains three filtering procedures:

- Combine multiple similar failures in a particular line section into a single fault
- Remove repeat scenarios found under different deviations
- By stopping fault propagation chains at predefined stopping points, mimic references to deviations as used in traditional HAZOP reports.

The current implementation of filtering in AutoHAZID is relatively limited. Many context-related knowledge based considerations could be implemented as filters. These filters could take as input the initial set of results produced by the HAZOP algorithm and carry out sophisticated reasoning to screen out irrelevant entries. This would simplify the operations done by the HAZOP algorithm by postponing the consideration of more complicated issues.

For instance, some tasks related to fluid property reasoning could conveniently be performed as filtering tasks. This makes filtering particularly interesting in the context of this work.

6.3 Reasoning with chemical knowledge

6.3.1 The fluid model approach

Objectives

The main objective of the fluid model system is to extend AutoHAZID with chemical and material knowledge and with the necessary techniques to utilise this knowledge in such a way that the quality of the HAZOP results generated by AutoHAZID is improved:

- more relevant hazards (feasible, likely and important) are identified
- fewer irrelevant hazards (infeasible, unlikely or unimportant) are suggested.

The aim is also to improve the efficiency of AutoHAZID. This can be done by preventing the inference machine from exploring irrelevant branches of the causal chains. Another way is to speed up the output filtering process.

It should be noted that the main objective should be achieved without causing too severe problems for the efficiency of AutoHAZID and thus weakening the improvements related to the second objective.

Earlier work by various research groups (see section 4.4) has indicated that fluid property reasoning has the potential to improve knowledge-based hazard identification in the desired way. In practice, a system such as AutoHAZID based on qualitative models of process equipment behaviour needs to be extended with new methods. These include a method for resolving fluid property related queries and a method for performing fluid compatibility checks.

Functional requirements

The approach used in the definition of requirements for the fluid model was twofold. It was important to build on previous work and the lessons learnt. The earlier work by VTT and LUT has been discussed in section 4.4.

In chapter 5, the first phase of this work on fluid property reasoning methodology development was discussed. The rule-based approach applied in the first phase was considered appropriate also for the purposes of the second stage, but only if completely re-implemented. In addition, the experience gained in the first phase from the use of a reaction matrix with reactivity groups indicates that fluid compatibility problems could be efficiently studied by a similar approach.

On the other hand, it was necessary to make sure that such issues are taken under consideration which appear in industrial hazard identification work. In order to do this, a number of HAZOP studies produced in a traditional way by a group of human experts were examined.

Typical sequences of fluid property related questions made during a HAZOP study are:

- How high a temperature is needed to turn liquid into gas at the given pressure? Is such a temperature increase possible?

- What is the flammability range of a gas? Is it possible that the air concentration in a vessel becomes such that the mixture is inside that range? Is it possible that the outside process concentration of the gas goes within the range at an indoors located part of the process?

- What is the decomposition temperature of a fluid? Is such a temperature increase possible? What are the properties, compatibility and hazards of the decomposition products?

- What is the compatibility of the reaction with extra components? Do other reactions take place in that situation? What are the reaction products and their properties, compatibility and hazards?

Appendix 2 contains a list of fluid model related problems extracted from the studied HAZOP reports. The associated list of questions to be answered by the fluid model is presented in Appendix 3.

A key requirement for the fluid model system is that it should be able to respond to fluid property related queries. If a cause-consequence link holds (or the event is relevant) only under certain conditions, these conditions should be considered by the fluid model. In case the conditions can not be evaluated based on available information, the unevaluated conditions should be returned as part of the reply. This allows the scenario to be reported as conditional on stated facts that could not be evaluated.

As follows from the requirements, the fluid model system is expected to add intelligence to AutoHAZID by providing the following features:

- rule based evaluation of fluid properties
- access to the properties of fluids in the specified deviated state of the process
- access to the properties of mixtures
- identification of fluid incompatibility problems.

The full use of the above features in connection to AutoHAZID unit models has the potential to improve its performance considerably.

The fluid property queries which were considered necessary in order to support the first version of the AutoHAZID model library were listed as the minimal requirement. The first version of the unit models was not fluid-oriented, and some interesting topics, such as reactions and fluid composition changes have been left out from the list. Although the list does not cover all the aspects of fluid property reasoning, it still gives a good overview of what can immediately be taken into use in connection to qualitative fault propagation models. The list of queries to be supported by the fluid model system is presented in Table 8. The proposed methods for resolving the queries have also been listed. How the internal library of fluid properties and external fluid properties packages are used in AutoHAZID to support these methods will be described in more detail in the next part of this section.

The implementation of some of these queries in AutoHAZID remains partial for a variety of reasons. The state of implementation is illustrated in the ”Implementation” column of Table 8 as follows:

A. The query is related to a fluid property which should be determined for the actual fluid composition (mixture) and the specified process conditions. Both the composition and process conditions may depend on the fault under consideration.

B. The query is related to the compatibility of the fluid components. The reaction matrix which supports these considerations needs to be linked to fluid model queries.

C. The method for resolving the query and the representation of the associated fluid properties need more detailed specification.

Table 8. Supported queries.

Query	Method	Implementation
boiling	temp > boiling pt	A
brittle	material property look-up	
corrosive	fluid property look-up	C
decomp	fluid property look-up	
dissolvedGas	fluid composition	A, C
explosion	reaction matrix	B
flammable	fluid properties and rules	A
flashRelease	fluid properties and rules	
freezing	temp < freezing pt	A
gasGeneration	reaction matrix	B
heatGeneration	reaction matrix	B
liquid	temp > freezing pt	A
liquidHammer	liquid or condensate present	C
nearBoilingPt	temp close to boiling pt	A
plugging	viscous or polymerizing	
polluting	fluid property look-up	
polymerizing	temp & fluid property	
pressurised	unit press > atm press	
solid	temp < freezing pt	A
solidsPresent	fluid composition	A
toxic	fluid property look-up	
unstable	fluid property look-up	
vacuum	unit press < atm press	
vapour	temp > boiling pt	A
viscous	fluid property look-up	

Architecture of AutoHAZID fluid model system

The queries to be resolved by the fluid model system require a variety of methods to be supported. The considerations involve the processing of both qualitative and quantitative information on fluid properties. A rule-based

architecture was selected because it provides sufficient flexibility to be used for relatively ill-defined problem solving such as fluid property reasoning. The architecture of the fluid model system is shown in Fig. 12.

The blocks with a shadow in Fig. 12 represent the modules belonging to the fluid model system. The roles of these modules are described below.

Query Handler

The queries to be resolved are initiated by the AutoHAZID fault propagation engine. This engine uses models of process unit fault behaviour to simulate fault propagation through a system of process equipment. The models are signed directed graphs which can have conditions attached to the arcs. These conditions often refer to the properties of the process fluids. In such situations, the fluid model system can be used for evaluating the conditions.

The first step of resolving a fluid property query is to accept a request from AutoHAZID. Then the fluid rules stored in the fluid library are checked in order to find rules which match with the query. If matching rules are found, the problem is passed to the backward chaining inference engine. The inference engine returns the result in the form of a tree which is then transformed into the form of a string-based conditional reply and returned to AutoHAZID.

Inference Engine

The fluid model inference engine follows a backward chaining procedure based on string comparison. It uses the given problem as the top goal and looks for rules where the right hand side matches to the problem. For each matching rule it then looks at the left hand side and sets each of the left hand clauses as the next goals. The results of this backward chaining process are recorded as a logical tree. Since the clauses can contain variables which refer to fluid properties, the inference engine calls a rule parsing routine for each clause. When the backward chaining procedure has been completed, the inference engine carries out an evaluation of the tree. This evaluation process may find out that part of the tree is not relevant because some of the clauses which refer to fluid properties evaluate to false.

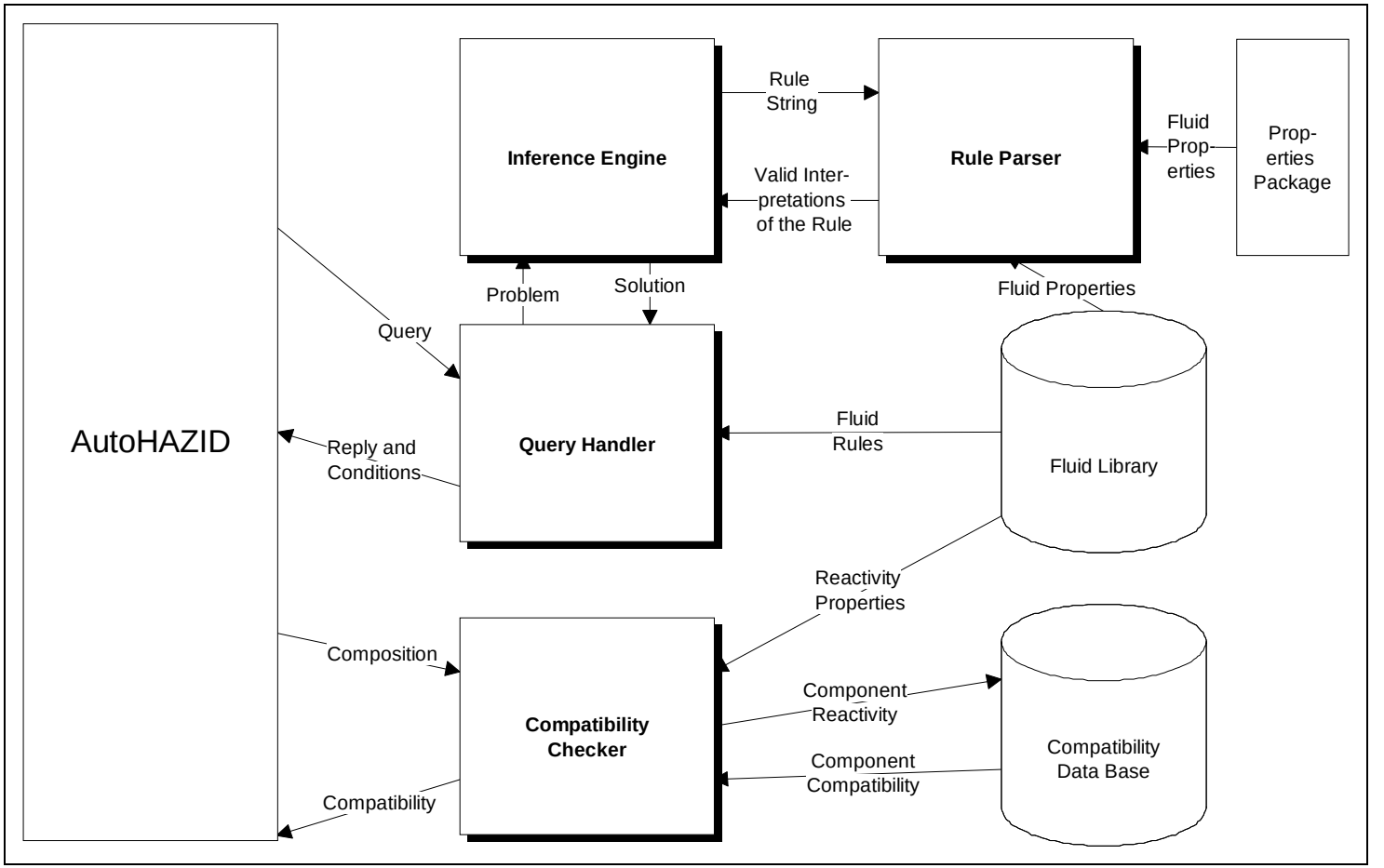


Fig. 12. Architecture of the fluid model system.

Rule Parser

The fluid model rule parser takes a rule string as an input and parses it in accordance to a simple grammar. When it detects a variable, it tries to assign a value for it either through a request to the properties package or from the internal static fluid library data. After all possible variable substitutions have been done, the revised rule is returned to the inference engine.

Compatibility Checker

The compatibility check feature of the fluid model system is used by AutoHAZID separately from fault propagation. The fluid information handling routines of AutoHAZID call the fluid compatibility check when a potential for the interaction of two fluids has been identified.

The compatibility checker accepts two fluid descriptions as input from AutoHAZID. Looking at each component of both fluids, it checks all the possible combinations. For each pair of components, reactivity properties are retrieved from the fluid library. The reactivity properties of a chemical are described by membership in a number of reactivity groups. The possible compatibility problems for each pair of reactivity groups are found in the compatibility data base. When the compatibility checker has gone through all components and the associated reactivity groups for both fluids, the resulting compatibility problem data is returned to AutoHAZID.

6.3.2 Properties of fluids

Internal library of fluid properties

Fluid property information is used for the evaluation of fluid rules. The fluid model system obtains fluid property data from two sources: the internal fluid library and the external properties package. The static fluid property data stored in the fluid library is used when it is not possible to get a property value from a properties package.

Only a limited set of queries can be supported with static chemical properties data. These queries relate to such fluid properties which do not change when process conditions or the concentrations of fluid components change.

Toxicity is a good example of a property of this kind. It does not normally change unless new components are introduced to the fluid either by the addition of extra components or as a result of a chemical reaction. Properties such as flammability and potential for decomposition or polymerisation can also quite reliably be examined using static property data on the fluid components.

The AutoHAZID fluid model system uses a library file to store the fluid rules and fluid property data. Table 9 lists the fluid properties stored in this static library file.

Table 9. Fluid properties stored in the internal library.

1	CAS number
2	molecular weight
3	freezing point (Celsius)
4	decomposition temperature (Celsius)
5	flash point (Celsius)
6	auto-ignition temperature (Celsius)
7	lower flammability limit (%v/v)
8	toxicity classification
9	latent heat (at normal boiling point in J/mol)
10	normal boiling point (ie. at ambient pressure in Celsius)
11	Antoine Const A (for vapour pressure calculation)
12	Antoine Const B (for vapour pressure calculation)
13	Antoine Const C (for vapour pressure calculation)
14	Liquid density (kg/m^3) (temperature dependence is not represented)
15	Vapour heat capacity constant, A (J/mol)
16	Vapour heat capacity constant, B (J/mol)
17	Vapour heat capacity constant, C (J/mol)
18	Vapour heat capacity constant, D (J/mol)
19	ERPG 3 values (measure of toxicity for Dow Exposure Index, mg/m3)
20	Liquid heat capacity (J/(kg.K))
21	Reactivity group 1
22	Reactivity group 2
23	Reactivity group 3
24	Reactivity group 4

The fluid property library is structured as a frame based system. The hierarchy has two levels. The top level can be used for specifying general information which holds for any chemical. The actual values of fluid properties are always specified at lower levels. Therefore, the default value for them at the top level is 'None'. However, all the existing fluid rules are general purpose rules and they are specified at the top level. This means that the same set of fluid rules will be inherited to every lower level frame representing an actual fluid. Table 10 presents the top level frame with part of the general level rules shown, and an example of a lower level frame.

In the version of AutoHAZID released in connection to the final prototype of STOPHAZ, the fluid library contains property data for the pure chemicals listed in Table 11: This prototype implementation does not attempt to fully cover the property data requirements of different applications. Instead, it is expected that the users of this version of AutoHAZID are prepared to check the correctness of the available data and extend the data when necessary. In particular, the handling of issues such as chemicals with multiple isomers are left to the responsibility of the user.

Table 10. Part of the top level frame from the fluid property library and an example of a lower level frame.

```
frame(fluiid isa substance,
  [property info [],
   physicalData info [],
   fluidData info
   ['None','None','None','None','None','None','None',
                'None','None','None','None','None','None','None',
                'None','None','None','None','None','None','None','None','None','None'],
   /* Rules for resolving queries based on internal fluid property data */
   ruleData info ['(temperature > boilPt)("vapFlammable") -> ("flammable")',
   '(temperature < boilPt)("liqFlammable") -> ("flammable")',
   '(flashPt > "-300.0")(temperature > flashPt) -> ("liqFlammable")',
   '(tox > "1") -> ("toxic")',
      ... {more here}
   /* Rules for interpreting the replies of the property package to the physical state
     query */
   /* If the property package can not resolve the query (reply is "0"), look at the
     the default boilPt value given for this fluid */
   '(phystate = "0")(temperature > boilPt) -> ("vapour")',
   /* The replies "1", "3", "5" and "7" indicate the presence of vapour phase,
     with or without the simultaneous existence of other phases */
   '(phystate = "1") -> ("vapour")',
   '(phystate = "3") -> ("vapour")',
   '(phystate = "5") -> ("vapour")',
   '(phystate = "7") -> ("vapour")'
   ]
]).
frame(hexane isa fluiid,
  [
   fluidData info
   ['110-54-3','86.177','-95.3','None','-22','260','1.1','1','28872',
   '68.7','15.8089','2654.81','-47.3','673','-1.746','58.089E-2',
   '-2.903E-4','60.541E-9','None','2227','29','35','None','None']
]).
```

Table 11. The pure chemicals covered by the fluid property library.

Component ID	**Formula**	**IUPAC Name**
ammonia	NH_3	ammonia
ammonium sulphate	$(NH_4)2SO_4$	ammonium sulfate
benzene	C_6H_6	benzene
butane	C_4H_{10}	butane
butene	C_4H_8	butene
carbon dioxide	CO_2	carbon dioxide
carbon monoxide	CO	carbon monoxide
chlorine	Cl_2	chlorine
ethane	C_2H_6	ethane
ethene, ethylene	C_2H_4	ethylene
ethylene dichloride	$C_2H_4Cl_2$	1,2-dichloroethane
Freon-11, trichlorofluoromethane	CCl_3F	trichlorofluoromethane
Freon-12, dichlorodifluoromethane	CF_2Cl_2	dichlorodifluoromethane
hexane	C_6H_{14}	hexane
hydrogen	H_2	dihydrogen
hydrogen chloride	HCl	hydrogen chloride
hydrogen cyanide	HCN	hydrogen cyanide
hydrogen sulphide	H_2S	dihydrogen sulfide
methane	CH_4	methane
nitrogen	N_2	dinitrogen
oxygen	O_2	dioxygen
propane	C_3H_8	propane
propene	C_3H_6	propene
sodium hydroxide	NaOH	sodium hydroxide
sulphur dioxide	SO_2	sulfur dioxide
sulphuric acid	H_2SO_4	sulfuric acid
toluene	$C_6H_5CH_3$	toluene
water	H_2O	water

External chemical properties packages

Properties packages provide an alternative source of fluid property information to be used for the evaluation of the fluid rules. The properties packages available in AutoHAZID cover the same selection of pure chemicals as the static fluid library (see Table 11). In addition, they provide access to the properties of mixtures.

Properties packages have two major advantages over a static data base of fluid property data. Firstly, the fluid properties are calculated for the actual process temperature and pressure instead of giving information which refers to some standard conditions. Secondly, the properties packages can handle mixtures which is impractical for a static database because the mixture ratio can vary.

In analysing an actual chemical process plant, these capabilities are of particular interest. What is required from a proper fluid model system, is not the capability to consider pure chemicals under standard conditions. On the contrary, the capability to consider mixtures of chemicals under various process conditions is absolutely necessary.

State of matter is a very interesting issue. It is a property which tells us whether the fluid is gas, liquid or solid. Vapour is handled as gas in this context. Even in the case of pure chemicals, the actual process conditions have to be examined and the behaviour of the fluid has to be known in order to find out what the state of matter is. In the case of mixtures, several phases are often present at the same time. Moreover, the behaviour of the mixture may differ from the combined behaviour of its components.

In the version of AutoHAZID released in connection to the final prototype of STOPHAZ, links to two properties packages were included:

- PROPERTIES PLUS by AspenTech Inc, and
- HYSYS Property System by Hyprotech S. L.

The two properties packages provide equal services to AutoHAZID. The property requests handled by the properties packages are listed in Table 12. Some of the requests can only be made for pure chemicals and not for mixtures.

These requests are marked with ”Pure” in the right hand side column of the table.

Table 12. Property requests handled by the properties packages.

Physical State	
Molecular Weight	
Boiling Temperature	
Freezing Temperature	
Vapour Pressure	
Viscosity	
Density	
Specific Entalphy	
Specific Heat	
Latent Heat of Fusion	
Latent Heat of Evaporation	
Temperature Limit of Phase Change	
Pressure Limit of Phase Change	
Flash Point (closed cup)	Pure
Auto-Ignition Temperature	Pure
Lower Flammability Limit	Pure
Upper Flammability Limit	Pure
CAS Number	Pure

The properties packages accept as input a description of the fluid composition as a combination of three parameters:

- Component List
- Component Mole Fractions
- Number of Components.

In addition, the process conditions in the form of pressure and temperature are given. One or both of them can be omitted when appropriate.

Chemical reasoning rules

Fluid property queries are made in order to assess the relevance of a fault or a consequence for the currently studied fluid. The fluid rules for resolving the queries are all general purpose rules which refer to the properties of the fluid.

As an example, general purpose rules for evaluating whether a vapour is flammable are presented in Table 13. If valid values of autoignition point and lower flammability limit are available, the rules check whether the vapour concentration and unit temperature are high enough to lead to a flammability hazard.

Table 13. Fluid rules for checking vapour flammability.

(unitTemp > boilPt)("vapFlammable") -> ("flammable")
(unitTemp = boilPt)("vapFlammable") -> ("flammable")
(autoIgnitionPt > -300.0)(unitTemp > autoIgnitionPt) -> ("vapFlammable")
(lowFlamLim > -1.0)(vaporConcentration > lowFlamLim) -> ("vapFlammable")

autoIgnitionPt <= -300.0 means undefined, lowFlamLim <= -1.0 means undefined.

The rule grammar is a simplified version of the HAZOPTOOL rule grammar presented in Appendix 1. This is a rather primitive grammar, for instance, or-relations are not allowed in this simple grammar. They have to be represented as separate rules which have the same conclusion as the right side of the rule. However, even simple rules are a much more convenient representation formalism than conventional programming languages. Table 14 presents the equivalent reasoning in the form of c-language code.

Table 14. Equivalent reasoning in c-code form for the rules given in Table 13.

```
if(prop==gst.symbol("flammable"))
{
  int returnVal=1;    // default = flammable possible.
  if(unitTemp>boilPt)
    returnVal = isVapFlammable(vapConc,unitTemp,flu);
  else …  // routine for liquid flammability
}
int isVapFlammable(double vapConc,double temperature,Fluid*flu1)
{
  double LFL,AIT;  // AIT - Auto-Ignition Temperature
  AIT = flu1->autoIgnitionPoint();
  if (AIT != -999)          // AIT == -999 means: can't auto-ignite
     if(temperature>=AIT) return 1;
  LFL = flu1->lowerFlamLim();
  if (LFL != -999)          // LFL == -999 means: has no LFL, can't ignite
     if(vapConc>=LFL) return 1;    // is flammable
    else return 0;   // is non-flammable
  else return 0;   // is non-flammable
 }
```

The rule-based approach makes it possible to make the definitions in a clearer and more declarative way. In addition, the rules can be modified without recompiling the program code.

Reasoning mechanism

The fluid rules are used by the reasoning machine which backward chains them for finding the conditions under which the fluid property in question may hold. The reasoning starts with the query keyword which is matched to the right hand sides of all relevant rules. The left hand sides of the matching rules form the set of first level conditions and each of them is then matched to the right hand sides of the rules to find the next level of conditions. This goes on as far as there are more detailed conditions to be found. As the result of this recursive procedure, a tree representation of the property evaluation logic is generated.

The rules can include a number of variables which describe the properties of the fluid and the current process conditions. These variables can refer either to fluid properties within the scope of the properties package, fluid properties outside its scope, or to process conditions passed to the fluid model system as part of the query. If a value is found, the variable is replaced with the value. Otherwise, the condition expression remains unchanged.

Once the variables have been substituted with values, the conditions containing one of the operators ”=”, ”!=”, ”>” and ”<” will be evaluated. Then, the truth value of the result tree is determined taking into account the logical relations between the conditions as defined in the fluid rules.

It should be noted that the truth value of the result tree will always be TRUE if one or more conditions in its branches are left unresolved. In such situations, the unresolved conditions are marked in the tree.

The result of the fluid property reasoning is a tree which represents the conditions under which the fluid property may hold, and the logical relations between the conditions. A truth value is associated with each node of the tree, and unresolved conditions are marked.

The fluid property reasoning mechanism is illustrated in Fig. 13.

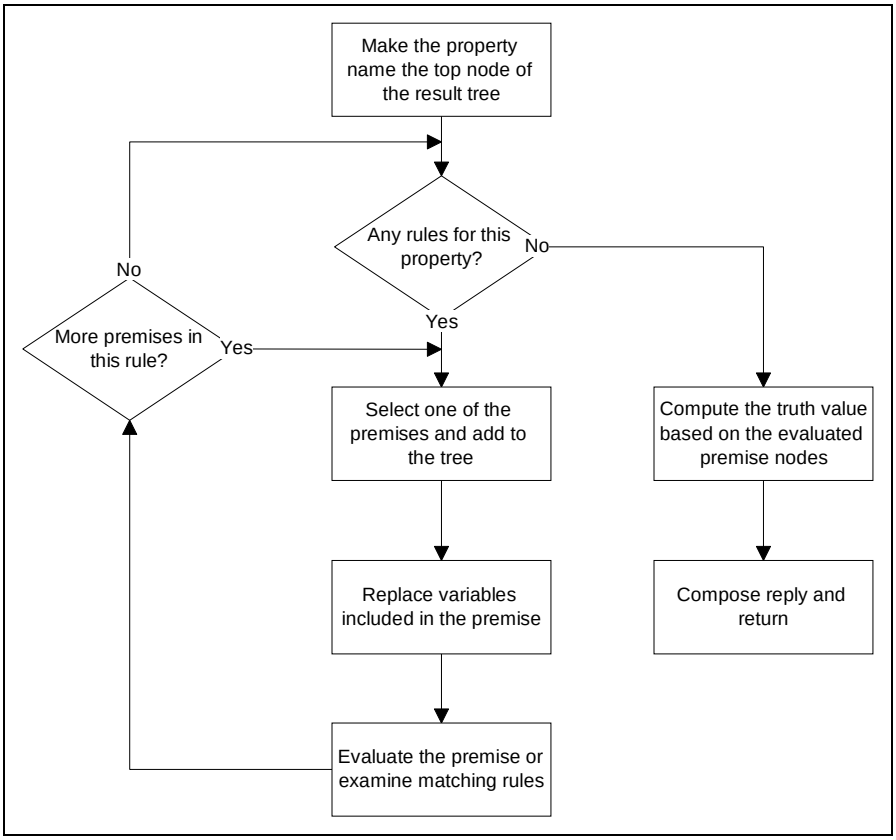


Fig. 13. Fluid property reasoning mechanism of AutoHAZID.

6.3.3 Fluid compatibility

Chemical compatibility data

Compatibility problems may lead to hazardous consequences, either directly as in the case of explosive mixtures, or indirectly as in the case of fluid composition changes which lead to changed fluid properties. Problems caused by the appearance of extra components in the fluid are handled by the chemical compatibility check feature of AutoHAZID. In those situations, unintended reactions or problems with equipment materials may occur.

The compatibility check feature is expected to handle issues related to the effects of mixing two fluids. Incompatibility of fluids with equipment materials should also be covered to some extent.

The mixing of two fluids may result in unwanted reactions and hazardous consequences. The compatibility check feature provides AutoHAZID with the capability to identify such situations. It finds out whether a reaction can take place between the fluids in question and, if so, what are the possible effects.

In case of fluid-material incompatibility, there is a need to identify hazards related to corrosion and embrittlement in addition to the possible reactions. However, those issues are difficult to handle with the compatibility check methodology used.

In the version of AutoHAZID released in connection to the final prototype of STOPHAZ, compatibility problems are investigated with the support of a compatibility matrix which uses the reactivity group information from the fluid property data base as the starting point. The compatibility matrix is able to identify compatibility problems between two fluids or a fluid and a material as presented in Table 15.

Table 15. Incompatibility problems covered by the compatibility matrix.

Heat generation
Fire
Innocuous and non-flammable gas generation
Toxic gas generation
Flammable gas generation
Explosion
Violent polymerisation
Solubilization of toxic substances
Potentially hazardous but unknown

In the AutoHAZID fluid property library, one or more reactivity groups are defined for each chemical. In accordance with Table 15, the compatibility matrix contains information of what happens if fluids belonging to different reactivity groups are mixed together.

Some of the desired functionality was left outside the scope of the basic version of AutoHAZID. The missing features include the description of the possible unwanted reactions, and chemical-material issues such as erosion and corrosion.

The reactivity group definitions for the chemical substances listed in the static fluid property library of AutoHAZID (see Table 9) are presented in Table 16. The group numbers without brackets are based on a publication of the US Environmental Protection Agency (Hatayama 1980) and the numbers with brackets are additional groups based on other sources. The additions are needed because Hatayama concentrates on reactivity properties based on the molecular structures of the substances and reports groups based on chemical reactivities only in few cases. The complete compatibility matrix is presented in Appendix 4. It should be noted that the matrix is a modified version of the one published by Hatayama (1980) and inherits a limitation which concerns the reactivity group 107 (Water Reactive Substances) in particular. When this type of a matrix is used to support considerations related to the handling of waste chemicals, it is sufficient to know that chemicals belonging to group 107 should never get mixed with other chemicals. However, when a chemical process is considered in which chemicals belonging to group 107 are processed, more detailed information about the compatibility problem would be required.

Table 16. The chemical list with reactivity group definitions.

Component ID	**Formula**	**Reactivity Groups**
ammonia	NH_3	10, (107)
ammonium sulphate	$(NH_4)2SO_4$	(104)
benzene	C_6H_6	16, (101)
butane	C_4H_{10}	29, (101)
butene	C_4H_8	28, (101, 103)
carbon dioxide	CO_2	
carbon monoxide	CO	(101)
chlorine	Cl_2	104, (107)
ethane	C_2H_6	29, (101)
ethene, ethylene	C_2H_4	28, (101, 103)
ethylene dichloride	$C_2H_4Cl_2$	17, (101)
Freon-11, trichlorofluoromethane	CCl_3F	17
Freon-12, dichlorodifluoromethane	CF_2Cl_2	17
hexane	C_6H_{14}	29, (101)
hydrogen	H_2	(101)
hydrogen chloride	HCl	1, (107)
hydrogen cyanide	HCN	1, 11, (101, 103)
hydrogen sulphide	H_2S	33, 105, (101)
methane	CH_4	29, (101)
nitrogen	N_2	
oxygen	O_2	(104)
propane	C_3H_8	29, (101)
propene	C_3H_6	28, (101, 103)
sodium hydroxide	NaOH	10
sulphur dioxide	SO_2	(105, 107)
sulphuric acid	H_2SO_4	2, 107
toluene	$C_6H_5CH_3$	16, (101)
water	H_2O	(106)

Compatibility checker

In order to fully utilise the potential of the compatibility matrix, there is a need to integrate the compatibility considerations with the fluid property queries. For instance, the query ”decomp” (see Table 8) which is used for determining the possibility of fluid decomposition under the actual process conditions could fail to identify a hazard if the occurrence of heat generation due to a change in fluid composition was not known. Moreover, queries such as ”gasGeneration” can only be resolved by using compatibility information.

In the version of AutoHAZID released in connection to the final prototype of STOPHAZ, the compatibility matrix is only taken into rather limited use. Integration to fluid property queries could not be included in the scope of the STOPHAZ project due to timetable and resource constraints. At present, the compatibility information is processed by a separate compatibility checker module. A number of ideas for more powerful use of the compatibility check feature are discussed later.

The compatibility checker accepts two fluid descriptions as input. Looking at each component of both fluids, it checks all the possible combinations. For each pair of components, reactivity properties are retrieved from the fluid library. The possible compatibility problems for each pair of reactivity groups are found in the compatibility data base. When the compatibility checker has gone through all the component pairings and the interactions of the associated reactivity groups of both fluids, the resulting compatibility problem data is returned.

It is up to the calling module when to initialise compatibility checks. The result of the compatibility check is a structure composed of a truth value and a description which contains an explanation in terms of the phenomenon sentences listed in Table 15. This result can then be combined with the other results in the desired way.

6.3.4 Linkage to AutoHAZID

Query mechanism

As already discussed in the previous sections, AutoHAZID needs to be able to consider fault propagation correctly when the propagation depends on the processed fluids. The fluid model provides AutoHAZID with that capability. In order to make fluid property queries to the fluid model, the AutoHAZID fault propagation engine needs a query mechanism to be used in connection with the fault propagation models.

Usually, the fluid model needs to be called during the causal reasoning process because the validity of the reasoning may depend on which process deviation and fault we are looking at. It might be appropriate in some cases to postpone the considerations to the results filtering stage. However, the fluid composition always has to be known, and it is most convenient to access that information when the faults are being considered.

AutoHAZID calls the fluid model system with the query keyword and the location object. Next, the fluid model inference engine is triggered with the query keyword as the top level goal. It retrieves the applicable fluid rules from the fluid library and parses them to carry out the necessary variable substitutions. The properties package is called when necessary to get the fluid property information needed to resolve the rules. Table 17 illustrates how the fluid property reasoning task is defined when a query keyword occurs in an arc during fault propagation. The objective in the example is to determine the toxicity of hexane in order to assess the relevance of an environmental consequence.

Table 17. Definition of the reasoning task in an example case based on the occurrence of a query keyword in an arc.

Arc:	*arc([fault,'line fracture'],1,[consequence,['serious environmental contamination',toxic]])*
Query keyword:	*toxic*
Location:	*L1* (the fault propagation routine knows the location under study)
Called function:	*checkFluidProperty* (see Appendix 5, page 1)
Fluid:	*hexane* (the value of the attribute L1.CompName)
Reasoning task:	resolve the top goal *toxic* using the fluid rules for *hexane*

If a properties package is not available or it can not provide the required fluid property information, the fluid model system uses the data stored in the fluid library instead. In that case, each component of the fluid is considered separately.

The final result of the backward chaining process is a description of the identified realisations of the keyword in the form of a logical tree. Next, the inference engine evaluates the tree based on the parsed conditions extracted from the fluid rules.

In practice, the query mechanism is based on defining the required fluid model queries inside the fault propagation models of AutoHAZID. The fault propagation inference mechanism of AutoHAZID is based on the use of the signed directed graph (SDG) formalism. The concept of an "arc" is used for representing a cause-consequence relationship as shown in Table 18. In these cause-consequences relationships, a deviation can appear as a cause or a consequence.

Table 18. Examples of the representation of cause-consequence relationships in AutoHAZID fault propagation models.

arc([in,temp],1,[out,temp])
High (low) input temperature causes high (low) output temperature (deviation → deviation)
arc([fault,'power supply fails'],1,[out,noFlow])
Failure of power supply causes no flow at the output (cause → deviation)
arc([fault,'line fracture'],1,[consequence,['serious environmental contamination',toxic]])
Line fracture causes serious environmental contamination if the fluid is toxic (cause → consequence)

The fluid model is linked to the AutoHAZID graph formalism in the way illustrated by the last example in Table 18 above. Keywords are defined to represent the various features of the fluid model, and these keywords are used as condition predicates within the arc definitions in the models. The fluid model can be called to find out if a specific cause-consequence relationship holds for the processing of a certain fluid.

In addition to the unit fault propagation models, the unit model library of AutoHAZID contains a system of predicates with flags which determine whether the fluid model can be used for resolving a query or not. This allows the development of the predicate mechanism and the fluid model rules independently of one another.

The keywords which appear in the unit models are handled as predicates. The list of predicates is defined in the beginning of the unit model library file. Each of the predicates is defined to belong to one of three categories:

- ”C” : To be evaluated using a c-language procedure,
- ”V” : To be evaluated by the rule based fluid property expert, or
- ”D” : Defined in terms of other predicates.

Testing of a predicate which belongs to the category ”V” causes a fluid model query to be initiated. The query carries the property name and a pointer to a location object.

After each predicate, a comment has been added which defines the methods available for the evaluation of that predicate. Fluid model queries should be used whenever possible because actual fluid properties from the properties package are used in connection to them. An example predicate definition stating that the flammability of the fluid should be checked using the fluid model system is given below:

```
predicate(flammable(Port),"V").    % C + V
```

Location objects are normally used in AutoHAZID to specify the intended fluid composition and the normal process conditions. However, a new location object with query specific information is constructed for a fluid model query. If the fluid composition has changed, the intended fluid composition can be replaced with an abnormal one. If the process conditions differ from normal, the pressure and temperature characterising the abnormal conditions can be given.

Table 19 lists the classes defined in AutoHAZID's c++ code which are involved in the fluid considerations carried out by the fluid model system.

Table 19. AutoHAZID program code classes involved in fluid considerations.

Fluid
- Runtime repository of the fluid property data found in FluidLib
- Fluid objects are collected in a global list called FluidList

FRule
- Runtime repository of the fluid rules found in FluidLIb
- FRule objects are collected in the ruleObjects field of Fluid objects

CondReply
- The inference engine returns the result in the form of a tree which is then transformed into the form of a string based conditional reply and returned as a CondReply object

StrTree
- The results of the backward chaining inference are represented as a tree built from StrTree objects
- A field is included for the truth value of the statement.

Reply conventions

When the fluid model system has resolved the fluid property query, it composes a reply in accordance to specified conventions. The reply is composed of a truth value, a list of unresolved conditions and a description.

The truth value is only FALSE when at least one of the necessary conditions for the queried fluid property is known to be false. Otherwise, TRUE is returned together with the conditions under which the property may hold. Conditions may remain unresolved due to the lack of fluid property information. They are returned as part of the reply for further use by AutoHAZID. If there are multiple unresolved conditions, their logical relations (AND/OR) are also included in the returned expression.

The description may contain textual information to justify or explain the given reply. For instance, the fluid property values used for the evaluation of conditions could be included in it. In the case of compatibility checking, additional information about the nature of the incompatibility problem could be provided as description. The description feature has not yet been taken into use in the version of AutoHAZID released in connection to the final prototype of STOPHAZ.

6.3.5 Summary of implemented features

Due to timetable and resource constraints, the STOPHAZ project had to focus on the implementation of features which fulfilled the most fundamental requirements set for the AutoHAZID fluid model system. Therefore, a number of advanced ideas for improving the capabilities of the system were left unimplemented. To avoid confusion, a brief summary of the features which were implemented is presented in the following. Table 20 summarises the implemented features of the AutoHAZID fluid model system. In section 6.3.6, limitations and ideas for further development will be discussed.

Table 20. Summary of implemented features of the AutoHAZID fluid model system.

Fluid model system feature	**Implementation**
Fluid property queries	• Conditional cause-consequence relationships • Evaluation of the conditions based on fluid properties
Quantification of fluid properties	• Use of external calculation package to determine fluid properties for specified fluid composition in specified process conditions • Comparison of property values to process conditions or predefined threshold values
Fluid compatibility study	• List of process chemicals and likely contaminants • Reactivity classification of chemicals • Use of a rectivity group based reaction matrix

The main objective was to implement an intelligent system to resolve fluid property queries. Whenever the relevance of a cause-consequence relationship is dependent on the properties of the processed fluid, AutoHAZID tries to solve the problem by investigating the fluid properties. This can be done by calling the rule-based fluid model system which is able to reason about those issues.

Generic rules have been defined for the knowledge-based processing of fluid property data. These rules cover all the alternative situations and return the final conclusions in the form of a truth value accompanied with explanatory information. This information includes a list of unresolved conditions and background information about the property values and other data used in the reasoning.

The rules are stored in the internal library of chemical properties together with static chemical property data. Data on 28 compounds is stored in this library. Some of the static property values are only used if a properties package can not be used but also some properties have been included which are not covered by those packages. Such properties include toxicity and reactivity.

Properties packages are able to calculate the values of fluid properties at the actual process pressure and temperature conditions. They can also estimate the properties of mixtures. A link has been implemented from AutoHAZID to two properties packages supplied by simulation software companies. Whenever a properties package has been made available, it will be used automatically to provide better estimates of a number of fluid properties.

In order to be able to consider fluid compatibility issues, a reaction matrix based on a reactivity group approach was added to complement the fluid-oriented part of AutoHAZID. It can be used to identify possible unintended reactions between two fluids. At present, this compatibility check feature has to be called separately from the other fluid related considerations covered by the fluid property queries.

6.3.6 Limitations and ideas for further development

Limitations of the current implementation of AutoHAZID

A number of aspects of fluid property reasoning remain poorly covered after the developments arising from the STOPHAZ project. However, the development and evaluation efforts carried out in the project helped to identify which areas of the methodology would require further development and what kind of added features could be expected to lead to significant improvement of performance.

In the version of AutoHAZID released in connection of the final prototype of STOPHAZ, the features of the software cover poorly the following aspects of fluid property reasoning:

- Properties of mixtures
- Dependency on process conditions
- Downstream impacts of unintended reactions
- Order of magnitude considerations
- Filtering philosophy
- Data acquisition from the user
- Properties of materials and reactions.

In the following, these aspects of fluid property reasoning are discussed in detail. Ideas are also proposed for the the further development of the AutoHAZID fluid model system. The objective is to provide those interested in the improvement of AutoHAZID's reasoning capability with guidance based on the lessons learnt durring the STOPHAZ project.

Properties of mixtures

AutoHAZID has well-organised access to fluid property information but its capabilities to use this information are still rather primitive. The intended fluid composition at various locations of the studied process is given to AutoHAZID as part of the plant description effort. Mechanisms for defining and studying abnormal fluid compositions are largely missing. By definition, intended fluid compositions should not initiate event chains which lead to hazards. Therefore, only very few fluid composition dependent hazards are identified.

A prerequisite for the more sophisticated use of the fluid model is that the possibilities for arriving at an abnormal composition of the fluid are known at the studied location in the process. One solution is to define a breakpoint corresponding to every possible location of the process where the composition of the fluid can change. In addition to intended changes to the fluid composition (mixer, reactor, etc.), the breakpoint procedure could cover unintended changes. Such changes can take place due to wrong or extra state of matter, ingress of extra components, polymerisation, decomposition, or unintended reactions leading to unintended reaction products.

Proper handling of fluid composition deviations would require changes in the hazard identification procedure. If the deviation considered in the current fault propagation implies a change in the fluid composition or in the state of matter, the fluid description should be changed accordingly. If a specific fault is considered and the consequences of that fault need to be identified next, the fluid description (composition and state of matter) should be revised according to the fault. For example, the initial fault may be a water leak into the system. In that case, the conditions attached to the potential consequences at the other end of the related event sequences need to be reconsidered taking into account the changed

composition and properties of the fluid, and the possible changes to process conditions.

The causes of unintended changes could often be handled by recognising the associated special circumstances and triggering the appropriate considerations. For instance, temperature increase may in some cases cause the decomposition of the fluid. This is an event chain which AutoHAZID should be able to identify. Whenever decomposition is found possible, the associated change to the fluid composition should be recognised and a study of the impacts of this change should be carried out.

The unintended ingress of extra components to the fluid is difficult to model. In principle, anything can be inserted in the process. For instance, a human operator may mistakenly empty into the process a rail tank containing a completely wrong chemical. However, it is not possible to design a process which is tolerant to any chemical. Therefore, an educated guess might be the best way to carry out a sufficient study of these issues. For instance, the plant description could include optional information on ”extra fluid components that could be present”. This would serve as an educated guess of what extra fluids could mistakenly enter the process. Another option would be to consider the possible breakdown of internal barriers, such as closed valves, heat exchanger interfaces and pump seals, which are expected to separate from each other the fluids at the opposite sides of the barrier.

Cause-consequence chains are often dependent on the composition and properties of the associated fluid. The fluid composition and properties can in turn depend on the initial fault which has triggered the sequence of events. For example, the initial fault may be a water leak into the system. In that case, the conditions attached to the potential consequences at the other end of related event sequences need to be reconsidered taking into account the changed composition and properties of the fluid, and the possible changes to process conditions. To handle this kind of situation properly, the fluid model system needs to be extended with a fault interpreter to modify the current fluid composition and process conditions according to the impacts of the fault under consideration.

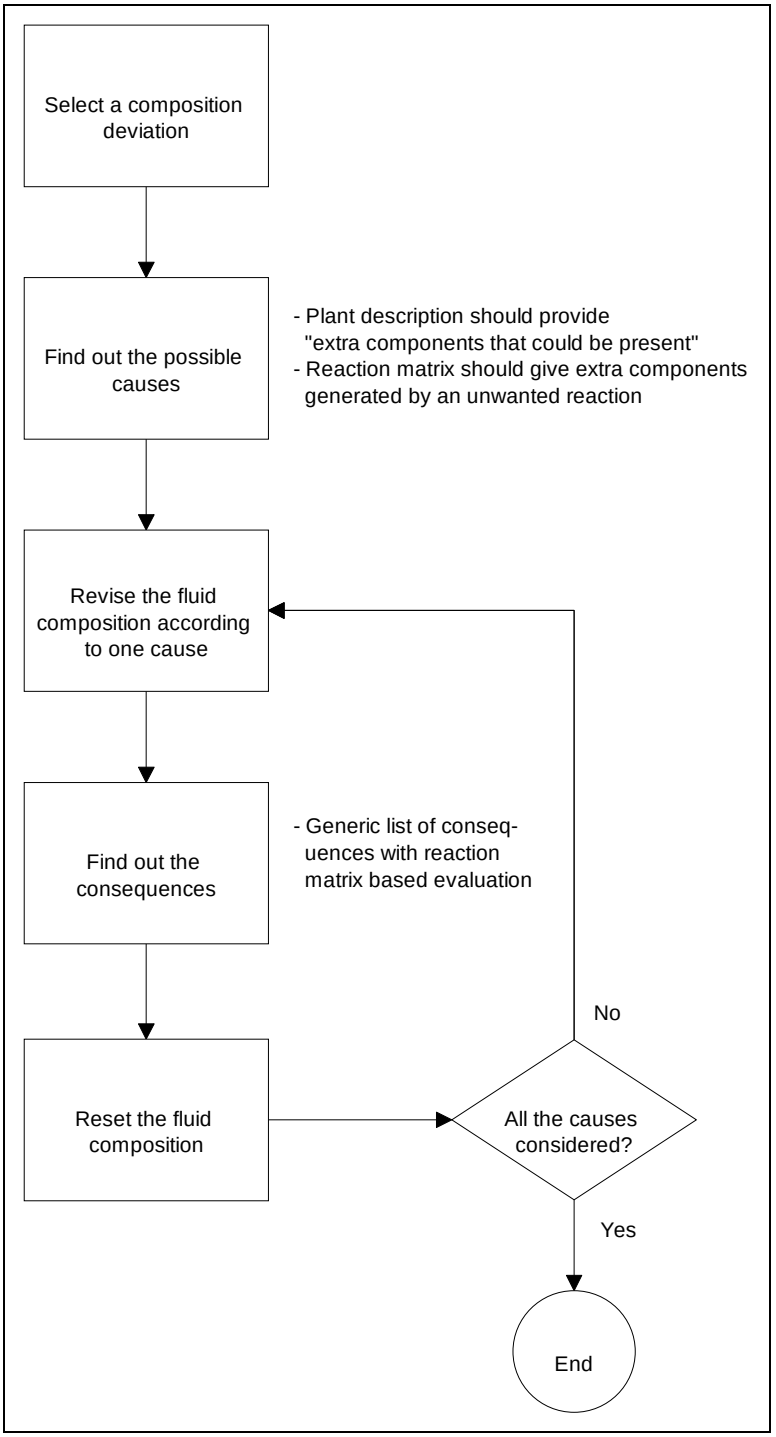


Fig. 14. Enhanced procedure for the analysis of composition deviations.

Fig. 14 illustrates the procedure for the analysis of composition deviations, covering both the presence of unintended fluid components and abnormal concentrations of intended fluid components. If phase deviations are considered as one type of fluid composition change, the same procedure can be applied to the consideration of unintended phase changes. Fluid displacement problems could be considered in terms of phase-type composition deviations in the AutoHAZID equipment models.

Dependency on process conditions

Consequence events, such as fire and overpressure rupture, depend on the actual process conditions. In the case of fire, the flammability of the fluid is the main issue but temperature increase may in the case of many fluids increase the fire hazard considerably. Overpressure rupture, on the other hand, is directly dependent on the magnitude of unintended pressure increase.

Intended process conditions are given to AutoHAZID as part of the plant description. However, hazards are nearly always associated with abnormal process conditions. How to estimate the actual process conditions in an abnormal situation is a topic for a separate discussion in the next part of this section. Process conditions are, however, also linked to the composition of the fluid.

Changes in fluid composition may lead to changes in process conditions. For instance, decomposition may lead to an increase of temperature and pressure. What makes this issue particularly complicated is that changes in process conditions can also lead to changes in fluid composition. The same decomposition reaction which led to a sudden increase of temperature and pressure may initially have been trigged by a smaller unintended temperature increase.

The equipment models of AutoHAZID could be extended to deal with these considerations. The models should include all possible links between composition deviations and other deviations, bearing in mind that the fluid model should take care of screening out all those links which do not hold under the studied circumstances.

The consequences of composition deviations in the unit models could be extended to correspond to a generic list of all the possible consequences of mixing incompatible fluids, and the fluid model should take care of screening out the irrelevant consequences using the reaction matrix.

One way to handle the changes in process conditions due to fluid composition changes is to evaluate the consequences using changed values of process parameters. This is in accordance with the enhanced procedure for analysing composition deviations (see Fig. 14). Another possibility is to build the dependencies in the equipment models. This would mean the addition of arcs which represent the relation between composition deviations and other deviations in the models. An enhanced procedure is presented in Fig. 15 for the handling of links between composition deviations and other deviations.

Downstream impacts of unintended reactions

In addition to the features related to resolving fluid property queries, AutoHAZID has the capability to identify fluid compatibility problems. However, fluid compatibility issues are at present considered in isolation from the other hazard identification reasoning. Unless the compatibility check feature is explicitly called and its results are correctly interpreted, potential hazards such as overpressure rupture due to an unintended reaction which leads to gas generation and increase of pressure are not identified. In order to fully utilise the potential of the compatibility matrix, the fluid model queries should be redesigned to include compatibility issues.

Compatibility problems may lead to hazardous consequences either directly as in the case of explosive mixtures, or indirectly as in the case of fluid composition changes which lead to changed fluid properties. Whenever new fluid components are added, the compatibility of the new components with the others should be checked using the compatibility matrix.

The compatibility check may reveal the presence of immediate hazards, such as fire, explosion, violent polymerisation and toxic release. The other findings of the compatibility check may reveal potential indirect downstream impacts.

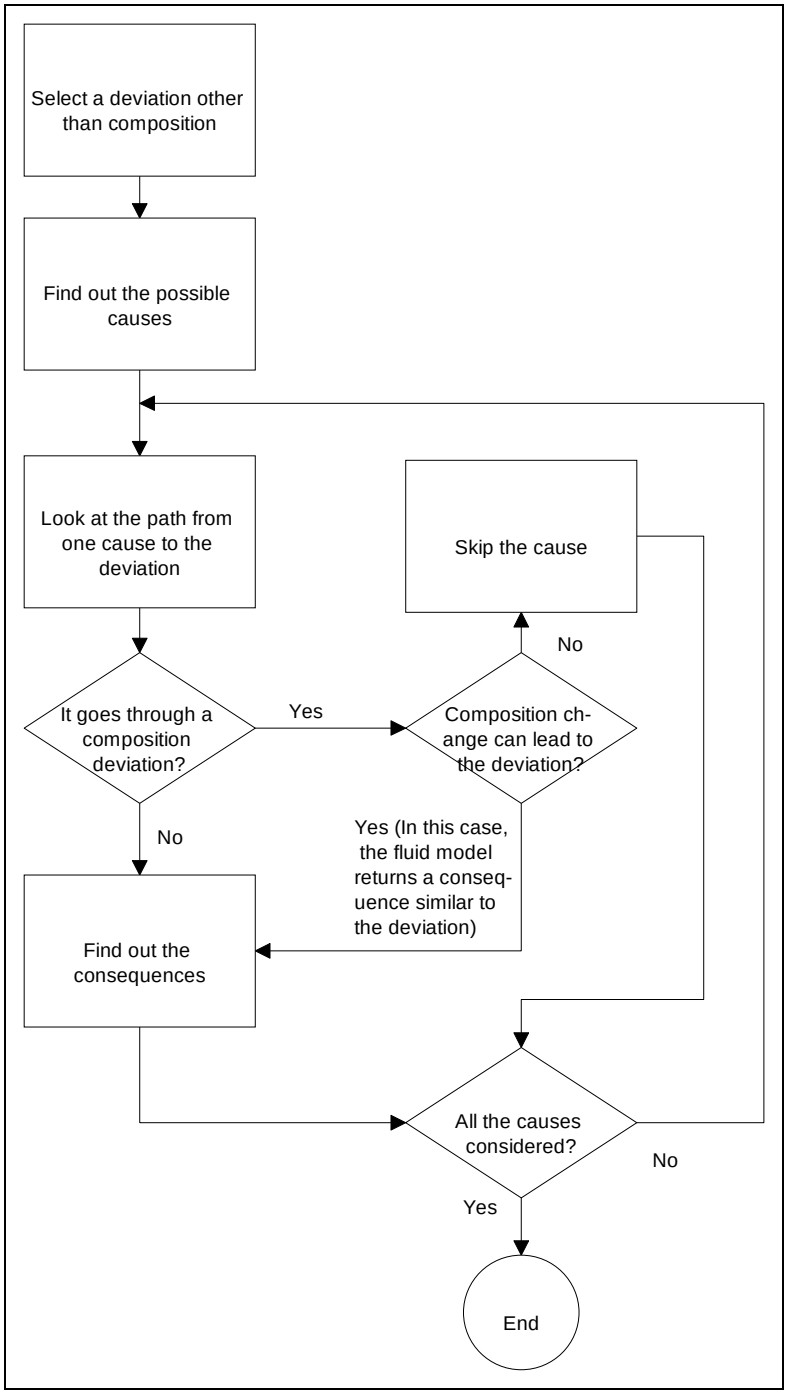


Fig. 15. Enhanced procedure for handling the links between composition deviations and other deviations.

For instance, a hazardous event chain might be initiated due to an unexpected presence of vapour, and the presence of vapour could be the result of an unwanted interaction of chemicals. To deal with this kind of situation, the fluid property query ”vapour” could be used as a condition to a cause-consequence relationship between a fault related to an unintended presence of a chemical and a composition deviation dealing with the unintended presence of vapour. The fluid model system would then examine the potential interactions between the extra chemical and the components of the intended fluids and find out whether vapour might appear as a result of these interactions. Gas generation is one of the phenomena considered in the compatibility matrix.

When an unwanted reaction takes place, unintended compounds often appear in the fluid. In addition to a reaction between two fluid components, the unwanted reaction can also be a decomposition reaction. In both cases, the change in the fluid composition should be taken into account in related hazard considerations.

The question of how to trigger AutoHAZID to consider the properties of reaction products in case of unwanted reactions is an important issue in relation to improving the performance of the fluid model system. If the change in fluid composition can be handled as a cause of a composition deviation, the enhanced procedure for the analysis of composition deviations can be used. Fig. 14 illustrates this procedure.

A more thorough study of how compatibility issues are covered in AutoHAZID reveals that the unit models do not cover fluid composition deviations properly. All possible links between composition deviations and other deviations should be included, bearing in mind that the fluid model should take care of screening out all those links which do not hold under the studied circumstances. In addition, the consequences of composition deviations in the unit models should be extended to correspond to a generic list of all the possible consequences of mixing incompatible fluids, and the fluid model should take care of screening out the irrelevant consequences using the reaction matrix.

Order of magnitude considerations

The main drawback of qualitative modelling is that quantitative information can not be incorporated in the models. The reasoning is based on studying the phenomena in qualitative terms. For instance, it may be stated that the fault ”pump overspeed” leads to the deviation ”high temperature”. Also, it may be stated that the fault ”external fire” leads to the same deviation ”high temperature”. It is impossible to judge from this kind of basic qualitative models how big the temperature increase is in the different fault situations.

This limitation of qualitative models leads to problems in the evaluation of candidate cause-consequence chains. The models suggest a causal relationship whenever the fault has an influence on the process parameter which matches with the process parameter deviation associated with the consequence. Only some of the suggested causal relationships are relevant because the impact of the fault is not always big enough to lead to the consequence. For instance, pump overspeed can rarely cause such an increase of temperature that the decomposition of the fluid would be triggered.

At present, AutoHAZID can not reason about order of magnitude issues. What would need to be done to improve AutoHAZID in this respect is to attach quantitative order of magnitude information to the faults in the models. The magnitude of the impact to a process parameter could be defined in terms of a value range, either referring to the absolute value as the result of the fault or to the amount of increase or decrease. In some cases it might be necessary to define the impact in terms of a multiplier value with a range. The use of such value ranges would give AutoHAZID the possibility to reason about the expected magnitude of the impact of a deviation and thereby to detect cases where the cause-consequence chain is clearly unrealistic.

Temperature change due to wrong ratio of fluid components in a mixture is a situation which could be handled with this technique rather well. If the fluid components enter a vessel through different inlets, the temperature of the mixture changes along with the change of composition. In most cases, this kind of a temperature change is likely to be relatively small. Furthermore, the absolute temperature limits can be estimated from the temperatures of the inlet streams.

It is clear that the proposed value range technique is not a complete solution to the order of magnitude problem. Firstly, some faults are not easy to quantify. Particularly difficult are leaks. The size of a leak can only be estimated properly if the initial fault which caused the leak is properly specified. For instance, it is difficult to quantify the leak size if general expressions like ‘external damage’ are used to describe the fault.

Secondly, fault propagation may complicate the order of magnitude reasoning. For instance, it is evident that high flow in the hot side of a heat exchanger leads to high temperature of the cold flow. It may be possible to estimate the oredr of magnitude of this high temperature deviation by investigating the temperature constraints and possible deviations of the hot flow. However, the magnitude of the temperature change can only be estimated properly by carrying out quantitative calculations which take into account the flow volumes and the efficiency of heat exchange.

Filtering philosophy

Filtering is a powerful mechanism for improving the relevance of the HAZOP results before they are presented to the user. For instance, filtering based on consequence severity can be carried out. This filtering option has been implemented in the version of AutoHAZID released in connection to the final prototype of STOPHAZ.

Consequence severity is very good as the basis of filtering. When consequences are classified from 1 for least severe to 5 for most severe, the user of AutoHAZID can study the identified problems level by level. When event chains with the most severe consequences have been examined and dealt with, the user can move on to the slightly less severe problems, and so on. Of course, the usefulness of this kind of working procedure is completely dependent on the validity of the consequence severity classification.

So far, only a simple classification of the existing selection of consequences to five categories has been done in AutoHAZID. The classification could be improved in two ways. Firstly, instead of a predefined classification, each actual consequence of an identified event chain could be classified in a context

sensitive manner based on all what is known about the event chain. Fluid property information on toxicity and flammability would have a significant role in these considerations. Furthermore, the availability of order of magnitude information would greatly improve the possibilities for the correct classification of the consequences.

Secondly, the list of consequences used in the models could be reconsidered. For instance, it would make consequence classification and the associated filtering easier if release incidents of different types were called with unique names. The consequence severity of a toxic release due to pipe rupture has to be estimated differently from the consequence severity of a toxic release from the relief valve of a vessel. This is because the quantity, duration and state of matter of the release are different and depend on different features of the process in the two cases.

The above discussion has suggested that more reasoning should be carried out in connection to the hazard identification procedure. However, a key motivation for the development of filtering operations is to move some of the reasoning effort out of the main procedure which consumes a lot of time and computer memory. Therefore, it may be necessary to keep the consequence severity based filtering and other possible filtering options simple.

Simple filtering options related to fluid properties could be added quite easily. AutoHAZID could be allowed to generate all toxicity and flammability related consequences during the main hazard identification reasoning. Then, a filtering option implemented for that purpose could be used to check these fluid properties and remove irrelevant consequences and the associated event chains from the results when needed. The main difficulty is that the properties of the fluid may have changed as a result of the fault and the subsequent events, and this can not be known by only looking at the data in its processed form as included in the final HAZOP results. To make this type of simple fluid property based filtering possible, AutoHAZID should be modified to store information on fluid property changes in connection to the identified event chains.

Data acquisition from the user

AutoHAZID is designed for fully automated HAZOP study. The only stages of user interaction are preparatory actions before starting the analysis and post-analysis operations for examining the HAZOP results. The preparatory actions include describing plant equipment, connectivity and fluids, and adjusting analysis option settings.

The nature of AutoHAZID would change if user interaction were to be introduced for the acquisition of complementary data. However, this could be justified from the point of view of a typical AutoHAZID user. If AutoHAZID is used as part of the standard design practice for the identification of safety problems during design, it is quite likely that it will be used for repeated small safety checks and the user will be present all the time ready to give complementary data.

User interaction would be particularly useful for improving the capability of AutoHAZID to make correct decisions on the validity and relevance of candidate cause-consequence chains. The correctness may depend on the properties of the processed fluids, associated unintended reactions, or order of magnitude issues.

In the case of a computer program like AutoHAZID which is originally designed to manage without requesting complementary data from the user, the introduction of user interaction can be targeted to such areas where it is most beneficial. Plain data, such as fluid property data, is better retrieved automatically from internal or external data bases. User interaction should concentrate on the acquisition of data which results from the human judgement of the expert user. Order of magnitude issues are a good example of such data.

For instance, estimating the size of leak-type faults is very difficult to perform without help from the user. In order to give a likely estimate, various scenarios may have to be considered and compared, for example in the case of various possible sources of external damage. This is almost impossible for the computer but a human expert is normally able to give a reasonable estimate.

Another type of information which could be acquired from the user concerns the properties of possible unintended reactions. AutoHAZID may be able to identify that an unintended reaction may occur but it would be very difficult to make it capable of understanding the reaction and of determining what are the resulting unintended reaction products. A human expert could at least give a quick assumption of what might happen and help AutoHAZID to go ahead with the reasoning based on that assumption.

If user interaction is added to AutoHAZID in the future, the data acquisition mechanism needs to be designed carefully. An explanation should be given to the user about the purpose each requested piece of information, including the situation to which the information is to be applied. If the user has given a piece of information, the same information should never be requested again. However, if the situation under consideration is different, the user must be given a chance to change the data to match the situation. Finally, the whole user interaction feature should be made optional, and AutoHAZID should remain designed to manage without it.

Properties of equipment materials and intended chemical reactions

The ideas for further research and development in relation to AutoHAZID presented so far have been related to improving the handling of fluid dependent hazard scenarios. Methods for dealing with the properties of equipment materials and intended chemical reactions would be natural extensions to the capabilities of AutoHAZID.

Hazards related to the materials of construction are currently not taken into account in AutoHAZID. For instance, the suspectibility of a material of construction to corrosion is not considered. Furthermore, the compatibility matrix can only be applied to materials of process equipment in a very rough manner, and faults related to fluid-material interactions are not covered by the models at all.

In principle, the properties of materials could be handled by the fluid model system in the same way as fluid properties. A set of material property queries to be resolved could be defined. Values for the interesting properties of materials

could be stored in an internal data base of material properties or retrieved from an external source.

Hazards of intended reactions and the impacts of reaction problems on process conditions and fluid composition are currently also neglected by AutoHAZID. The typical behaviour of different types of reactions in abnormal situations could be described in a way similar to the HAZOPTOOL Chemical Reaction Knowledge Base (RKB). This aspect of HAZOPTOOL is discussed in section 5.3.3 of this thesis.

6.4 Implementation of the fluid model system

6.4.1 Overview

The fluid model system facilitates fluid property queries and compatibility checks. Its module architecture has been illustrated in Fig. 12. Compatibility check services are provided by the Compatibility Checker, and the other modules, i.e. Query Handler, Inference Engine and Rule Parser, are involved in the fluid property query operations.

In the following, the implementation of the fluid property query operations will be described. The implementation of the compatibility check operations concentrates on database look-up issues and, therefore, is left out from this description as less interesting.

The functions involved in fluid property query operations and the call relationships between them are illustrated in Fig, 16. The top level function is called checkFluidProperty. The functions backwardChain, substituteVariables and evaluateNodes include recursive calls and therefore appear twice in the diagram, both as the calling function and the called function.

The rest of this section consists of the descriptions of the individual functions. Appendix 5 contains the corresponding c++ -code listings.

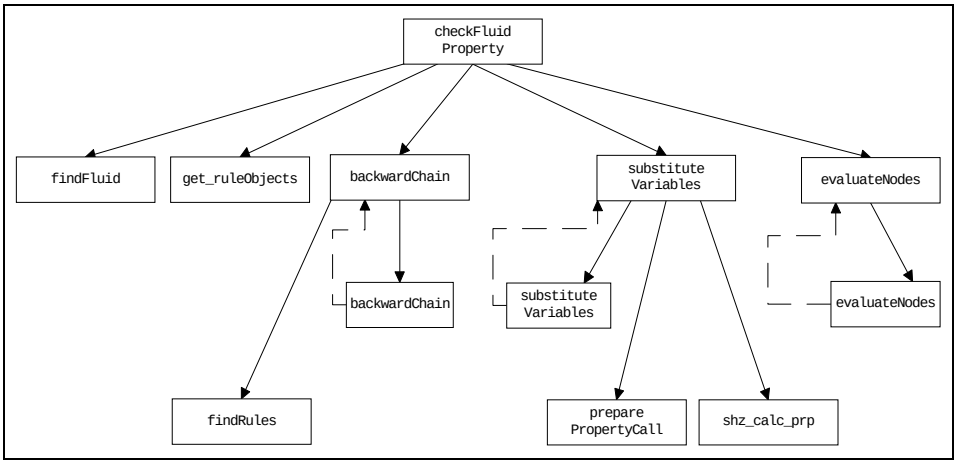


Fig. 16. Functions involved in the fluid property query operations.

6.4.2 Function checkFluidProperty

The purpose of the checkFluidProperty function is to manage the recursive tree handling operations which examine the relevant fluid rules and the associated property values in order to evaluate the requested fluid property. In order to supply the tree handling functions with appropriate problem-specific input data, it first has to locate the required fluid data and rules in the internal data repository. When the tree handling operations have been completed, the checkFluidProperty function has to compose the final reply.

The inputs and outputs of the function are listed in Table 21.

Table 21. Inputs and outputs of the checkFluidProperty function.

	Name	**Object type**	**Explanation**
INPUTS:	P1	Predicate	Query keyword which defines the property to be evaluated
	L1	Location	Process conditions at a given position, including fluid compositions (components and concentrations)
OUTPUT:	crep	condReply	Reply (truth value and possible conditions)

An overview of the function program code is presented in Table 22.

Table 22. Overview of the program code of the checkFluidProperty function.

```
For each fluid component fp1 at location L1
{
        call findFluid to retrieve fluid data stored in FluidLib;
        call get_ruleObject to pre-process the fluid property rules applicable to fp1;
        call backwardChain to find out from the rules how to resolve the query;
        call substituteVariables to apply the available fluid property data;
        call evaluateNodes to compute the truth value;
        add the answer to the condReply object;
}
return the cumulated answer in the form of the condReply object.
```

6.4.3 Function findFluid

The purpose of the findFluid function is to provide access to the data stored in the file FluidLib. This file has been pre-processed in connection to the initialisation of AutoHAZID. The data for all fluids has been loaded in the memory in the form of Fluid objects. The findFluid function tries to match the names of the fluid components at the studied location to the chemical names associated with the Fluid objects. When there is a match, a pointer to the matching object in the global list of Fluids, fluidList, is returned.

The inputs and outputs of the function are listed in Table 23.

Table 23. Inputs and outputs of the findFluid function.

	Name	**Object type**	**Explanation**
INPUT:	N1	String	Name of the fluid component under consideration
OUTPUT:	flu	Fluid	Pointer to the corresponding Fluid object

An overview of the function program code is presented in Table 24.

Table 24. Overview of the program code of the findFluid function.

```
For each fluid object in the global list of fluids
{
        if the value of its name attribute equals to N1
        {
            return a pointer to the fluid object;
        }
}
```

6.4.4 Function get_ruleObjects

The purpose of the get_ruleObjects function is to provide access to the rules which are applicable to the studied fluid. The object structure of the fluid data implies that each Fluid object knows the list of applicable rules. It is stored in the memory as a list of Frule objects. Therefore, the function get_ruleObjects only has to read the pointer to this list from the relevant Fluid object.

The inputs and outputs of the function are listed in Table 25.

Table 25. Inputs and outputs of the get_fluidObjects function.

	Name	**Object type**	**Explanation**
INPUT:	fp1	Fluid	The fluid under consideration
OUTPUT:	objList	Pointer to a list of Frule objects	The fluid property rules applicable to the given fluid.

An overview of the function program code is presented in Table 26.

Table 26. Overview of the program code of the get_fluidObjects function.

Return the value of the ruleObjects attribute of the Fluid object fp1. (Preprocessing has taken place during the initialisation of AutoHAZID and, thus, Frule objects containing the fluid rules exist and they are organised in a ready-to-use list)

6.4.5 Function backwardChain

The purpose of the backwardChain function is to process the fluid property rules and thereby to construct a logical tree for evaluating the requested property. The implementation is based on the backward chaining technique. The query keyword is used as the top level goal, and the premises of matching rules are used as intermediate lower level goals. This recursive procedure is continued until no rules are found which match the lowest level intermediate goals. As part of this recursive procedure, a tree structure is composed of the if-then statements expressed in the rules.

The inputs and outputs of the function are listed in Table 27.

Table 27. Inputs and outputs of the backwardChain function.

	Name	**Object type**	**Explanation**
INPUTS:	topEvent	String	The goal to be matched to the members of the rule set
	ruleList	List of ruleObjects	The set of applicable rules
OUTPUT:	condTree	Tree	Results of already completed stages of the backward chaining process extended with the results of this stage

An overview of the function program code is presented in Table 28.

Table 28. Overview of the program code of the backwardChain function.

```
Create a node in condTree for topEvent (goal of this stage of the backward chaining
process);
call findRules to pick out the matching rules from ruleList;
for each rule in rule_p_vector (the matching rules)
{
        for each premise (left hand side statement) of the rule
        {
                call backwardChain with the premise as topEvent;
        }
}
```

6.4.6 Function findRules

The purpose of the findRules function is to pick the matching rules from the list of applicable rules. The rule matches if the conclusion (right hand side) statement of the rule equals to the given expression. In this context, the given expression is the goal of the current stage of the backward chaining process. The result is returned in the form of a vector which contains pointers to the matching rules in ruleList.

The inputs and outputs of the function are listed in Table 29.

Table 29. Inputs and outputs of the findRules function.

	Name	**Object type**	**Explanation**
INPUTS:	topEvent	String	The goal to be matched to the members of the rule set
	ruleList	List of Frule objects	The set of applicable rules
OUTPUT:	rule_p_vector	Array of pointers to ruleObjects	Pointers to matching rules

An overview of the function program code is presented in Table 30.

Table 30. Overview of the program code of the findRules function.

```
Initialise the elements of the pointer array rule_p_vector to NULL;
For each Frule object in ruleList
{
        Compare topEvent to the conclusion (right hand) side of the rule;
        If topEvent equals to the conclusion (string comparison)
                append rule_p_vector with a pointer to the Frule
object;
}
```

6.4.7 Function substituteVariables

The purpose of the function substituteVariables is to apply the available fluid property data to the tree structure generated by the function backwardChain which describes how the fluid property query can be resolved. When possible, the values of certain fluid properties are determined using an external properties package. Other values are retrieved from the internal library of static property data. The substituteVaribles function goes through the tree structure in a recursive manner. When fluid property variables occur in the nodes, it substitutes them with values. The truth values of the nodes are not evaluated as part of this process.

The inputs and outputs of the function are listed in Table 31.

Table 31. Inputs and outputs of the substituteVariables function.

	Name	**Object type**	**Explanation**
INPUTS:	condTree	Tree	Pointer to the current position in the tree structure generated by backwardChain
	var_list_p	List of Strings	For multiple value processing; not used in the current single value based implementation
	flu	Fluid	The fluid under consideration
	L1	Location	Process conditions at a given position, including fluid compositions (components and concentrations)
OUTPUT:	condTree	Tree	Pointer to the current position in the tree containing the results of already completed stages of the variable substitution process extended with the results of this stage

An overview of the function program code is presented in Table 32.

Table 32. Overview of the program code of the substituteVariables function.

```
Extract the event text from the tree node under consideration;
Repeat
{
        Skip special characters and expressions within double quotes;
        Extract the next word from the event text;
        Compare the word to predefined list of fluid property variables;
        If the word is a fluid property variable
        {
                If the value can be obtained from the properties package
                {
                        Call prepareCallPropertyCall to prepare a call to the prop. package;
                        Call shz_calc_prp to calculate the property value;
                }
                else retrieve the property value from the internal library;
        }
        else if the word is a process variable (pressure or temperature)
        {
                retrieve the value from the location object L1;
        }
        else exit with no value;
        If a value was found
        {
                Substitute the word with the value in the event text;
                Revise accordingly the event text in the tree node under consideration;
        }
}
until the end of the event text has been encountered;
Call substituteVaribles to go through the remaining nodes of this branch of condTree.
```

6.4.8 Function preparePropertyCall

In order to call the shz_calc_prp function with the required parameters, the necessary data has to be collected and assigned to the predefined structure. The purpose of the preparePropertyCall function is to carry out those operations. One motivation for the implementation of the preparative operations as a separate function is that only this function needs to be modified in case changes are made to the property package calling mechanism.

The inputs and outputs of the function are listed in Table 33.

Table 33. Inputs and outputs of the preparePropertyCall function.

	Name	**Object type**	**Explanation**
INPUTS:	routine	Symbol	Fluid property request keyword from a fixed list defined as part of the property package
	L1	Location	Process conditions at a given position, including fluid compositions (components and concentrations)
OUTPUT:	pPrpSpec	Call structure for shz_calc_prp	Structure containing a definition of the fluid and process conditions

An overview of the function program code is presented in Table 34.

Table 34. Overview of the program code of the preparePropertyCall function.

```
Set the pPrpSpec process condition data based on the corresponding data in L1;
Extract from L1 the list of fluid components and their fractions;
For each fluid component
{
        Add an element to the pPrpSpec fluid data array and set the data;
}
```

6.4.9 Function shz_calc_prp

The purpose of the shz_calc_prp is to make the services of the property package available to the calling program. When called with appropriate input data, the function returns a value for the requested property. The actual process conditions (pressure and temperature) and the composition of the fluid are taken into account in the calculation of property values.

The inputs and outputs of the function are listed in Table 35.

Table 35. Inputs and outputs of the shz_calc_prp function.

	Name	**Object type**	**Explanation**
INPUT:	pPrpSpec	Call structure for shz_calc_prp	Structure containing a definition of the fluid and process conditions
OUTPUTS:	pPrpRes	Return structure	Structure containing the result of the calculation (either as integer or float)
	iErr	Error flag	Shows whether the calculation was successful or not

The shz_calc_prp function program code is part of the property package owned by the developer company and has not been made available to others. As part of the STOPHAZ project, both Aspentech and Hyprotech delivered a property package for use in the fluid property considerations.

6.4.10 Function evaluateNodes

The purpose of the function evaluateNodes is to calculate a truth value as a reply to the fluid property query. The evaluation of the requested fluid property follows the mechanism defined in the fluid rules and represented, as a result of the backwardChain and substituteVariables functions, in the form of a tree with conditions specific to the studied case. The conservative approach followed implies that the truth value can be ”false” only when the evaluation involves no unknown properties and all the conditions related to known properties evaluate to ”false”.

The inputs and outputs of the function are listed in Table 36.

Table 36. Inputs and outputs of the evaluateNodes function.

	Name	**Object type**	**Explanation**
INPUT:	condTree	Tree	Pointer to the current position in the tree structure generated by backwardChain and manipulated by substituteVariables
OUTPUT:	condTree	Tree	Pointer to the current position in the tree containing the results of already completed stages of evaluation process extended with the results of this stage

An overview of the function program code is presented in Table 37.

Table 37. Overview of the program code of the evaluateNodes function.

```
Set the default value of the current node to ”true”;
If the node is an intermediate one
{
        Call evaluateNodes to first evaluate the branch of condTree underneath the
        current node;
        Evaluate the current node based on the truth values of its child nodes;
}
else if (the node is a terminal one and) the event text includes no unevaluated
variables
{
        Detect the arithmetic operator (>, <, = or !=);
        Interpret the left hand side of the statement;
        Interpret the right hand side of the statement;
        Evaluate the statement by comparing the two sides as defined by the arithmetic
        operator;
}
```

6.5 Evaluation of methodology development phase 2 results

6.5.1 Evaluation approach

The objectives for fluid model development in connection to AutoHAZID were stated in section 6.3. The main goal of the development was to achieve a significant improvement in the hazard identification capability of AutoHAZID. This means that more relevant hazards (feasible, likely and important) should be identified and fewer irrelevant hazards (infeasible, unlikely or unimportant) should be suggested. At the same time, the efficiency of AutoHAZID should improve because less time is spent in exploring irrelevant branches of the causal chains.

The limited implementation of the fluid model features included in AutoHAZID as part of the release of the final prototype of the STOPHAZ software tries to achieve the main objective without causing problems for the efficiency of AutoHAZID and thus weakening the improvements related to the second objective. The two-stage evaluation exercise described in this section is based on this limited implementation of the fluid model.

The implementation was done using the C++ programming language. Only standard features of the programming language were used in order to ensure that the same program code can be executed in Unix workstation computer environment and in personal computer (PC) environment with Microsoft Windows 95. Due to the research oriented nature of the project, very little emphasis was put on the systematic testing of the software. However, due to the experimental nature of the developed software, each new piece of software had to be tried out with a set of test examples in order to ensure that the desired performance had been achieved.

The first stage of evaluation was carried out using a set of artificial test cases extracted from the water separator example presented in a well-known article on HAZOP by Lawley (1974). With this set of small size test cases it was possible to compare the performance of AutoHAZID with the limited fluid model capabilities to a version which was modified for the purposes of this exercise to skip all fluid related considerations.

Because the fluid related considerations are implemented in the AutoHAZID model library in such a way that they always aim to reject candidate cause-consequence relationships, only the capability of AutoHAZID to reject irrelevant hazards from its output could be evaluated by this comparative approach. An attempt to evaluate the capability of AutoHAZID to identify relevant hazards will be reported in connection to the second stage of evaluation.

The comparative approach also allowed the evaluation of AutoHAZID's efficiency. This was done by recording the execution time for each test case with the limited fluid model and without it.

The second stage of evaluation was based on the results of an evaluation workshop organised by the STOPHAZ project. In this workshop, the final prototype of AutoHAZID was evaluated by comparing the HAZOP results generated by AutoHAZID to the results of a human HAZOP team in five industrial example cases. The fluid model features were not explicitly evaluated in the workshop although it is clear based on the results of the first stage of evaluation that already the limited implementation had a significant impact on the capability of AutoHAZID to reject irrelevant hazards. A discussion is added after the presentation of the general results of the workshop dealing with the potential of more advanced fluid model features to support the identification of such hazards which the evaluated version of AutoHAZID failed to identify.

6.5.2 First stage: comparative evaluation

Test cases

The LU researchers created a set of five test cases as a testbench for the development of AutoHAZID models. The original objective was to solve these test cases of increasing complexity one by one using AutoHAZID. The results generated by AutoHAZID were subjected to critical evaluation by experienced HAZOP practitioners, and a number of improvements were made to the models each time before moving to the next test case of more complexity.

The test cases were extracted from an example design of a water separator system published in Lawley (1974). Diagrams of the test cases are attached as Appendix 6.

The first two test cases have a narrower scope than the rest. The test cases cover the water separator system as follows:

Test case 1: Feed section.
Test case 2: Simplified separator.(no pressure control)
Test case 3: Feed plus simplified separator (no pressure control)
Test case 4: Separator with pressure control.
Test case 5: Feed plus separator with pressure control.

The test cases 1 and 2 are the most interesting from the fluid model point of view because they cover the two parts of the water separator system which are used as the basis of the remaining test cases. When the evaluation of the test case 1 and 2 results has been done, only the possible additional findings need to be examined for the test cases 3 to 5.

Feed section

The feed section is a typical simple system for pumping a fluid from one location to another. Such systems are basic elements for all fluid based processes. Therefore, the use of the feed section as the first test case to assess and improve the quality of AutoHAZID models was justified.

When the feed section was reused for the purposes of this evaluation study, a proven version of AutoHAZID released in connection to the final prototype of the STOPHAZ software was used. As has been mentioned before, this version includes a limited fluid model which is able to carry out simple fluid property based considerations. The feed section was analysed using this version of AutoHAZID and another version in which all fluid property based considerations had been turned off. The results are shown in Table 38.

Table 38. Identification results in the feed section test case.

	Causes identified	Consequences identified
No fluid model	**47**	**26**
Limited fluid model	**34**	**20**
Decrease of number, %	**28**	**23**

The limited implementation of the fluid model features included in AutoHAZID in the final prototype of STOPHAZ only attempts to reject irrelevant causes and consequences. Therefore, the significance of its impact can be assessed by comparing the number of identified causes and consequences obtained using two versions of AutoHAZID which only differ from each other in this respect. The above presented results show a 28 % decrease in the number of identified causes and a 23 % decrease in the number of identified consequences. The complete HAZOP listings of this test case are presented in Appendix 7.

The results of this test case are very promising concerning the improved capability of AutoHAZID to reject irrelevant hazards from its output. A closer look at the rejected causes and consequences shows that AutoHAZID with the limited fluid model had correctly decided to reject them. The rejected causes and consequences are listed in Table 39 together with the relevant fluid property query (see Table 8 in section 6.3.1) and a justification for the rejection. It should be noted that causes can sometimes get rejected together with a consequence and vice versa. For example, when only one consequence has been identified for a certain fault propagation path and this consequence gets rejected due to fluid considerations, the cause is also removed from the results because causes without consequences are not reported. Such indirectly rejected causes and consequences are not listed in Table 39.

Table 39. Rejected causes and consequences in the feed section test case.

Rejected Item	Query	Justification
CAUSES		
blockage - solids	solidsPresent	fluid composition look-up
blockage - frozen fluid	freezing	process temperature
flashing across valve	liquid & nearBoilingPt	not close to boiling point
CONSEQUENCES		
toxic release	toxic	hexane defined as non-toxic
non-hazardous release	not (toxic) & not (flammable)	hexane defined as flammable
cavitation due to evolution of gases	dissolvedGas	fluid composition look-up
brittle fracture	brittle	material property look-up
possible fluid decomposition	decomp	no decomposition hazard

It is also important to keep the execution time under control. A comparison of execution times recorded for the feed section test case using the two versions of AutoHAZID is presented in Table 40.

Table 40. Execution times in the feed section test case.

	Execution time, s
No fluid model	**107**
Limited fluid model	**93**
Decrease of time, %	**13**

The use of the limited fluid model led to a slight decrease of execution time in this case. This indicates that it is possible to extend AutoHAZID with simple and useful fluid model features with no negative impact on execution time, and even to speed it up depending on the efficiency of the implementation.

Simplified separator

Together with the transfer of a fluid from one location to another, storage is a typical function of a fluid based process. The second case study dealing with a simplified separator system represents this aspect of fluid processing.

The comparative study of AutoHAZID analysis quality was done in a manner similar to the study of the first test case. The results are again presented in terms of the number of identified causes and consequences in Table 41. The complete HAZOP listings are presented in Appendix 8.

Table 41. Identification results in the simplified separator test case.

	Causes identified	Consequences identified
No fluid model	**37**	**29**
Limited fluid model	**26**	**13**
Decrease of number, %	**30**	**55**

The decrease was very clear both in the case of causes and in the case of consequences. Because the models used for the analysis of this case are almost completely separate from the models used for the analysis of the first test case, the results appear to confirm that even a limited implementation of fluid property based considerations can have a significant impact on the capability of AutoHAZID to reject irrelevant hazards.

The rejected causes and consequences are presented in Table 42 together with the relevant fluid property query (see Table 8 in section 6.3.1) and a justification of the rejection. Indirectly rejected items are left out from the table in the same way as in the discussion of the feed section test case. Once again the rejections are correct.

Table 42. Rejected causes and consequences in the simplified separator test case.

Rejected Item	Query	Justification
CAUSES		
blockage - solids	solidsPresent	fluid composition look-up
blockage - frozed fluid	freezing	process temperature
CONSEQUENCES		
solid settling	solidsPresent	fluid composition look-up
potential layering and rollover	rollover	inlet fluid definitions
toxic release	toxic	hexane defined as non-toxic
non-hazardous release	not (toxic) & not (flammable)	hexane defined as flammable
potential static fire/explosion hazard	flammable (vapour space)	nitrogen defined as not flammable
increased port erosion - dissolved gas	dissolvedGas	fluid composition look-up
increased port erosion	solidsPresent	fluid composition look-up
frothing	foamer	default value = false
increased vibration due to flashing liquid	nearBoilingPt	not close to boiling point
air ingress - explosion risk	flammable (vapour space)	nitrogen defined as not flammable
toxic vapour release	toxic (vapour space)	nitrogen defined as non-toxic
flammable vapour release	flammable (vapour space)	nitrogen defined as not flammable
freezing	freezing	process temperature
crystallisation	crystalliser	default value = false
increased corrosion	corrosion	default value = false
toxic liquid release	toxic	hexane defined as non-toxic

The results in terms of execution time are presented in Table 43.

Table 43. Execution times in the simplified separator test case.

	Execution time, s
No fluid model	**63**
Limited fluid model	**62**
Decrease of time, %	**2**

In spite of the remarkable improvement in the rejection of irrelevant hazards, there seems to be no negative impact on the execution time. However, the very small decrease obtained in the study of this test case indicates that more benefit can be expected from more accurate hazard identification than improved efficiency.

Combined system without pressure control

The next test case was composed by combining the feed section to the simplified separator. Therefore. The HAZOP results obtained are largely the same than the results obtained from the previous test cases put together.

Linking the two systems together caused AutoHAZID to propose such causes which relate to the feed section for the consequences identified for the separator (and vice versa). These additional causes correspond to a detailed explanation of the feed inlet associated causes already considered when the separator was studied in isolation. Table 44 presents an illustrative example of a cause-consequence relationship taken from the reports of test case 2 and test case 3.

It is easy to see that the four groups of causes from test case 3 are directly related to the four individual causes from test case 2. Therefore, these causes are not really new compared to test case 2. When compared to test case 1, these cause may or may not appear in the report depending on whether they happen to belong to one of the reported cause-consequence relationships.

Table 44. Example of cause-consequence relationship correspondence in the test cases 2 and 3.

bufferTank1 is the separator *dummyHead1 is its feed inlet*	
Test case 2 results:	
CONSEQUENCES:	Solid settling Potential layering and rollover
CAUSES:	dummyHead1 blockage – solids bufferTank1 morePressure vapour dummyHead1 blockage – frozen fluid dummyHead1 low pressure upstream
Test case 3 results:	
CONSEQUENCES:	Solid settling Potential layering and rollover
CAUSES:	valve1 blockage – solids levelControlValve blockage – solids halfMileLine blockage – solids valve3 blockage – solids bufferTank1 morePressure vapour levelControlValve blockage – frozen fluid halfMileLine blockage – frozen fluid valve1 blockage – frozen fluid valve3 blockage – frozen fluid levelControlValve loss of control levelControlValve leak to environment valve1 leak to environment pumpJ1 less pressure out halfMileLine leak to environment valve3 leak to environment valve1 partly closed valve3 partly closed

It was decided not to include new causes of the above presented type in the subsequent quantitative evaluation of new findings. When they were rejected, only a few really new findings remained. Table 45 shows the results obtained by this approach.

Table 45. Additional identification results in the combined system test case.

	Additional causes identified	Additional consequences identified
No fluid model	**3**	**1**
Limited fluid model	**2**	**1**
Decrease of number, %	**33**	**0**

No firm conclusions can be made from these small numbers. However, the results indicate that the fluid model can also have an impact on the study of such event chains which represent fault propagation from one part of a system to another.

The additional identification results contained only one item rejected by the fluid considerations. This item was the cause ’flashing across valve‘ which was correctly rejected as the result of the ’nearBoilingPt‘ query because the fluid had been defined not to be close to its boiling point. This is how flashing is modelled in the current implementation although it is a more complicated phenomenon in reality and should be modelled more accurately in an industrial quality version of the fluid rules.

Comparison of the execution times show a slight decrease as could be expected. The execution time results are presented in Table 46.

Table 46. Execution times in the combined system test case.

	Execution time, s
No fluid model	**163**
Limited fluid model	**152**
Decrease of time, %	**7**

Separator with pressure control

The fourth test case was composed from the simplified separator (Test case 2) by adding pressure control equipment. Therefore, most of the identified cause-consequence relationships were equal to the findings of the test case 2 study. In addition, most of the additional causes represented detailed explanations of deviations in the internal state of the separator. For instance, 12 new causes were identified explaining how low vapour pressure could occur in the separator due to problems with the added pressure control equipment.

It was again decided that these less interesting new causes would be left out from the quantitative evaluation. Table 47 shows the results obtained by this approach. They are in accordance with the results obtained from the study of test case 3.

Table 47. Additional causes and consequences identified when pressure control equipment was added to the separator.

	Additional causes identified	Additional consequences identified
No fluid model	**3**	**3**
Limited fluid model	**1**	**3**
Decrease of number, %	**67**	**0**

Again, the two additional causes rejected due to fluid considerations were occurrences of ‘flashing across valve’ and the decision to reject them was based on the fluid not being cose to its boiling point.

A slight reduction of execution time was again observed. Table 48 presents the execution speed results.

Table 48. Execution speed results obtained when pressure control equipment was added to the separator.

	Execution time, s
No fluid model	**97**
Limited fluid model	**94**
Decrease of time, %	**3**

Combined system with pressure control

The last test case covered all the parts of the example system which had been studied in various combinations in the four previous test cases. The results did not contain any new findings compared to the studies of the test cases 1 to 4. Therefore, no quantitative evaluation of the fluid model impact to the rejection of irrelevant hazards will be presented.

The execution time was almost the same with the two versions of AutoHAZID. It may be that the impact of supporting features like the fluid model becomes less significant when the fault propagation paths becomes longer. The execution time results are presented in Table 49.

Table 49. Execution times in the combined system test case.

	Execution time, s
No fluid model	**196**
Limited fluid model	**195**
Decrease of time, %	**1**

Summary of results

The case studies done in the comparative evaluation were conducted using a version of AutoHAZID released in connection to the final prototype of the

STOPHAZ software. This version incorporates a limited implementation of the intended features of the fluid model system. The comparative evaluation was restricted to the impact of this limited implementation.

The output from the system as released was compared with the output from a special version which was modified in such a way that it skipped all fluid related considerations. A clear improvement was observed in the capability of AutoHAZID to reject irrelevant hazards and the associated cause-consequence relationships. This observation is chiefly based on the results obtained in the first two case studies. The remaining case studies mainly repeated and confirmed the findings of those two studies and did not produce much new information. In the case of the feed section case study which gave worse results than the simplified separator case study, 28 % of candidate causes and 23 % of candidate consequences were rejected as irrelevant by the limited fluid model. A closer look at the rejected causes and consequences showed that they had been correctly rejected. These results indicate that even a limited implementation of fluid property considerations can lead to a significant improvement in the performance of an automated hazard identification system.

Execution time decreased slightly in all the five case studies, the highest reduction being of the order of 10 %. This indicates that it is possible to implement a useful set of fluid property considerations in connection to AutoHAZID without making the fault propagation reasoning more time consuming.

6.5.3 Second stage: study of industrial example cases

Study set-up

In order to validate the performance of AutoHAZID, a comparative study of five process plant examples was carried out in the final phase of the STOPHAZ project. Although these examples provided by the industrial partners of the project were not complete descriptions of a process plant, they were selected to represent various challenging types of processes and their size corresponded to the expected size of process plant sections which would be analysed at one time in an industrial project.

The validation was carried out in a workshop manner. Three parallel groups considered different process plant examples. 3-4 process safety experts from STOPHAZ partner companies participated in each group.

The five process plant section descriptions considered in the workshop were:

- an absorption section,
- a benzene storage section,
- a separation section,
- a trichloroethane storage section, and
- a propane rectification section.

In one of the validation studies, study of the trichloroethane storage section, it was possible to pay special attention to fluid related considerations. Evaluation of the fluid model impact to the hazard identification capability of AutoHAZID was based on a detailed analysis of its results.

Study procedure

A highly structured procedure was followed in order to allow the quantitative analysis of the workshop output. The objective was to identify the extent to which the potential problems identified by AutoHAZID corresponded with those identified by a conventional HAZOP team.

The procedure considers each scenario in the human team's report in turn and determines whether or not AutoHAZID identified the scenario, then the procedure considers the additional scenarios identified by AutoHAZID. In this context, a scenario refers to a fault-consequence link or a fault which has no consequence identified. The detailed steps of the procedure are presented in Table 50.

Table 50. AutoHAZID validation workshop procedure.

Consider each deviation in the HAZOP team's report in turn:

Consider each scenario in the HAZOP team's report in turn:

➔ is the scenario agreed to be a feasible one for the current deviation?

NO – categorise scenario (1) and consider next scenario.

YES – continue.

Has an equivalent scenario been considered previously?

NO – continue.

YES – consider next scenario.

Is there an equivalent scenario in the AutoHAZID output?

NO – categorise scenario (2) and consider next scenario.

YES – does the scenario appear under an acceptable deviation in AutoHAZID output?

NO – categorise scenario (3) and mark on AutoHAZID's report sheet that it has been considered.

YES – categorise scenario (4) and mark on AutoHAZID's report sheet that it has been considered.

➔ consider each protection listed in the HAZOP team's report against the scenario:

is the protection a valid one for the current scenario?

NO – categorise protection (5) and consider the next protection.

YES – is the protection specified for the current scenario in AutoHAZID's output?

NO – categorise the scenario-protection link (6) and consider the next protection.

YES – categorise the scenario-protection link (7) and mark on AutoHAZID's report sheet that it has been considered.

➔ consider the next protection.

Consider each deviation in AutoHAZID's report in turn:

Consider each scenario in the AutoHAZID output:

➔ has it been marked off already?

YES – consider next scenario.

NO – is the scenario agreed to be a feasible one for the current deviation?

NO – categorise scenario (8) and consider the next one.

YES – is the feasible scenario of interest?

YES – categorise scenario (9a) and consider the next one.

NO – categorise scenario (9b) and consider the next one.

Consider each protection listed in AutoHAZID's output against the scenario:

➔ has it been marked off already?

YES – consider next protection.

NO – is the protection a valid one for the current scenario?

YES – categorise scenario-protection link (10) and consider next protection.

NO – categorise scenario-protection link (11) and consider next protection.

The classification method used for classifying the scenarios can be represented in terms of sets as defined in Table 51.

Table 51. Classification method in terms of set definitions.

Set	**Explanation**
A	Scenarios identified by the HAZOP team
$A_{Feasible}$	Feasible scenarios included in A
$A_{Incorrect}$	Incorrect scenarios included in A
B	Scenarios identified by AutoHAZID
$B_{Feasible}$	Feasible scenarios included in B
$B_{Incorrect}$	Incorrect scenarios included in B
$B_{Mislocated}$	Scenarios included in $B_{Feasible}$ which are reported under wrong HAZOP keyword
$B_{Interesting}$	Interesting scenarios included in $B_{Feasible}$
$B_{Uninteresting}$	Uninteresting scenarios included in $B_{Feasible}$
C	Protections identified by the HAZOP team
$C_{Feasible}$	Feasible protections included in C
$C_{Incorrect}$	Incorrect protections included in C
D	Protections identified by AutoHAZID
$D_{Feasible}$	Feasible protections included in D
$D_{Incorrect}$	Incorrect protections included in D

The following equations hold between the defined sets:

$A = A_{feasible} + A_{Incorrect}$

$B = B_{feasible} + B_{Incorrect}$

$B_{Feasible} = B_{Interesting} + B_{Uninteresting}$

$B_{Mislocated} \subset B_{Feasible}$

$C = C_{feasible} + C_{Incorrect}$

$D = D_{feasible} + D_{Incorrect}$.

The classification codes used in the workshop procedure are listed in Table 52 together with an interpretation and a set-based definition of each code.

Table 52. Classification codes used in AutoHAZID validation.

Code	Interpretation	Definition
1	Incorrect scenario identified in HAZOP team's report	$A_{Incorrect}$
2	Valid scenario identified by the HAZOP team but not by AutoHAZID	$A_{Feasible} \cap not(B_{Feasible})$
3	Valid scenario identified by both the HAZOP team and AutoHAZID but under unacceptable guideword in AutoHAZID output	$A_{Feasible} \cap B_{Mislocated}$
4	Valid scenario identified by both the HAZOP team and AutoHAZID under acceptable guideword in AutoHAZID output	$A_{Feasible} \cap B_{Feasible} \cap not(B_{Mislocated})$
5	Protection incorrectly identified by the HAZOP team	$C_{Incorrect}$
6	Protection identified by the HAZOP team but not by AutoHAZID	$C_{Feasible} \cap not(D_{Feasible})$
7	Protection identified by both the HAZOP team and AutoHAZID	$C_{Feasible} \cap D_{Feasible}$
8	Incorrect scenario identified by AutoHAZID	$B_{Incorrect}$
9a	Feasible and interesting scenario identified by AutoHAZID but not by the HAZOP team	$B_{Interesting} \cap not(A_{Feasible})$
9b	Feasible but uninteresting scenario identified by AutoHAZID but not by the HAZOP team	$B_{Uninteresting} \cap not(A_{Feasible})$
10	Valid protection identified by AutoHAZID but not the HAZOP team	$D_{Feasible} \cap not(C_{Feasible})$
11	Incorrect protection identified by AutoHAZID	$D_{Incorrect}$

It should be noted that the classification scheme was devised to be exhausting and not all classifications were necessarily used in any particular analysis.

Summary of general results

The output of the validation workshop was analysed in quantitative terms. The intention was to get quantitative estimates of how well AutoHAZID identifies hazards compared to a human HAZOP team. These estimates are presented in Table 53.

Table 53. Summary of AutoHAZID validation results.

	Absorption	**Trichloro-ethane storage**	**Propane rectification**	**Benzene storage**	**Separation**
100 *($B_{Feasible}$ / B) ***Scenarios identified by AutoHAZID judged to be correct, %***	49	33	69	83	53
100 * ($B_{Interesting}$ / B) ***Scenarios identified by AutoHAZID judged to be correct and of interest, %***	10	29	24	27	NA
100 * (($B_{Feasible}$ ∩ $A_{Feasible}$) / $A_{Feasible}$) ***Scenarios identified by HAZOP team also found by AutoHAZID, %***	36	33	60	50	53
100 * ($D_{Feasible}$ / D) ***Protections identified by AutoHAZID judged to be correct, %***	10	29	NA	77	NA

NA = NOT AVAILABLE. Those studies were not concluded.

In the case of the benzene storage system, it should be noted that the plant descriptions considered by the HAZOP team and AutoHAZID were thought to be so different that the validation exercise was rendered unreliable as a test of AutoHAZID's performance.

The main conclusion drawn from the quantified results is that AutoHAZID appears to find a reasonable portion of the scenarios identified by a HAZOP team. In the validation workshop cases, this percentage varied between 33 and 60 %. It has been estimated based on discussions with potential commercial exploiters of AutoHAZID that an automated hazard identification tool would be exploitable if it could identify 25% of the scenarios identified by a human HAZOP team.

However, the useful output of AutoHAZID is heavily masked by a large amount of information which is either correct but uninteresting or simply incorrect. This could stand in the way of the industrial use of AutoHAZID unless improvements can be made.

Incorrect information is often the result of not taking order of magnitude issues into account during fault propagation. This is typical for a system based on qualitative models of equipment behaviour. A discussion of this problem can be found in section 6.3.6. In connection to the validation exercise, flow deviations in a heat exchanger were mentioned as an example. If the cooling water to a heat exchanger is lost the temperature of the process stream will rise and AutoHAZID will report possible overtemperature of the exchanger. Although identified by AutoHAZID as a potential hazard, overtemperature is normally not a problem for the exchanger itself as it is designed to withstand the normal process inlet temperature.

When it comes to decreasing the amount of uninteresting information in AutoHAZID output, different users may have different opinions on what is uninteresting. Therefore, the best approach to improving AutoHAZID in this respect is to offer the users appropriate options for filtering the output. The consequence severity threshold option already exists in AutoHAZID for that purpose. One of the workshop groups considered leakage scenarios uninteresting. Filtering out leakage scenarios could be another useful filtering option.

The performance of AutoHAZID can also be evaluated with respect to the number of false positives and false negatives. A false positive is an incorrect scenario identified by AutoHAZID, and a false negative is a valid scenario not identified by AutoHAZID. Table 54, which is actually another interpretation of

the figures presented in Table 53, summarises the results of the evaluation workshop in terms of false positives and false negatives.

Table 54. Summary of the evaluation workshop results in terms of false positives and false negatives.

	Absorption	**Trichloro-ethane storage**	**Propane rectification**	**Benzene storage**	**Separation**
100 * ($B_{Incorrect}$ / B) ***False positives, % from AutoHAZID results***	51	67	31	17	47
100 * (1 – ($B_{Feasible}$ / $A_{Feasible}$)) ***False negatives, % from HAZOP team results***	64	67	40	50	47

Fluid model oriented study of the evaluation results

During the workshop-type validation studies carried out as part of the STOPHAZ project, the developed fluid model features were not taken into full use. A conscious decision was made to concentrate on the validation of the core features of AutoHAZID. The full rule-based implementation of fluid property reasoning and the compatibility check feature were considered to be outside of the core.

This was necessary because AutoHAZID is composed of several independent innovative methods. The fault propagation method based on signed directed graphs (SDG) is the core of AutoHAZID. Other methods include model construction, fluid property reasoning, compatibility check features, order-of-magnitude processing, and filtering based on user preferences. Validation of the SDG-based fault propagation method in isolation from the complementary methods simplified the interpretation of the study results and helped to keep the number and size of required test cases low enough to allow the arrangement of the validation as an expert workshop with a limited size and duration. The study of industrial examples provided by the industrial partners of STOPHAZ was a challenge to AutoHAZID as such. Therefore, it was felt necessary to minimise

the complexity of the plant descriptions fed into the system. As a result, such details were omitted which are only used by the fluid model and other advanced features of AutoHAZID.

The validation results still form a good basis for an extended validation which covers other methods which are expected to improve the performance of AutoHAZID. The fluid model features composed of the fluid property reasoning method and the compatibility check feature are such methods. What needs to be done in order to validate these complementary methods is to look at the scenarios not identified by AutoHAZID in its core version. In other words, the extended validation should be based on examining the set of false negatives determined as one outcome of the evaluation workshop (see Table 54). If it can be shown that a complementary method enables AutoHAZID to identify some of those scenarios, this can be interpreted as an indication of the validity of that method.

In the following discussion, the fluid model features of AutoHAZID will be evaluated based on the results of the AutoHAZID validation workshop. The introduction of complementary fluid model features is assumed not to cause AutoHAZID to fail to identify any of the scenarios which were successfully identified without a full implementation of the fluid model.

It was decided to pay more attention to fluid modelling issues in one of the workshop sessions which dealt with one of the considered process plant examples. The process plant example considered in this session was the trichloroethane storage system. A process diagram of the system as drawn by the LU STOPHAZ project group using the Graphical Tool which is part of the STOPHAZ prototype software is attached as Appendix 9. The main characteristics of this system are summarised in the following:

- Crude BTri (B-trichloroethane) is pumped at 4 bara and 20-50ºC through a long line into a bunded stock tank and further through a system of two parallel pumps to an outlet leading to a reactor system.

- The level in the stock tank is not controlled, but a high level will cause an actuated valve in the inlet line to close. In order to guard agaist liquid hammer, this valve closes slowly. If an extra high level were to occur, the

contents would overflow into the bund through a line which is sealed with a bursting disc.

- A small flow of nitrogen flows into the tank through a bubbler. If the pressure in the tank fails, a pressure controller opens a control valve in a separate supply of nitrogen. The tank vents through a scrubber to atmosphere.

- The tank is fitted with a pressure/vacuum relief valve and a fire relief hatch.

This example is suitable for this kind of complementary study because only 33% of the scenarios identified by the human HAZOP team were identified by AutoHAZID.

The scenarios identified by the human HAZOP team and not identified by AutoHAZID are listed in Table 55 and a case by case discussion of them follows.

Table 55. Scenarios not identified by AutoHAZID in the trichloroethane test case.

ID	Item	Deviation	Causes	Effect or Hazard
1	Trichloroethane in	High pressure	Closure of shutdown valve	Overpressure of feed pipeline
2	Trichloroethane in	Abnormal composition	Abnormal composition at process inlet	Density change (leads to problems with instrument readings)
3	Vapour system	No flow	Blockage due to water freeze in scrubber	Pressure rise in tank
4	Trichloroethane out	High flow	Failure of flow controller	Change of outlet composition
5	Trichloroethane out	Low flow / No flow	Loss of instrument air	Change of outlet composition
6	Trichloroethane out	Low flow / No flow	Pump failure	Change of outlet composition
7	Trichloroethane out	Reverse flow	Organics pumped in from connecting line	Potential to recycle to the trichloroethane feed
8	Trichloroethane out	Testing	Hydraulic proof testing	Wetting of pipeline
9	Recycle system	Low flow	Pump seal failure	Spraying of product local to pump
10	Tank connections	Contamination	Water ingress	Corrosion

Scenario 1: This is a problem of the pipeline model. **Not a fluid model issue.**

Scenario 2: The susceptibility of instruments to changes in fluid density can be represented in the SDG models of instruments. If this is done, then potential instrument problems will always be flagged when composition deviations are considered. **Fluid model issue.**

Scenario 3: Freezing can be included in the equipment models as one of the faults. When a fluid property query is attached to it which checks if the fluid has a potential to freeze due to a relatively high freezing point, freezing will only be suggested when appropriate. **Fluid model issue.**

Scenario 4: This obvious scenario was not reported by AutoHAZID because no consequences had been found in relation to high flow out from the process section. In the analysis settings defined for the validation workshop, AutoHAZID had been told not to report faults with no consequences. **Not a fluid model issue.**

Scenario 5: No consequences had been identified in relation to no/low flow out from the process section either. **Not a fluid model issue.**

Scenario 6: This scenario was also not reported due to the lack of consequences. In addition, the configuration with two parallel pumps and a recycle line makes it even more difficult for AutoHAZID to consider flow deviations at the process section outlet. **Not a fluid model issue.**

Scenario 7: Reverse flow leading to possible upstream contamination was reported as one of the scenarios. However, AutoHAZID did not report the fact that two fluids are mixed together at the downstream end of the section, and reverse flow could cause the flow of this mixture or one of the fluids upstream along the wrong line. The proper reporting of this type of scenarios could be based on preparatory bookkeeping done when AutoHAZID considers the plant description and what are the correct flow directions. **Not a fluid model issue.**

Scenario 8: It is unclear from the human team's report what the hazard associated with the wetting of the pipeline is. Assuming that there is no hazard other than corrosion, addition of a simple description of this kind of a problem to

the pipeline model would be easy. This simple solution would only cover water as the cause for corrosion, and it would cause AutoHAZID to report the scenario in connection to every connection which is defined as a pipeline. More advanced solutions would require fluid model type consideration of corrosion. **Not a fluid model issue.**

Scenario 9: Pump leaks have been reported but not exactly in this form. **Not a fluid model issue.**

Scenario 10: Similar to scenario 8, this scenario could easily be identified if corrosion due to water ingress were added to the model of a vessel. **Not a fluid model issue.**

In the case of the trichloroethane process example, the full use of the fluid model system would have helped AutoHAZID to identify two of those scenarios which it had failed to identify. This could have produced an increase of the identification capability from 33% to 46% from the scenarios identified by the human HAZOP team.

It should be noted that a comparison to the results of a human HAZOP team could in some cases give misleading results. For instance, in the HAZOP study of the separation example the human team has identified no fluid related hazards. Table 56 gives the number of fluid related scenarios and the total number of scenarios in the five example processes studied in the validation workshop. The presented figures should be taken as rough estimates because detailed analysis of fluid related issues was not performed in cases other than the trichloroethane storage case.

Table 56. Number of fluid related scenarios in relation to the total number of scenarios in the validation workshop examples.

	Absorption	**Trichloro-ethane storage**	**Propane rectification**	**Benzene storage**	**Separation**
Number of scenarios related to fluid issues	11	2	4	10	0
Total number of scenarios identified by the HAZOP team	25	15	23	42	18
Percentage, %	44	13	17	24	0

It might be expected that the examples with a lower percentage of scenarios associated with fluid issues would be easier for AutoHAZID. However, the validation summary results presented in Table 53 do not support that assumption. The highest percentage of identification compared to the HAZOP team, 60%, was achieved with the propane rectification example which has a relatively low fluid issue percentage. However, the lowest percentage of identification, 33%, resulted from the study of the trichloroethane example which also has a low fluid issue percentage.

The percentage of scenarios associated with fluid issues seems to correspond more closely to the percentage of identified interesting scenarios. This percentage was lowest, 10%, with the absorption example which has the highest fluid issue percentage. The highest percentage of interesting scenarios was achieved with the trichloroethane storage example which has a relatively low fluid issue percentage. Unfortunately, the percentage of interesting scenarios was not calculated for the separation example for which the number of scenarios related to fluid issues was zero.

If there is a relationship between the number of scenarios associated with fluid issues in a HAZOP study and the capability of AutoHAZID to identify correct and interesting scenarios, the degree to which AutoHAZID understands what goes on in the process could explain this relationship. The more important the role that fluid issues play in a HAZOP study, the less AutoHAZID understands about the process unless very sophisticated fluid model features are put in place. Therefore, without such features, AutoHAZID tends to produce lot of information which is does not properly apply to the process under the actual circumstances. From this, a conclusion could be drawn that the fluid model will play a very significant role when more sophisticated methods are put in place to screen out invalid and uninteresting results from AutoHAZID's output.

7. Discussion

7.1 Methodological problems

In their study of early computerised hazard identification systems, Suokas and Karvonen (1985) reported that the insufficient quality and coverage of the model library for the description and analysis of real processes was a major problem of these systems. In connection to the development of HAZOPTOOL in the first methodology development stage of the work described in this thesis, it was also noticed that the quality of the causal fault propagation knowledge remained far from sufficient.

The development of AutoHAZID was based on earlier fault propagation modelling work of Loughborough University researchers (Kelly and Lees 1986a, b, c and d; Parmar and Lees 1987; Zerkani and Rushton 1992; Chung 1993). The model library of AutoHAZID has reached a high level of sophistication and, thus, it formed a good basis for the development of fluid property reasoning techniques. Although some extensions to the modelling methodology have been proposed (see section 6.3.6), it was found possible to leave the development of causal fault propagation models outside the scope of this work.

Deficiencies in the causal models lead to poor performance in relation to the correctness and completeness of the resulting HAZOP study. This is due to the fact that the current systems are based on highly generalised knowledge on the behaviour of process equipment and can not perform advanced reasoning on the special characteristics of the chemical process. In addition to trying to improve the quality and coverage of the model library, this problem can also be tackled by developing techniques which enable the knowledge-based HAZOP program to process the models in a more intelligent manner. Fluid property reasoning in a strong candidate for a technique of this kind.

Even when an automated hazard identification system is extended with appropriate advanced reasoning techniques, it is very difficult to reach a sufficient level of detail with knowledge-based HAZOP systems. This is because the models would have to be made highly conditional on process-specific issues to handle detailed information correctly. Adding a lot of conditions related to

process-specific issues would significantly complicate the modelling task, and it would also lead to an increase of required data input effort from the user.

The laborious nature of the data input stage is one deficiency of current knowledge-based HAZOP systems. In this work, data input techniques were not studied. However, efficient utilisation of the supplied data in a context sensitive manner would motivate the users to put more effort into the data input stage. Advanced techniques such as fluid property reasoning enable knowledge-based HAZOP programs to make better use of the plant-specific data made available to them.

It is characteristic of the original HAZOP technique, due to its purely qualitative nature, that it does not provide the quantitative ranking of the identified problems. The result of a HAZOP study is a set of identified hazards with their potential causes. The most severe problems are well identified but, at the same time, a number of less relevant event chains are suggested. Some of these findings may even prove to be incorrect or clearly irrelevant when they are examined more closely in a quantitative manner.

Reasoning about the order of magnitude associated with a process deviation caused by a given fault is a complicated issue. The presence of ambiguities in qualitative reasoning on the behaviour of a chemical process has been widely recognised. For instance, Vaidhyanathan and Venkatasubramanian (1996b) have extended their HAZOPExpert system with the use of a semi-quantitative methodology to overcome this problem. The fluid property reasoning methodology proposed in this thesis includes the consideration of fluid properties in quantitative form.

Basically, resolving queries related to the properties of the processed chemicals and using the results to add conditional elements to the identification of event chains is methodologically quite straightforward. However, a number of questions still need to be answered before high performance of fluid property reasoning can be reached:

- How to estimate the magnitude of change in process conditions in the presence of a process variable deviation expressed in qualitative terms?

- How to estimate the properties of a mixture of chemicals when no data source covers that mixture or when the exact component ratios are uncertain?

- What is an appropriate way to model the behaviour of an intended reaction in abnormal conditions?

These open questions do not complicate the use of the developed methodology as such. However, without answers to such questions, it becomes difficult to carry out case studies to demonstrate the benefits from its use.

Another factor which makes the empirical testing of the fluid property reasoning methodology very problematic is its inevitable negative effect to the speed of automated HAZOP. The use of fluid property reasoning in its complete version within fault propagation would mean that lots of rare consequences are considered and checked against the actual properties of the processed fluid. Moreover, numerous similar checks should also be made in relation to the causal propagation of events when the possible links between potential faults and consequences are identified.

In spite of the efforts described in this thesis and other work, various aspects of fluid property reasoning still remain poorly covered. Recommendations will be presented in Chapter 9 for dealing with the following issues in the further development on fluid property reasoning methodologies:

- Properties of mixtures
- Dependency on process conditions
- Downstream impacts of unintended reactions
- Order of magnitude considerations
- Filtering philosophy
- Data acquisition from the user
- Properties of materials and reactions.

7.2 Summary of the results

A novel methodology for fluid property reasoning in connection to knowledge-based HAZOP has been proposed in this thesis. Building on the earlier achievements of Loughborough University (LU) and Technical Research Centre of Finland (VTT) researchers, the methodology enables knowledge-based hazard identification programs to make a more intelligent assessment of the potential hazards and their causes.

Early work on this topic at LU mainly concentrated on the consideration of leak, blockage, materials of construction susceptibility, and hazards related to a chemical release to atmosphere (Parmar and Lees 1987, Zerkani and Rushton 1992). VTT started to study the impacts of fluid and reaction issues on knowledge-based HAZOP in the STARS project (Heino et al. 1992), in which independent knowledge bases and associated interactive tools were developed for chemical substances, intended chemical reactions and unwanted chemical reactions (Christensen et al. 1991). The elements of the earlier achievements of both the LU and VTT researchers were included in the scope of the methodology development reported in this thesis.

The Purdue University researchers have recently taken similar issues into consideration (Vaidhyanathan and Venkatasubramanian 1996b). In their HAZOPExpert system, process materials and their properties form a part of the process-specific knowledge. The process-independent knowledge on abnormal causes and adverse consequences makes use of this information when required. In the reported work of Purdue University researchers, this generic framework has only been applied to simple fluid property checks such as freezing, fire hazards, and health hazards related to loss of containment. The objective of the work described in this thesis was to improve the capability of knowledge-based HAZOP to deal with all fluid property dependent issues, including those which require more complicated techniques than the evaluation of conditions dependent on fluid property values.

In total, six research groups worldwide have recently worked on methods for the computerised HAZOP study of chemical process systems. According to what has been published in major scientific journals relevant to this area of research,

no other significant developments exist concerning the automation of HAZOP studies and related methods such as fluid property reasoning.

Two implementations of the fluid property reasoning methodology have been presented in this thesis. Each was done as part of a contract research project. The first implementation was done in 1992–1994 for the Taiwanese engineering company CTCI Corporation. The European collaborative project STOPHAZ co-ordinated by the UK-based company ICI Engineering Technology served as the framework for the second implementation.

During the two phases of development numerous hours were spent in methodological discussions with highly qualified experts from the STOPHAZ consortium and the Taiwanese engineering company CTCI Corporation. The need for fluid property reasoning was recognised by these experts. In addition, the proposed methodology was reviewed and accepted by them. This can be interpreted as a strong indication of the validity of the proposed methodology.

The proposed fluid property reasoning methodology was successfully implemented in two environments. It was shown that fluid property reasoning can be linked to causal reasoning. External chemical properties packages were also successfully linked to the same environment to support the evaluation of fluid properties in specified conditions.

In the first methodology development phase, a rule-based fluid and reaction property reasoning system was developed in connection to the HAZOPTOOL program. This system was designed to be very flexible and easy to modify. The list of query keywords was not fixed. Instead, the authors of new HAZOP rules were allowed to use any word as a query keyword, and new fluid rules and attributes were to be added in such a way that the system is able to give an answer to the request by backward chaining appropriate rules and referring to related attributes. In the case of HAZOPTOOL, the desire to maximise flexibility led to problems in keeping the evolution of the rule-based unit and fluid models under control. The implementation also become relatively slow due to the extensive processing of symbolic data.

In the second methodology development phase, the LU fault propagation reasoning methodology implemented in the AutoHAZID program was extended

with fluid property reasoning capabilities. The main objective was to implement an intelligent system to resolve fluid property queries. Whenever the relevance of a cause-consequence relationship is dependent on the properties of the processed fluid, AutoHAZID tries to solve the problem by investigating the fluid properties. This can be done by calling the rule-based fluid model system which is able to reason about those issues.

Generic rules were defined for the knowledge-based processing of fluid property data. The rules are stored in the internal library of chemical properties together with static chemical property data. Some of the static property values are only used if a properties package can not be used but also some properties have been included which are not covered by those packages. Such properties include toxicity and reactivity.

Properties packages are able to calculate the values of fluid properties at the specified process pressure and temperature conditions. They can also estimate the properties of mixtures. A link has been implemented from AutoHAZID to two properties packages supplied by simulation software companies. Whenever a properties package has been made available, it will be used automatically to provide better estimates of a number of fluid properties.

In order to be able to consider fluid compatibility issues, a reaction matrix based on a reactivity group approach was added to complement the fluid-oriented part of AutoHAZID. It can be used to identify possible unintended reactions between two fluids. At present, this compatibility check feature has to be called separately from the other fluid related considerations covered by the fluid property queries.

7.3 Utility and limitations of the results

The developed methodology is aimed at the performance improvement of knowledge-based HAZOP. The scope of this study was limited to that application domain. Other improvements to automated HAZOP than those related to fluid property reasoning were not considered.

The accident examples discussed in Chapter 3 link this study to the ultimate goal of preventing serious accidents. The developed fluid property reasoning methodology contributes to the methodological progress towards this goal. It makes information about the role of chemicals as potential accident contributors available to safety analysts and process designers in their computerised environment for safety considerations.

The results of the methodology development reported in this thesis are associated with the accident examples in Table 57. The problems which should have been eliminated in order to prevent the accident have been listed together with the fluid property reasoning features required to identify such problems in knowledge-based HAZOP. The fourth column states whether the listed fluid property reasoning features exist in the versions of HAZOPTOOL or AutoHAZID discussed in this thesis.

Table 57. Accident examples associated with the developed fluid property reasoning features.

Accident Example	**Problem**	**Feature**	**Existence**
Ashton 1917	Hot acid started the fire of wooden staging	• Materials of construction flammability • Explosion threshold	No Both
Texas City 1969	Thermal decompo-sition of fluid	• Abnormal concentrations • Decomposition threshold	No Both
Baton Rouge 1976	Fluid-fluid incompatibility	• Unintended contact of two fluids • Unwanted reaction	AutoHAZID Both
Breahead 1977	Impact of external fire	• Thermal expansion	Both
Bhopal 1984	Fluid-water incompatibility	• Unintended water ingress • Unwanted reaction	HAZOPTOOL Both
Seveso 1986	Change of reaction in abnormal conditions	• Dependence of reaction on temperature	No
Jonova 1989	Sudden increase of tank pressure	• Dependence of vapour pressure on temperature	Both
Stanlow 1990	Fluid-fluid incompatibility	• Unintended contact of two fluids • Unwanted reaction • Impact of an unintended product	AutoHAZID Both No

The table lists eleven different features from which seven exist at least in one of the programs. The remaining four features are known areas for the further improvement of the fluid property reasoning methodology. This indicates that the focus of the development work has correctly been in satisfying requirements which are imposed by real problem areas of industrial safety.

AutoHAZID was subjected to comprehensive evaluation at the final stage of the STOPHAZ project. It was shown that AutoHAZID is able to find a reasonable portion of the scenarios identified by a HAZOP team. In the validation workshop cases, this percentage varied between 33 and 60 %. This performance was achieved despite deliberate restriction of plant description to what was considered reasonable in terms of input effort. It has been estimated based on discussions with potential commercial exploiters of AutoHAZID that an automated hazard identification tool would be exploitable if it could identify 25% of the scenarios identified by a human HAZOP team.

However, the useful output of AutoHAZID is heavily masked by a large amount of information which is either correct but uninteresting or simply incorrect. This could stand in the way of the industrial use of AutoHAZID unless improvements can be made. To study this issue in more detail, a fluid model oriented study of the evaluation workshop results was made which indicated that the fluid model will play a very significant role when more sophisticated methods are put in place to screen out invalid and uninteresting results from AutoHAZID's output.

In comparative testing based on a set of test cases, a clear improvement was observed in the capability of AutoHAZID to reject irrelevant hazards and the associated cause-consequence relationships when the fluid property reasoning methodology was used. Even in the worst case, 28 % of candidate causes and 23 % of candidate consequences were rejected as irrelevant by the limited fluid model features included in the tested version of AutoHAZID. Moreover, a closer look at the rejected causes and consequences showed that AutoHAZID had correctly decided to reject them. The results of the comparative testing indicate that even a limited implementation of fluid property considerations can lead to a significant improvement in the performance of an automated hazard identification system.

Execution time decreased slightly in all the five case studies, the highest reduction being of the order of 10 %. This indicates that it is possible to implement a useful set of fluid property considerations in connection to an automated hazard identification system without making the fault propagation reasoning more time consuming.

In summary, the developed methodology was shown to have the potential to improve the consideration of fluid dependent hazards significantly. However, a vital prerequisite for gaining full benefit from the use of the methodology is that the knowledge-based HAZOP system to be extended with fluid property reasoning features is capable of estimating the process conditions and the fluid composition in abnormal situations.

8. Conclusions

The study of serious accidents which have occurred in the chemical process industry in recent times highlights the need to understand fluid property related phenomena and the interactions between chemicals under abnormal process conditions or with abnormal fluid compositions. Typically, an unwanted chain of events is triggered by an unintended increase of temperature, often coupled with an abnormal composition of the fluid either as a cause or a consequence. Consideration of these issues should be common practice in professional safety analysis work, and computer programs designed to support this work have to be able to deal with them, preferably in an intelligent manner.

Currently, the most advanced features of commercial HAZOP software packages include knowledge libraries, intelligent checklists and analysis process guidance. No automated HAZOP features are included in any of these packages. This is a clear indication of the fact that automated HAZOP still has such problems which make it commercially uninteresting. However, the achievements of the various research groups who have studied the knowledge-based identification of chemical hazards indicate that it should be possible to reach a completeness and coverage of the results which is close to the output of conventional group work HAZOP carried out by a group of human experts.

This thesis describes a novel methodology for fluid property reasoning in connection to knowledge-based HAZOP. Building on the earlier achievements of Loughborough University (LU) and Technical Research Centre of Finland (VTT) researchers, the methodology enables knowledge-based hazard identification programs to make a more intelligent assessment of the potential hazards and their causes.

In the first methodology development phase, a rule-based fluid and reaction property reasoning system was developed in connection to the HAZOPTOOL program. This system was designed to be very flexible and easy to modify.

In the second methodology development phase, the Loughborough University fault propagation reasoning methodology implemented in the AutoHAZID program was extended with fluid property reasoning capabilities. The main objective was to implement an intelligent system to resolve fluid property

queries. Whenever the relevance of a cause-consequence relationship is dependent on the properties of the processed fluid, AutoHAZID tries to solve the problem by investigating the fluid properties. This can be done by calling the rule-based fluid model system. In order to access property values in real process conditions, a link was implemented from AutoHAZID to two properties packages supplied by simulation software companies. In addition, a reaction matrix based on a reactivity group approach was added to complement the fluid-oriented part of AutoHAZID.

AutoHAZID was subjected to extensive evaluation at the final stage of the STOPHAZ project. It was shown that AutoHAZID is able to find a reasonable portion of the scenarios identified by a HAZOP team. The performance was found to reach the level required from a commercially exploitable tool.

However, the useful output of AutoHAZID is heavily masked by a large amount of information which is either correct but uninteresting or simply incorrect. This could stand in the way of the industrial use of AutoHAZID unless improvements can be made. To study this issue in more detail, a fluid model oriented study of the evaluation workshop results was made which indicated that the fluid model will play a very significant role when more sophisticated methods are put in place to screen out invalid and uninteresting results from AutoHAZID's output.

In comparative testing based on a set of test cases, a clear improvement was observed in the capability of AutoHAZID to reject irrelevant hazards and the associated cause-consequence relationships when the fluid property reasoning methodology was used. Moreover, a closer look at the rejected causes and consequences showed that AutoHAZID had correctly decided to reject them. In addition, it was observed that the execution time decreased slightly in all cases.

The two implementations of the chemical reasoning methodology presented in this work show that it is possible to extend knowledge-based HAZOP with a capability to reason about fluid properties and interactions. The results indicate that the potential benefits are large compared to the effort required to make available a reasonable collection of chemical property data and to make the qualitative causal propagation models refer to it.

Based on the results of this work, it is recommended that fluid property reasoning is taken into use in any application of knowledge-based HAZOP. Assuming that the knowledge-based HAZOP system is capable of estimating the process conditions and the fluid composition properly in abnormal situations, the developed methodology has the potential to improve the consideration of fluid dependent hazards significantly.

Future research should be focused on removing the problems inherent in qualitative models of behaviour such as the signed directed graph. An extended method for qualitative causal propagation would be needed which is able to reason with value intervals and uncertainties.

9. Recommendations and future research

Based on the results of this work, it is recommended that fluid property reasoning is taken into use in any application of knowledge-based HAZOP. The results indicate that the potential benefits are large compared to the effort required to make available a reasonable collection of chemical property data and to make the qualitative causal propagation models refer to it.

Future research should be focused on removing the problems inherent in qualitative models of behaviour such as the signed directed graph. Because the magnitude of phenomena is not known, it becomes very difficult to draw any justified conclusions on the impacts of chemicals. Once the hazard under consideration is known, the composition of the fluid and the process conditions could be studied to estimate the required magnitude of a deviation. By considering a fault and its impact on the normal process conditions, the initial magnitude of the phenomenon could be estimated. An extended method for qualitative causal propagation would then be needed which is able to reason with value intervals and uncertainties.

More comprehensive evaluation studies than were possible within the scope of this work should also be carried out. Only through the practical application of the fluid property reasoning methodology it is possible to learn more about its strengths and weaknesses. The further development of the methodology would benefit from the existence of more comprehensive testing results.

Below, detailed recommendations for the further development of AutoHAZID software will be summarised. AutoHAZID would serve as an excellent framework for studies on fluid property reasoning in the short term. Significant improvement in the performance of AutoHAZID could be expected as a result of further work in these areas:

Properties of mixtures: When fluids are mixed together, the properties of the resulting fluid may significantly differ from the properties of the original fluids. To estimate the properties of a mixture, its composition needs to be known. One option is to define a breakpoint corresponding to every possible location of the process where the composition of the fluid can change. In addition to intended changes to the fluid composition (mixer, reactor, etc.), the breakpoint procedure

could cover unintended changes. The plant description could include optional information on ”extra fluid components that could be present”. This would serve as an educated guess of what extra fluids could mistakenly enter the process. Another option would be to consider the possible breakdown of internal barriers, such as closed valves, heat exchanger interfaces and pump seals, which are expected to separate from each other the fluids at the opposite sides of the barrier. Additionally, if the deviation considered in the current fault propagation implies a change in the fluid composition or in the state of matter, the fluid description should be changed accordingly.

Dependency on process conditions: The equipment models of AutoHAZID could be extended to include all possible links between composition deviations and other deviations, bearing in mind that the fluid model should take care of screening out all those links which do not hold under the studied circumstances. One way to handle the changes in process conditions due to fluid composition changes is to evaluate the consequences using changed values of process parameters. Another possibility is to build the dependencies into the equipment models. This would require the addition of arcs which represent the relation between composition deviations and other deviations in the models.

Downstream impacts of unintended reactions: Whenever unintended fluid components can enter the process fluid, the compatibility of the new components with the others should be checked using the compatibility matrix. The consequences of composition deviations in the unit models could be extended to correspond to a generic list of all the possible consequences of mixing incompatible fluids, and the fluid model should take care of screening out the irrelevant consequences using the reaction matrix.

Order of magnitude considerations: What would need to be done to improve the handling of order of magnitude issues in AutoHAZID is to attach quantitative information to the faults in the models. The magnitude of the impact on a process parameter could be defined in terms of a value range, either referring to the absolute value as the result of the fault or to the amount of increase or decrease. In some cases it might be necessary to define the impact in terms of a multiplier value with a range. The use of this kind of value ranges would give AutoHAZID the possibility to reason about the expected magnitude

of the impact and thereby to detect such cases where the cause-consequence chain is clearly unrealistic.

Filtering philosophy: Appropriate options should be offered to the users of AutoHAZID for filtering the output to screen out results which, whilst they may be correct, are uninteresting to them. For instance, some users might want to screen out leakage scenarios. The classification of consequences could be improved in two ways. Firstly, instead of a predefined classification, each actual consequence of an identified event chain could be classified in a context sensitive manner based on all what is known about the event chain. Secondly, the list of consequences used in the models could be reconsidered. For instance, it would make consequence classification and the associated filtering easier if release incidents of different types were given unique names.

Data acquisition from the user: Estimating the size of leak-type faults is very difficult to perform without help from the user. It is almost impossible for the computer to consider and compare various relevant scenarios but a human expert is normally able to give a reasonable estimate. Another type of information which could be acquired from the user concerns the properties of possible unintended reactions. It would be very difficult to make AutoHAZID capable of understanding the reaction and of determining what are the resulting unintended reaction products. A human expert could at least give a quick assessment of what might happen and help AutoHAZID to go ahead with the reasoning based on that assessment.

Properties of materials and reactions: The properties of materials of construction could be handled by the fluid model system in the same way as fluid properties. A set of material property queries to be resolved could be defined. Values for the interesting properties of materials could be stored in an internal data base of materials' properties or retrieved from an external source. In addition, the typical behaviour of different types of reactions in abnormal situations could be described in a way similar to the HAZOPTOOL Chemical Reaction Knowledge Base (RKB).

References

ADL. 1997. HAZOPtimizer for Windows. Arthur D. Little, Inc. Cambridge, Massachusetts, USA. 2 p.

CEP. 1997. CEP software directory. Chemical Engineering Progress. January 1997.

Catino, C. A. and Ungar, L. H. 1995. A model-based approach to automated hazard identification of chemical plants. AIChE Journal, 41, 97–109.

CCPS. 1985. Guidelines for hazard evaluation procedures. Center for Chemical Process Safety (CCPS). AIChE, New York.

Christensen, P., Smith-Hansen, L., Heino, P., Fassera, G. and Poucet, A. 1991. The use of knowledge bases in advanced computer aided safety analysis. Reliability'91, London, June 1991.

Chung, P. W. 1993. Qualitative analysis of process plant behaviour. Proc. of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Chung, P. W. H., Lovegrove, G. and Ali, M. (eds), Gordon and Breach Science Publishers, Edinburgh. pp 277–283.

CIA. 1977. A guide to hazard and operability studies. London. Chemical Industry safety and Health Council of the Chemical Industries Association Limited. 42 p.

De Boisheraud, H. and Eury, S. 1991. Ariane's Vulcain Engine – 1991 A Vintage Year. ESA Bulletin 68. Pp. 67–75.

DYADEM. 1996. DDM-HAZOP 2.1. Advanced process hazard analysis software. Dyadem International Ltd., Richmond Hill, Ontario, Canada. 2 p.

EEC. 1982. Council directive on the major-accident hazards of certain industrial activities (82/501/EEC). Official Journal of the European Communities, No. 230, 5.8.1982, pp. 1–18.

Hatayama, S. 1980. A method for determining the compatibility of hazardous wastes. US Environmental Protection Agency, EPA-600/2-80-076/April 1980.

Heino, P., Kotikunnas, E., Shei, W. F., Shao, C. C. and Chen, C. H. 1995. Computer-aided HAZOP with knowledge-based identification of hazardous event chains. In: Loss Prevention and Safety Promotion in the Process Industries, Volume 1. J. J. Mewis, H. J. Pasman and E. E. De Rademaeker (eds.). Elsevier. pp. 645–656.

Heino, P., Poucet, A. and Suokas, J. 1992. Computer tools for hazard identification, modelling and analysis. Journal of Hazardous Materials, 29. pp. 445–463.

Hollnagel, E. and Cacciabue, C. 1992. Reliability assessment of interactive systems with the system response generator. In: Petersen, K. E. and Rasmussen, B. (eds.). Safety and Reliability ’92. London. Elsevier Applied Science. pp. 140–150.

Kakko, R. 1991. The quantitative risk assessment (QRA) of major toxic hazards. Espoo. Technical Research Centre of Finland, Publications 84. 82 p. + app. 114 p.

Karvonen, I., Heino, P. and Suokas, J. 1990. Knowledge-based approach to support Hazop-studies. Espoo. Technical Research Centre of Finland, Research Reports 704. 52 p. + app. 12 p.

Kelly, B. E. and Lees, F. P. 1986a. The propagation of faults in process plants: 1. Modelling of fault propagation. Reliability Engineering, 16, pp. 1–38.

Kelly, B. E. and Lees, F. P. 1986b. The propagation of faults in process plants: 2. Fault tree synthesis. Reliability Engineering, 16, pp. 39–62.

Kelly, B. E. and Lees, F. P. 1986c. The propagation of faults in process plants: 3. An interactive, computer-based facility. Reliability Engineering, 16, pp. 63–86.

Kelly, B. E. and Lees, F. P. 1986d. The propagation of faults in process plants: 4. Fault tree synthesis of a pump system changeover sequence. Reliability Engineering, 16, pp. 87–108.

Kletz, T. A. 1983. HAZOP & HAZAN. Notes on the identification and assessment of hazards. Rugby. The Institution of Chemical Engineers. 81 p.

Lawley, H. G. 1974. Operability studies and hazard analysis. Chemical Engineering Progress, Vol. 70, No. 4. pp. 45–56.

Lees, F. P. 1996. Loss prevention in the process industries. Second edition. London. Butterworths.

Leone, H. 1996. A knowledge-based system for HAZOP studies. The knowledge representation structure. Computers chem. Engng., Vol. 20, Suppl., pp. S369–S374.

OSHA. 1992. OSHA. Process safety management of highly hazardous chemicals; explosives and blasting agents; final rule, 29 cfr 1910.119. 6356–6417, Federal Register, Department of Labor, United States.

Parmar, J. C. and Lees, F. P. 1987. The propagation of faults in process plants: Hazard identification. Reliability Engineering, 17, pp. 277–302.

Poucet, A. 1983. Computer aided fault tree synthesis. Commission of the European Communities, Nuclear Science and Technology, Report EUR 8707 EN. 44 p.

Poucet, A. 1990. STARS: Knowledge based tools for safety and reliability analysis. Reliability Engineering and System Safety, Vol 30, pp. 379–397.

Preston, M. 1995. STOPHAZ support tools for process hazard and operability studies. In: Mewis, J. J. et al. (eds.) Loss Prevention and Safety Promotion in the Process Industries, Vol. II. Elsevier. pp. 655–660.

Primatech. 1996. PHAWorks 3.0 for Windows. Primatech Inc., Columbus, Ohio, USA. 2 p.

Rouhiainen, V. 1990. The quality assessment of safety analysis. Espoo. Technical Research Centre of Finland. Publications 61. 133 p. + app. 30 p.

Rushton, A. G. 1997. Knowledge-based HAZOPs'. Part 2. European Process Safety Centre (EPSC). 18 p.

Shimada, Y., Suzuki, K. and Sayama, H. 1996. Computer-aided operability study. Computers chem. Engng., Vol. 20, No. 6/7, pp. 905–913.

Shimada, Y., Yang, Z.-X., Song, J.-W., Suzuki, K. and Sayama, H. 1995. Computer-aided operability study for batch plants. Proc. Loss Prevention and Safety Promotion in the Process Industries, Antwerp, Belgium, June 6–9, 1995. Vol. 2. Eds. Mewis, J. J., Pasman, H. J. and De Rademaeker, E. E. Elsevier. Amsterdam. pp. 587–598.

SINTEF. 1995. CARA. Computer aided risk analysis. Version 2.1. Users manual. SINTEF, Trondheim, Norway.

Srinivasan, R. and Venkatasubramanian, V. 1996. Petri net-digraph models for automating HAZOP analysis of batch process plants. Computers chem. Engng, Vol. 20, Suppl., pp. S719–S725.

STOPHAZ. 1997. Final technical report. Deliverable D 7.3, ESPRIT project 8228 STOPHAZ. 6 p. + app. 389 p.

Suh, J. C., Lee, S. and Yoon, E. S. 1997. New strategy for automated hazard analysis of chemical plants. Part 1: Knowledge modelling. J. Loss. Prev. Process Ind., Vol. 10, No. 2, pp. 113–126.

Suh, J. C., Lee, S. and Yoon, E. S. 1997. New strategy for automated hazard analysis of chemical plants. Part 2: Reasoning algorithm and case study. J. Loss. Prev. Process Ind., Vol. 10, No. 2, pp. 127–134.

Suokas, J. 1985. On the reliability and validity of safety analysis. Espoo. Technical Research Centre of Finland, Publications 25. 69 p. + app. 8 p.

Suokas, J. and Karvonen, I. 1985. A comparison of automatic fault tree construction with hazard and operability study. Espoo. Technical Research Centre of Finland, Research Reports 330. 36 p. + app. 8 p.

Taylor, J. R. 1982. An algorithm for fault tree construction. IEEE Transactions on Reliability, R-26, 2. Pp. 137–146.

Technica. 1991. HAZSEC. DnV Technica, London, UK. 1 p.

Technica. 1994. SAFETIProfessional 5.1. DnV Technica, London, UK. 1 p.

Vaidhyanathan, R. and Venkatasubramanian, V. 1995. Digraph-based models for automated HAZOP analysis. Reliability Engineering and System Safety, 50, pp. 33–49.

Vaidhyanathan, R. and Venkatasubramanian, V. 1996a. Experience with an expert system for automated HAZOP analysis. Computers chem. Engng., Vol. 20, Suppl., pp. S1589–S1594.

Vaidhyanathan, R. and Venkatasubramanian, V. 1996b. A semi-quantitative reasoning methodology for filtering and ranking HAZOP results in HAZOPExpert. Reliability Engineering and System Safety, 53, pp. 185–203.

Vecchetti, A. and Leone, H. 1996. SERO: A knowledge-based system for HAZOP studies. In: Proc. AIChE Symp. #312 Intelligent Systems in Process Engineering. Pp. 287–290.

Venkatasubramanian, V. and Rich, S. H. 1988. An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis. Comput. Chem. Engng, Vol. 12, No. 9/10, pp. 903–921.

Venkatasubramanian, V. and Vaidhyanathan, R. 1994. A knowledge-based framework for automating HAZOP analysis. AIChE Journal, Vol. 40, No. 3, pp. 496–505.

Zerkani, H. and Rushton, A. G. 1992. Computer emulation of hazard identification. In: Interactions between process design and process control. International Federation of Automatic Control Workshop, J. D. Perkins (ed.), Pergamon, Oxford. pp. 221–226.

Appendices

1. Grammar for HAZOP expert rules
2. Example problems to be solved by the fluid model
3. Fluid model functionalities for solving the example problems
4. Compatibility matrix
5. Program listings of the autohazid fluid model functions
6. AutoHAZID comparative avaluation test cases
7. Autohazid reports for test case 1
8. Autohazid reports for test case 2
9. Trichloroethane storage system

Appendix 1: Grammar for HAZOP expert rules

```
hazop_rule              left_side '-' '>' right_side ;

left_side               '(' event ')' |
                        '(' event ')' left_side ;

right_side              '(' event ')' ;

event                   free_text_event |
                        variable_event |
                        input_event |
                        output_event |
                        call_event ;

free_text-event         "" string_expression "" |
                        "" string_expression "" "" string_expression "" |
                        "" string_expression "" "" string_expression "" ""
string_expression "";

variable_eventvariable |
                        combined_variable relation_operator
                          combined_variable |
                        combined_variable "" string_expression "" |
                        "" string_expression "" combined_variable |
                        "" string_expression "" combined_variable ""
string_expression "" ;

relation_operator       '>' |
                        '=' |
                        '<' ;

input_event             'INPUT' flow deviation_expression ;

output_event            'OUTPUT' flow deviation_expression ;
```

```
call_event              kb_keyword goal parameters ;

deviation_expression    normal_deviation |
                        phase_deviation |
                        concentration_deviation |
                        extra_deviation ;

normal_deviation        'NO FLOW' |
                        'LOW FLOW' |
                        'HIGH FLOW' |
                        'REVERSE FLOW' |
                        'OTHER FLOW' |
                        'LOW TEMPERATURE' |
                        'HIGH TEMPERATURE' |
                        'LOW PRESSURE' |
                        'HIGH PRESSURE' |
                        'LOW LEVEL' |
                        'HIGH LEVEL' |
                        'LOW VISCOSITY' |
                        'HIGH VISCOSITY' |
                        'LOW pH' |
                        'HIGH pH' |
                        'HIGH SIGNAL' |
                        'LOW SIGNAL' ;

phase_deviation         'EXTRA PHASE'       phase_keyword |
                        'WRONG PHASE'       phase_keyword ;

concentration_deviation 'LOW CONCENTRATION'     flow_component |
                        'HIGH CONCENTRATION'    flow_component |
                        'MISSING SUBSTANCE'     flow_component ;

extra-deviation         'EXTRA FLOW'            matter |
                        'EXTRA SUBSTANCE'       matter ;
```

```
phase_keyword       'SOLID' |
                    'LIQUID' |
                    'GAS' ;

flow_component      combined_variable |
                    "" string_expression "" ;

matter              variable |
                    "" string_expression "" ;

flow                variable |
                    "" string_expression "" ;

kb_keyword          'SKB' |         /* substances */
                    'RKB' |         /* planned reactions */
                    'UWKB' ;        /* unwanted reactions */

goal                "" string_expression "" ;
                    /* should follow the hazard keywords in the called kb */

parameters          parameter |
                    parameter parameters ;

parameter           combined_variable |
                    "" string_expression "" ;
                    /* according to what parameters are needed by the
                       called kb */

combined_variable   variable |
                    point_expression;

point_expression    flow '.' 'subst';

variable            string ;

string_expression   string ;
```

Appendix 2: Example problems to be solved by the fluid model

Peat Power Plant

1. Fire of peat due to friction heat at the outlet transporter device
2. Dust explosion of peat dust
3. Fire of peat due to excess air when the manhole is opened or then extra holes are made to clear blockages
4. Peat cyclone erodes because of wrong material selection

LPG Storage System

5. Thermal expansion of LPG
6. LPG leakage and fire/explosion
7. Gasification of LPG
8. Unintended mixing of propane and butane

Black liquor system

9. Toxicity of black liquor
10. Burning / Water removal of weak water/liquor mixture
11. Formation of "soft soap cake" on the surface of liquid black liquor
12. "Water explosion" in black liquor burner
13. Problems in mixing tank lead to thickening of the mixture
14. Blockage due to burning of black liquor inside a heated pipeline

15. Explosion in "smoke gas hall" due to flames from the burner

16. Burning of strong water/liquor mixture

Propylene storage system

17. Thermal expansion of propylene

18. Propylene contaminated with water, freezing

19. Boiling of liquid propylene

20. Gasification of propylene inside pipeline

21. Pipe rupture due to low temperature

22. Explosion of propylene/air mixture

23. Hazards of mixing propylene with compressor cooling medium

24. Liquidation of propylene gas

25. Polymerization of propylene in 300 C

26. Decomposition of propylene in 400 C

27. Freezing of propylene

Cumene system

28. Toxicity of CHP

29. Gasification of CHP

30. Flammability of CHP

31. Decomposition of CHP, effect of contaminants

32. Decomposition of cumene

33. Formation of salts

34. Flammability of active carbon

35. Too low temperature for cumene oxidation

36. Extra input of alkalic chemical disturbs the oxidation reaction

37. Extra input of NaOH causes slow decomposition of CHP and formation of phenol

38. Cumene decomposes into sulphur acid due to high concentration of sulphur in cumene input, sulphur acid decomposes CHP into phenol which disturbs the oxidation reaction

39. Corrosion due to high concentration of sulphur in cumene flow

40. Oxidation reaction is disturbed by extra air

41. High concentration of CHP in cumene input disturbs the oxidation reaction

42. Extra flow of phenol or sulphur into reactor disturbs the oxidation reaction

43. Violent decomposition of CHP due to high temperature

44. Violent decomposition of CHP due to increase of concentration after reaction has been stopped

45. Catalysis is prevented due to the presence of alkalic compounds

46. Catalysis is stopped temporarily due to extra water

47. Catalysis is stopped temporarily due to extra glycol

48. Catalysis is prevented due to extra salts

49. Corrosion due to organic acids formed as side product in CHP decomposition into phenol

50. Erosion due to extra metals

51. Oxidation reaction is disturbed by extra metals

52. Flammability of air/CHP mixture

Appendix 3: Fluid model functionalities for solving the example problems

State of matter in given pressure or temperature

2: Is it solid in atmospheric conditions?

7: How high temperature is needed to turn liquid into gas in the given pressure? Is such a temperature increase possible?

How low pressure is needed to turn liquid into gas in the given temperature? Is such a pressure decrease possible?

11: How low temperature is needed to turn liquid into solid in the given pressure? Is such a temperature decrease possible?

How high pressure is needed to turn liquid into solid in the given temperature? Is such a pressure increase possible?

12. How high temperature is needed to turn liquid into vapour in the given pressure? Is the design temperature of the unit significantly above that? (see also Decomposition)

13: As 11, but for a mixture

18: As 11, but for a contaminant

19: As 7, but concerning the formation of vapour

20: As 7

24: How low temperature is needed to turn gas into liquid in the given pressure? Is such a temperature decrease possible?

How high pressure is needed to turn gas into liquid in the given temperature? Is such a pressure increase possible?

27: As 11

29: As 7

Flammability

1. Is a solid material flammable (look at oxygen content?) ? What is the flammability limit concerning water content? Is it possible that the material contains less water than that?

2. What is the flammability range of dust from a solid material? Is it possible that the dust concentration goes within that range?

3. As 2

6. What is the flash point of the fluid? Is the outside temperature above that?

10. As 1

15. What is the flammability range of a gas? Is it possible that the outside process concentration of the gas goes within that range at an indoors located part of the process? What is the flash point of the gas? Is the outside temperature or the surface temperature of process equipment above that?

16. As 1

22. As 15, but also concerning the inside process concentration of air

30. As 6 and 22

34. As 1

52. As 22

Compatibility

4. Is the processed fluid or solid material erosive? What is the sensitivity of the material to erosion?

8. What are the properties of a mixture in the given pressure: flash point, flammability range, boiling point?

23. Do the mixed fluids react? Is the reaction violent? Is there heat or gas generation? What are the reaction products and what are their properties, compatibility and hazards?

33. As 23

38. As 23, but see also reaction problems

39. Does the fluid corrode the materials of the process equipment?

49. As 39, but see also decomposition

50. As 4

Toxicity

9. Is the fluid toxic?

28. As 9

Decomposition

14. What is the decomposition temperature of a solid material? Is it possible to reach that temperature on the surface of process equipment? What are the properties and hazards of the decomposition products?

25. Does the fluid polymerise? What is the polymerisation temperature? Is such a temperature increase possible?

26. What is the decomposition temperature of a fluid? Is such a temperature increase possible? What are the properties, compatibility and hazards of the decomposition products?

31. As 26

32. As 26

49. as 26, but see also compatibility

Material limits

21. What is the lower temperature limit of the material? Is such a temperature decrease possible?

Reaction problems

35. What is the lower temperature limit for the reaction? Is such a temperature decrease possible? Do other reactions take place in that situation? What are the reaction products and their properties, compatibility and hazards?

36. What is the compatibility of the reaction with extra components? Do other reactions take place in that situation? What are the reaction products and their properties, compatibility and hazards?

37. As 36

38. As 36, but see also decomposition

40. What is the sensitivity of the reaction to concentration deviations? Does it lead to the violent acceleration of the reaction? Do other reactions take place in that situation? What are the reaction products and their properties, compatibility and hazards?

41. As 40

42. As 36

43. What is the higher temperature limit for the reaction? Is such a temperature increase possible? Does it lead to the violent acceleration of the reaction?

45. As 36

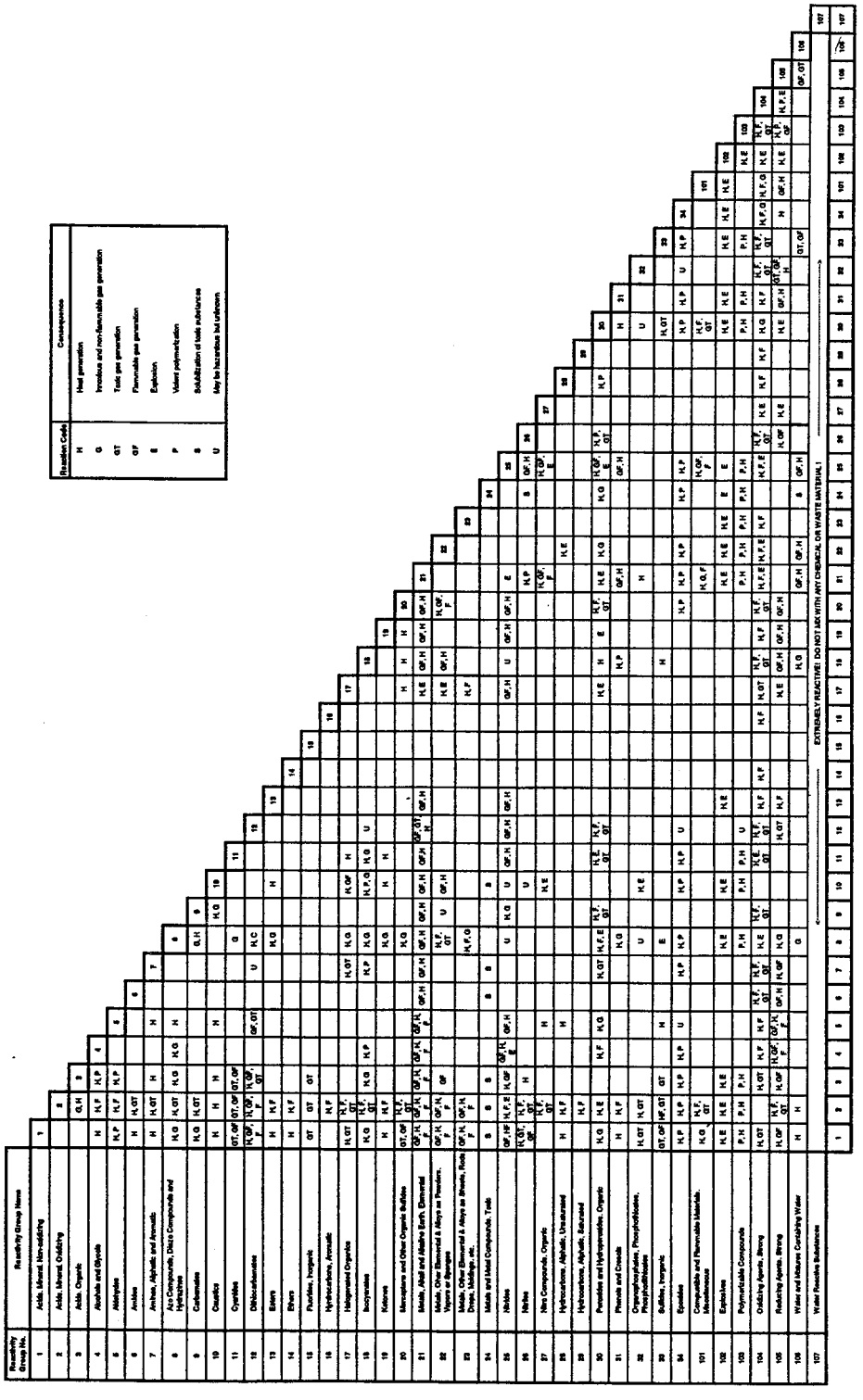
46. As 36

47. As 36

48. As 36

51. As 36

Appendix 4: Combatibility matrix



Appendix 5: Program listings of the autohazid fluid model functions

```
extern void checkFluidProperty(CondReply *crep, const Pred& P1,
                               const Location& L1)
{
   // This section checks every component in the list of components
for L1,
   // and if it cannot find a component in the fluid library, or if
the
   // property is non-false for any of the components, then the
   // function returns 1.

   for (int i = 0 ; i < L1.NoComps() ; i++)
   {
      Str N1 = L1.CompName(i);
      Fluid *fp1 = findFluid(N1);
      if (fp1 == NULL)
      { crep->set_reply(1);
         return;
      }

      // if got here fp1 is the fluid object of interest

      List *objList;
      objList = fp1->get_ruleObjects();

      if (!objList || objList->is_empty())
      {
          crep->set_reply(1);   /* if no rules are found, assume that
                                   the property holds */
          return;
      }

      /* Call the backward chaining procedure to resolve the query
         topEvent using the rules objList which are valid for the
         fluid fluidName */

      StrTree condTree;
      Str topEvent = P1.Name();

      /* add double quotes and parentheses to topEvent */
      char *dqTopEvent, *topPtr;
      dqTopEvent = (char *)malloc(strlen(topEvent) + 5);
      topPtr = dqTopEvent;
      *dqTopEvent = '(';
      dqTopEvent++;
      *dqTopEvent = '\"';
      dqTopEvent++;
      strcpy(dqTopEvent, topEvent);
      dqTopEvent = dqTopEvent + strlen(topEvent);
      *dqTopEvent = '\"';
      dqTopEvent++;
```

```
      *dqTopEvent = ')';
      dqTopEvent++;
      *dqTopEvent = '\0';

      /* call backward chaining of rules */
      int bc_ok = 0;
      bc_ok = backwardChain(&condTree, topPtr, objList);

      /* substitute variables
         NOTE! in case of multi-valued variables this should
         create multiple branches to the tree; an association list
         of variables and values should be used in the recursive
         variable substitution function to avoid using different
         value for the same variable within the same subtree */

      List var_list;
      List *var_list_p;
      var_list_p = &var_list;

      int sv_ok = 0;
      sv_ok = substituteVariables(&condTree, var_list_p, fp1, L1);

      /* evaluate the nodes */
      int en_ok = 0;
      en_ok = evaluateNodes(&condTree);

      /* return the answer */
      if (condTree.true()) crep->set_reply(1);
      else if (condTree.false()); /* leave the reply as it is */
      else crep->set_reply(2); /* error */
   }

return;
}
```

```
Fluid *findFluid(const Str& N1)
{
   // Scan the list of fluids known about by the program :
   for (Cell *c1 = fluidList.first() ; c1 ; c1 = fluidList.next(c1))
   {
      Fluid *fp1 = (Fluid*)(fluidList.content(c1));
      assert(fp1 != NULL);
      // Check for matching name :
      if (N1 == fp1->name())
         return fp1;
   }
   // If not found so far, return NULL :
   return NULL;
}
```

```
List* get_ruleObjects(){return &ruleObjects;}
```

```
int backwardChain(StrTree *condTree, char *topEvent, List *ruleList)
{
        FRule* rule_p_cvector[MAX_RULES_FOR_EVENT];
        int empty_node_flag;

        /* create tree node (topEvent) */

        StrTree *new_node_p;
        new_node_p = new StrTree();

        Str *eventStr;
        eventStr = new Str(topEvent);

        new_node_p->set_nodetext(*eventStr);

        new_node_p->set_gate_type(GTT_TERM);

        if (condTree->first_child() == NULL) /* no child nodes
                                                before */
        {
                condTree->set_first_child(new_node_p);
                new_node_p->set_parent(condTree);
                condTree->set_gate_type(GTT_OR);
        }
        else /* other child nodes exist */
        {
                new_node_p->set_next_sister(condTree->first_child());
                (condTree->first_child())
                         ->set_prev_sister(new_node_p);
                condTree->set_first_child(new_node_p);
                new_node_p->set_parent(condTree);
        }

        /* find rules (topEvent) */

        FRule *fr;
        fr = (FRule *)(ruleList->content(1));
        fr->findRules(&rule_p_vector[0], topEvent, ruleList);

        /* examine the rules, to prevent loops, do not consider such
           rules which are already in use within this branch of the
           recursion */

        int rule_ind = 0;
        while ( rule_p_vector[rule_ind] != NULL )
        {
           if (rule_p_vector[rule_ind]->bc_in_use() == EXPF_IN_USE)
               rule_ind++;
           else
           {
            rule_p_vector[rule_ind]->set_in_use(EXPF_IN_USE);

            /* if there are many rules, add an empty node to the tree
               to avoid mixing OR-statements with AND-statements;
               unnecessary empty nodes are removed before the end
               result is returned */
```

```
int left_length = (rule_p_vector[rule_ind]
                          ->left())->length();
if ((left_length  > 1)  && (rule_p_vector[1] != NULL))
    empty_node_flag = EMPTY_NODE;
else
    empty_node_flag = NO_EMPTY_NODE;

if (empty_node_flag == EMPTY_NODE)
{
    /* create a node */

    StrTree *empty_node_p;
    empty_node_p = new StrTree();

    empty_node_p->set_parent(new_node_p);

    /* use the empty node in place of the actual node */
    /* if there is already an empty node, make it a sister
       of the new empty node */

    empty_node_p->set_next_sister(new_node_p
                  ->first_child());
    if (empty_node_p->next_sister() != NULL)
    {
        (empty_node_p->next_sister())
         ->set_prev_sister(empty_node_p);
    }

    /* the parent of an empty node should always be an
       OR gate */
    new_node_p->set_gate_type(GTT_OR);

    new_node_p->set_first_child(empty_node_p);
    new_node_p = empty_node_p;

}

/* examine the left side statements of the current rule */

List* left_p = rule_p_vector[rule_ind]->left();
int and_flag = GTT_OR;
int left_ind = 1;

while (left_ind <= left_p->length())
{

     char *left_event;
     left_event = (char *)(left_p->content(left_ind));

     int bc_rec_ok;
     bc_rec_ok = backwardChain(new_node_p, left_event,
                 ruleList);
```

```
                /* if there is only one left side statement, this
                   makes the node an OR-gate */
                /* if there are more left-side statements, there will
                   be a next round of this loop */
                /* and the node will be made an AND-gate */
                new_node_p->set_gate_type(and_flag);
                and_flag = GTT_AND;
                left_ind++;
            }

            /* if this is an empty node, return to its parent node */
            if (empty_node_flag == EMPTY_NODE) new_node_p =
                                       new_node_p->parent();

            /* as we have just returned from the recursive branch,
               mark the rule unused again */
            rule_p_vector[rule_ind]->set_in_use(EXPF_NOT_IN_USE);

            /* go to next rule */
            rule_ind++;
           }
        }

        return 1;
}
```

```
void FRule::findRules(FRule** rule_p_vector_p, char* topEvent, List*
ruleList)
{
            /* initialize the pointer vector */

            int rv_ind;

            for (rv_ind = 0; rv_ind < MAX_RULES_FOR_EVENT; rv_ind++)
            {
                rule_p_vector_p[rv_ind] = NULL;
            }
            rv_ind = 0;

            /* loop through the rules */

            int rule_ind = 1;

            while (rule_ind <= ruleList->length())
            {
                      /* compare the top event to the right side of
                         the rule */
                      char *string1, *string2;

                      FRule *rule_obj_p;
                      rule_obj_p = (FRule *)(ruleList
                                              ->content(rule_ind));

                      List *right_p;
                      right_p = rule_obj_p->right();

                      char *stat_p;
                      stat_p = (char *)(right_p->content(1));
                      lowercase(stat_p);
                      lowercase(topEvent);

                      string1 = this->Remove_Blancs(stat_p);
                      string2 = this->Remove_Blancs(topEvent);

                      if (!strcmp(string1,string2)){
                          rule_p_vector_p[rv_ind] = rule_obj_p;
                          rv_ind++;
                      }

                      /* go to next rule */

                      rule_ind++;
            }

            return;

}
```

```
int substituteVariables(StrTree *condTree, List *var_list_p,
                        Fluid *flu, const Location& L1)
{

    char *text_p, *word, *original_event_text, *event_p, *new_event_p;
    char new_event[MAX_EVENT_LENGTH];
    int word_end, rest_start;

    /* allocate memory for string manipulation results */
    char *string1, *string2, *string3;
    string1 = (char *)malloc(MAX_EVENT_LENGTH);
    string2 = NULL;
    string3 = (char *)malloc(MAX_EVENT_LENGTH);

    /* go through the subtree */

    List value_list;
    int value_flag = TRUE;

    while (value_flag == TRUE)
    {
     if (condTree != NULL)
     {
        if (condTree->this_node() == "")
        {
         value_flag = FALSE;
         int sv_ok = 0;
         sv_ok = substituteVariables(condTree->first_child(),
                                     var_list_p, flu, L1);
         sv_ok = substituteVariables(condTree->next_sister(),
                                     var_list_p, flu, L1);
        }
        else
        {
         /* set a pointer to the beginning of event_text and start to
            extract variable names until the end of event_text */

         value_flag = FALSE;
         text_p = (condTree->this_node());
         text_p++; /* skip the leading parenthesis */
         while ((value_flag == FALSE) && (text_p != NULL)
                 && (*text_p != '\0') && (*text_p != ')'))
         {
            int skip_flag = FALSE;
            word = NULL;

            /* remove extra spaces and skip the operators <, >, = and
               !=, operator check added by PMH 2.7.1996 */
            while ((text_p != NULL) && (
                   (*text_p == ' ') || (*text_p == '<') ||
                   (*text_p == '>') || (*text_p == '=') ||
                   (*text_p == '!'))) text_p++;
            if ((text_p != NULL) && (*text_p == '\0'))
               skip_flag = TRUE;
```

```
/* if the next character is a double quote, move to the
   next double quote */
if ((skip_flag == FALSE) && (*text_p == '\"'))
{
 text_p++;
 text_p = strstr(text_p, "\"");
 if (text_p != NULL) text_p++;
 skip_flag = TRUE;
}

word_end = (int)strcspn(text_p, ") \0");

word = (char *)malloc(word_end+1);

strncpy(word, text_p, word_end);
word[word_end] = '\0';

/* when a variable name is found, retrieve values from the
   fluid object, convert the variable name
   to lower case */

if (skip_flag == FALSE)
{
   lowercase(word);
   double *d_value;
   d_value = (double *)malloc(sizeof(double));
   *d_value = -999.000;

   if (!strcmp(word, "cas"))
   {
      *d_value = flu->CasNum();
   }
   else if (!strcmp(word, "molwt"))
   {
      *d_value = flu->molWeight();
   }
   else if (!strcmp(word, "freezept"))
   {
      *d_value = flu->freezingPt();
   }
   else if (!strcmp(word, "decomptmp"))
   {
      *d_value = flu->decompTemp();
   }
   else if (!strcmp(word, "flashpt"))
   {
      *d_value = flu->flashPoint();
   }
   else if (!strcmp(word, "autoignitionpt"))
   {
      *d_value = flu->autoIgnitionPoint();
   }
   else if (!strcmp(word, "lowflamlim"))
   {
      *d_value = flu->lowerFlamLim();
   }
```

```
else if (!strcmp(word, "tox"))
{
   *d_value = flu->toxicity();
}
else if (!strcmp(word, "lat"))
{
   *d_value = flu->latHeat();
}
else if (!strcmp(word, "boilpt"))
{
   *d_value = flu->bpatamb();
}
else if (!strcmp(word, "phystate"))
/* this is a call to the properties package */
{
   /* default value 0 triggers the evaluation of
      convential rules for state of matter */

   *d_value = 0.0;

   #if _MSDOS
   int i_value, iErr;

   prepareProperty Call(pPrpSpec, PHYSTATE, L1);

   iErr = shz_calc_prp(pPrpSpec, pPrpRes);
   if (iErr) i_value = 0; /* default */
   else i_value = pPrpRes->Result1.iResult;
   *d_value = (double) i_value;
   #endif /* MSDOS */
}
else if (!strcmp(word, "pressure"))
/* unit pressure from the input parameter */
{
   *d_value = L1.Pressure();
}
else if (!strcmp(word, "temperature"))
/* unit temperature from the input parameter */
{
   *d_value = L1.Temperature();
}

if ((*d_value < -999.005) || (*d_value > -998.995))
/* in other words, d_value != -999.000 which means
   value found */
{
   value_list.append(d_value);
}

/* exit with no modifications if there are no values,
   and set the default truth value to true because
   there are variables with no values, PMH 2.7.1996 */

if (value_list.first() == NULL)
{
```

```
        condTree->set_true();
        skip_flag = TRUE;
   }
}

/* for the first value, modify event_text */

value_flag = FALSE;
if ((skip_flag == FALSE) && (value_list.first() != NULL))
{
 event_p = (condTree->this_node());
 new_event_p = &new_event[0];
 while ((*event_p != '\0') && (event_p != text_p))
 {
     *new_event_p = *event_p;
     new_event_p++;
     event_p++;
 }

 char value[MAX_DOUBLE_LEN];
 char *value_str;
 value_str = &value[0];
 int dec, sign;

 double *var_value;
 var_value = (double *)(value_list.content(1));
 value_str = ecvt( *var_value, MAX_DOUBLE_LEN,
                   &dec, &sign);
 if (sign != 0) /* negative */
 {
    *new_event_p = '-';
    new_event_p++;
 }
 if (dec > 0) /* part before decimal point */
 {
    strncpy(new_event_p, value_str, dec);
    new_event_p = new_event_p + dec;
    value_str = value_str + dec;
 }
 *new_event_p = '.'; /* the decimal point */
 new_event_p++;
 strcpy(new_event_p, value_str);
 new_event_p = new_event_p + strlen(value_str);
 rest_start = new_event_p - &new_event[0];
 event_p = text_p + word_end;
 while (*event_p != '\0')
 {
     *new_event_p = *event_p;
     new_event_p++;
     event_p++;
 }
 *new_event_p = '\0';
```

```
            /* store the original event_text for later use */

            if (condTree->this_node() == "")
                original_event_text = NULL;
            else
            {
               original_event_text =
                  (char *)malloc(strlen((condTree->this_node()))+1);
               strcpy(original_event_text, (condTree->this_node()));
            }

            condTree->set_nodetext(&new_event[0]);

            text_p = (condTree->this_node());
            text_p = text_p + rest_start;
            word_end = 0;

           }

           /* remove all old values from value_list */
           int vl_len;
           vl_len = value_list.length();
           value_list.drop(vl_len);

           /* these help the system to work correctly even if no
              values were found */
           if (word != NULL) free(word);
           if (word_end == 0) text_p++;
           else text_p = text_p + word_end;
       }

       int sv_ok = 0;
       if (value_flag == FALSE)
           sv_ok = substituteVariables(condTree->first_child(),
                                       var_list_p, flu, L1);

       /* go through the sisters, too */
       if ((value_flag == FALSE) &&
           (condTree->next_sister() != NULL))
       {
         sv_ok = substituteVariables(condTree->next_sister(),
                                     var_list_p, flu, L1);
       }
      }
     }
     else value_flag = FALSE; /* condTree == NULL */
    }

    free(string1);
    free(string3);
    if (string2 != NULL) free(string2);

    return 1;
}
```

```
void prepareProperty Call(S_PRP_SPEC *pPrpSpec, SHZ_PROP routine,
                          const Location& L1)
{
    pPrpSpec->PropName = routine;
    pPrpSpec->zMethod = "DEFAULT";
    pPrpSpec->fSpec1 = (float) L1.Temperature() + (float) 273.0;
                                                    // Convert to K
    pPrpSpec->fSpec2 = (float) L1.Pressure();

    if (routine == TB)
    {
       pPrpSpec->fSpec1 = (float) L1.Pressure();
       pPrpSpec->fSpec2 = 0.0;
    }
    else if (routine == TFREEZE)
    {
       pPrpSpec->fSpec1 = 0.0;
       pPrpSpec->fSpec2 = 0.0;
    }

    pPrpSpec->iNComps = L1.NoComps();

    // Read the list of components and component fractions from L1

    for (int i = 0; i<L1.NoComps(); ++i)
    {
        pPrpSpec->faCompFrac[i] = (float) L1.CompValue(i);
        // The calls require the indexes of the component names
        // -1 is returned if the component specified for L1 is
        // not found in the global list
        // This means that non-existing components will be indicated
        // by 0 in iaCompList
        pPrpSpec->iaCompList[i] =
                     globalCompList.Find(L1.CompName(i)) +1;
    }

}
```

```
int evaluateNodes(StrTree *condTree)
{
    char *char_p, *left_p, *left_end_p, *string1, *right_p,
         *right_end_p, *string2;
    char *dec_p, *cmp_str;
    int alloc_size, dl1, dl2, cmp_alloc, declen;

    StrTree *child_p;
    child_p = condTree->first_child();
    while (child_p != NULL)
    {
       int en_ok;
       en_ok = evaluateNodes(child_p);
       child_p = child_p->next_sister();
    }

    /* if the node is a terminal node, evaluate it,
       but if the default value is already true there is no need to
       evaluate because that means that there are variables with no
       value within nodetext,        PMH 2.7.1996
       the function intforcmp and related preparative code added by
       PMH 31.7.1996 */
    if ((condTree->first_child() == NULL) && (condTree->false()))
    {
    char_p = strstr((condTree->this_node()), "!=");
    if (char_p != NULL)
    {
        left_p = (condTree->this_node());
        while (((*left_p == '(') || (*left_p == ' ') ||
                (*left_p == '\"'))
                && (*left_p != '\0')) left_p++;
        left_end_p = char_p;
        left_end_p--;
        while ((*left_end_p == ' ') || (*left_end_p == '\"'))
               left_end_p--;
        left_end_p++;
        alloc_size = strlen(left_p) - strlen(left_end_p) + 1;
        if (alloc_size < 1) alloc_size = 1;
        string1 = (char *)malloc(alloc_size);
        strncpy(string1, left_p, alloc_size - 1);
        string1[alloc_size - 1] = '\0';
        right_p = &(char_p[2]);
        while (((*right_p == ' ') || (*right_p == '\"')) &&
                (*right_p != '\0')) right_p++;
        right_end_p = &char_p[2];
        while (*right_end_p != '\0') right_end_p++;
        right_end_p--;
        while ((*right_end_p == ' ') || (*right_end_p == '\"')
               || (*right_end_p == ')')) right_end_p--;
        right_end_p++;
        alloc_size = strlen(right_p) - strlen(right_end_p) + 1;
        if (alloc_size < 1) alloc_size = 1;
        string2 = (char *)malloc(alloc_size);
        strncpy(string2, right_p, alloc_size - 1);
        string2[alloc_size - 1] = '\0';
```

```
    dec_p = strstr(string1, ".");
    if (dec_p == NULL) dl1 = 0;
    else dl1 = strlen(dec_p) - 1;

    dec_p = strstr(string2, ".");
    if (dec_p == NULL) dl2 = 0;
        else dl2 = strlen(dec_p) - 1;

    if (dl1 > dl2)
    {
      cmp_alloc = strlen(string2) + dl1 + 1;
      cmp_str = (char *)malloc(cmp_alloc);
      strcpy(cmp_str, string2);
      free(string2);
      string2 = cmp_str;
      declen = dl1;
    }

        if (dl2 > dl1)
    {
      cmp_alloc = strlen(string1) + dl2 + 1;
      cmp_str = (char *)malloc(cmp_alloc);
      strcpy(cmp_str, string1);
      free(string1);
      string1 = cmp_str;
      declen = dl2;
    }

    intforcomp(string1, declen);
    intforcomp(string2, declen);

    if (!strcmp(string1, string2)) condTree->set_false();
    else condTree->set_true();

    free(string1);
    free(string2);
}
else
{
  char_p = strstr((condTree->this_node()), "=");
  if (char_p != NULL)
  {
       left_p = (condTree->this_node());
       while (((*left_p == '(') || (*left_p == ' ')
              || (*left_p == '\"'))
             && (*left_p != '\0')) left_p++;
       left_end_p = char_p;
       left_end_p--;
       while ((*left_end_p == ' ') || (*left_end_p == '\"'))
             left_end_p--;
       left_end_p++;
       alloc_size = strlen(left_p) - strlen(left_end_p) + 1;
       if (alloc_size < 1) alloc_size = 1;
       string1 = (char *)malloc(alloc_size);
```

```
      strncpy(string1, left_p, alloc_size - 1);
      string1[alloc_size - 1] = '\0';
      right_p = &(char_p[1]);
      while (((*right_p == ' ') || (*right_p == '\"'))
             && (*right_p != '\0')) right_p++;
      right_end_p = &char_p[1];
      while (*right_end_p != '\0') right_end_p++;
      right_end_p--;
      while ((*right_end_p == ' ') || (*right_end_p == '\"')
             || (*right_end_p == ')')) right_end_p--;
      right_end_p++;
      alloc_size = strlen(right_p) - strlen(right_end_p) + 1;
      if (alloc_size < 1) alloc_size = 1;
      string2 = (char *)malloc(alloc_size);
      strncpy(string2, right_p, alloc_size - 1);
      string2[alloc_size - 1] = '\0';

      dec_p = strstr(string1, ".");
      if (dec_p == NULL) dl1 = 0;
      else dl1 = strlen(dec_p) - 1;

      dec_p = strstr(string2, ".");
      if (dec_p == NULL) dl2 = 0;
          else dl2 = strlen(dec_p) - 1;

      if (dl1 > dl2)
      {
        cmp_alloc = strlen(string2) + dl1 + 1;
        cmp_str = (char *)malloc(cmp_alloc);
        strcpy(cmp_str, string2);
        free(string2);
        string2 = cmp_str;
        declen = dl1;
      }

      if (dl2 > dl1)
      {
        cmp_alloc = strlen(string1) + dl2 + 1;
        cmp_str = (char *)malloc(cmp_alloc);
        strcpy(cmp_str, string1);
        free(string1);
        string1 = cmp_str;
        declen = dl2;
      }

      intforcomp(string1, declen);
      intforcomp(string2, declen);


      if (!strcmp(string1, string2)) condTree->set_true();
      else condTree->set_false();

      free(string1);
      free(string2);
}
```

```
else
{
 char_p = strstr((condTree->this_node()), "<");
 if (char_p != NULL)
 {
  left_p = (condTree->this_node());
  while (((*left_p == '(') || (*left_p == ' ')
         || (*left_p == '\"'))
         && (*left_p != '\0')) left_p++;
  left_end_p = char_p;
  left_end_p--;
  while ((*left_end_p == ' ') || (*left_end_p == '\"'))
        left_end_p--;
  left_end_p++;
  alloc_size = strlen(left_p) - strlen(left_end_p) + 1;
  if (alloc_size < 1) alloc_size = 1;
  string1 = (char *)malloc(alloc_size);
  strncpy(string1, left_p, alloc_size - 1);
  string1[alloc_size - 1] = '\0';
  right_p = &(char_p[1]);
  while (((*right_p == ' ') || (*right_p == '\"'))
         && (*right_p != '\0')) right_p++;
  right_end_p = &char_p[1];
  while (*right_end_p != '\0') right_end_p++;
  right_end_p--;
  while ((*right_end_p == ' ') || (*right_end_p == '\"')
         || (*right_end_p == ')')) right_end_p--;
  right_end_p++;
  alloc_size = strlen(right_p) - strlen(right_end_p) + 1;
  if (alloc_size < 1) alloc_size = 1;
  string2 = (char *)malloc(alloc_size);
  strncpy(string2, right_p, alloc_size - 1);
  string2[alloc_size - 1] = '\0';

  dec_p = strstr(string1, ".");
  if (dec_p == NULL) dl1 = 0;
  else dl1 = strlen(dec_p) - 1;

  dec_p = strstr(string2, ".");
  if (dec_p == NULL) dl2 = 0;
      else dl2 = strlen(dec_p) - 1;

  if (dl1 > dl2)
  {
    cmp_alloc = strlen(string2) + dl1 + 1;
    cmp_str = (char *)malloc(cmp_alloc);
    strcpy(cmp_str, string2);
    free(string2);
    string2 = cmp_str;
    declen = dl1;
  }
  else if (dl2 > dl1)
  {
    cmp_alloc = strlen(string1) + dl2 + 1;
    cmp_str = (char *)malloc(cmp_alloc);
```

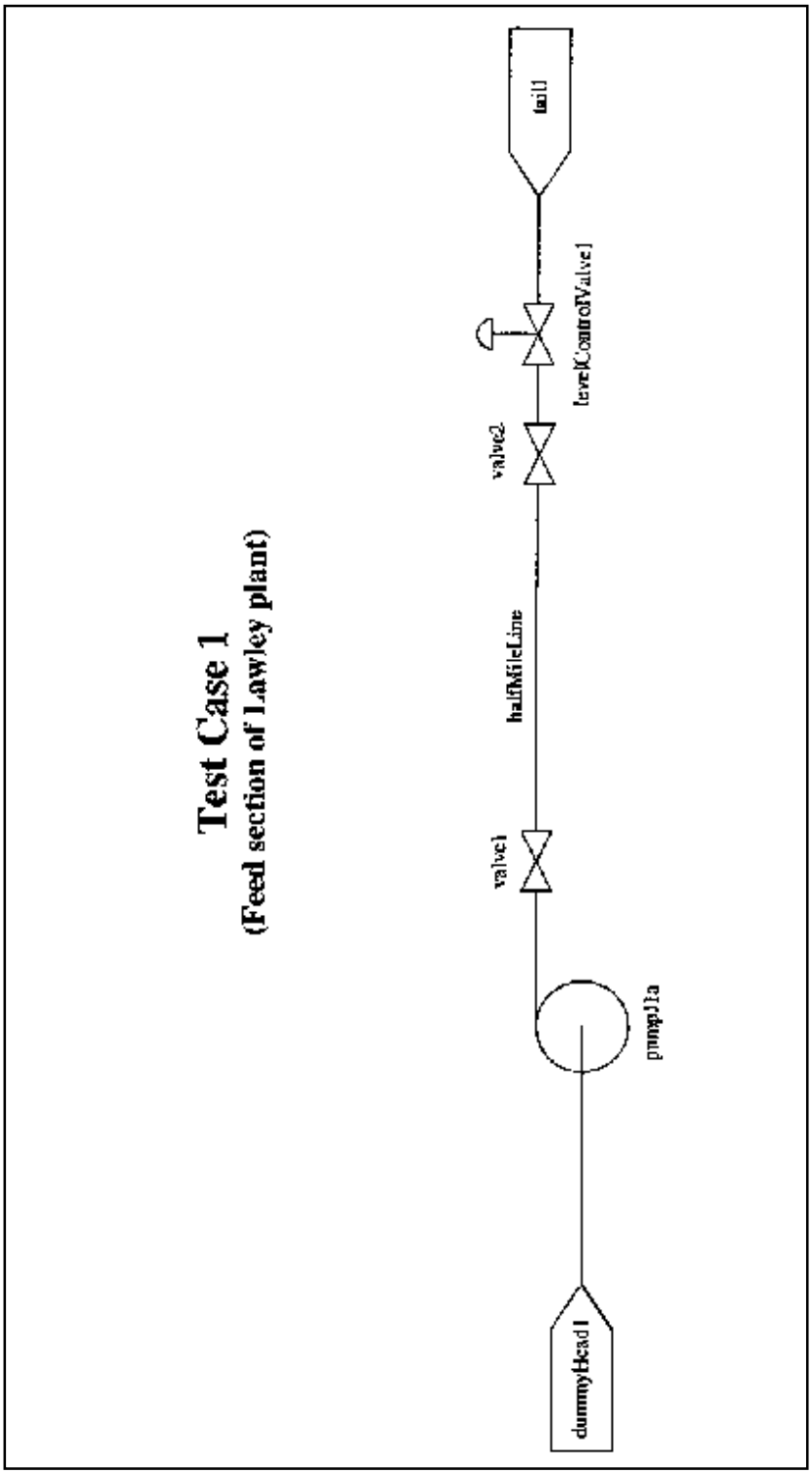
```
  strcpy(cmp_str, string1);
  free(string1);
  string1 = cmp_str;
  declen = dl2;
 }
 else /* dl1 == dl2 */
  declen = dl1;

 intforcomp(string1, declen);
 intforcomp(string2, declen);

 if (*string1 == '-')
 {
 if (*string2 == '-')
 {
     if (strlen(string1) > strlen(string2))
        condTree->set_true();
     else
     {
        if (strlen(string1) < strlen(string2))
           condTree->set_false();
        else
        {
         if (*string1 > *string2) condTree->set_true();
         else condTree->set_false();
        }
     }
 }
 else condTree->set_true();
 }
 else
 {
 if (*string2 == '-') condTree->set_false();
 else
 {
     if (strlen(string1) < strlen(string2))
        condTree->set_true();
     else
     {
        if (strlen(string1) > strlen(string2))
           condTree->set_false();
        else
        {
         if (*string1 < *string2) condTree->set_true();
         else condTree->set_false();
        }
     }
 }
 }

 free(string1);
 free(string2);
}
else
{
```

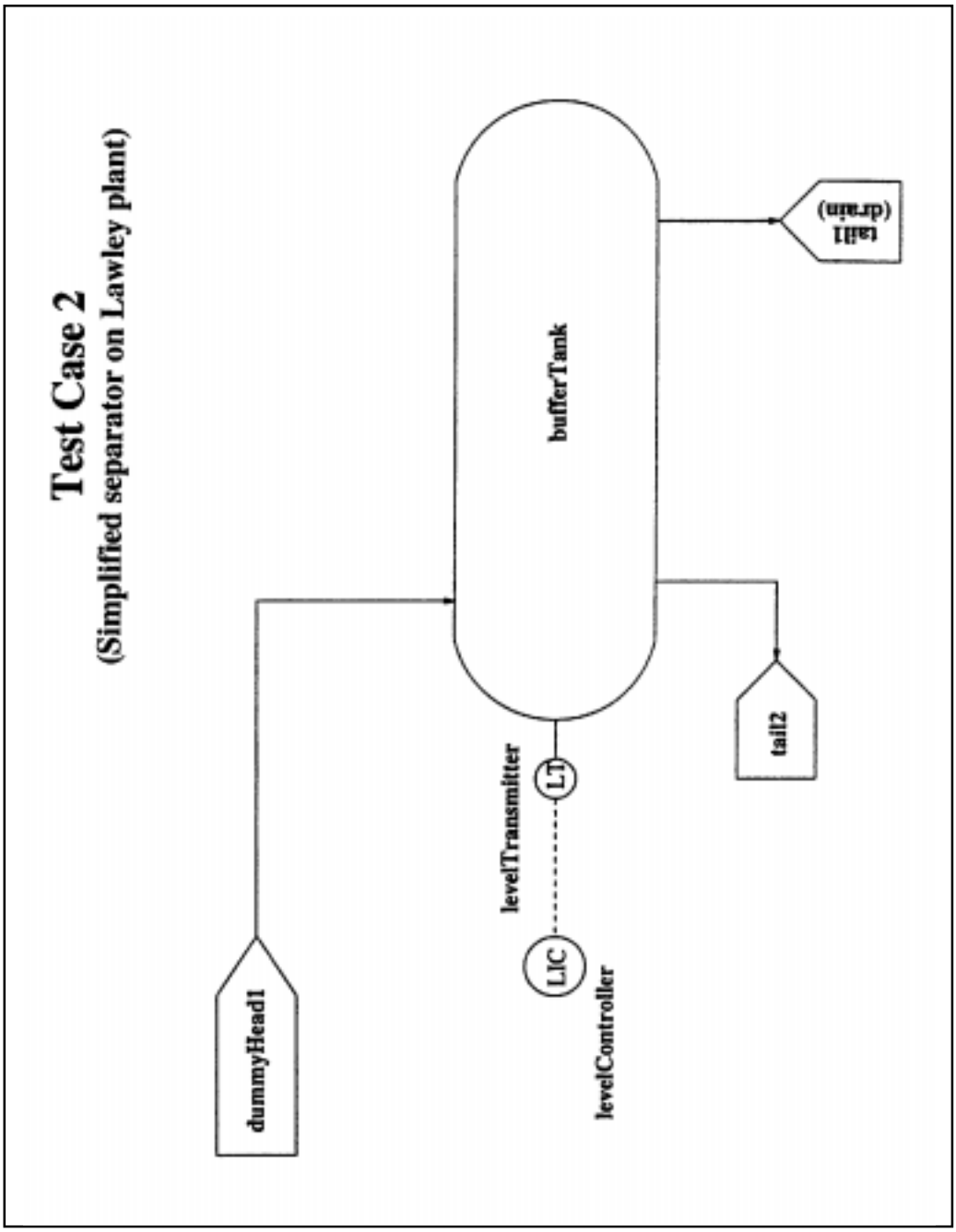
```
char_p = strstr((condTree->this_node()), ">");
if (char_p != NULL)
{
 left_p = (condTree->this_node());
 while (((*left_p == '(') || (*left_p == ' ')
        || (*left_p == '\"'))
        && (*left_p != '\0')) left_p++;
 left_end_p = char_p;
 left_end_p--;
 while ((*left_end_p == ' ') || (*left_end_p == '\"'))
       left_end_p--;
 left_end_p++;
 alloc_size = strlen(left_p) - strlen(left_end_p) + 1;
 if (alloc_size < 1) alloc_size = 1;
 string1 = (char *)malloc(alloc_size);
 strncpy(string1, left_p, alloc_size - 1);
 string1[alloc_size - 1] = '\0';
 right_p = &(char_p[1]);
 while (((*right_p == ' ') || (*right_p == '\"'))
        && (*right_p != '\0')) right_p++;
 right_end_p = &char_p[1];
 while (*right_end_p != '\0') right_end_p++;
 right_end_p--;
 while ((*right_end_p == ' ') || (*right_end_p == '\"')
        || (*right_end_p == ')')) right_end_p--;
 right_end_p++;
 alloc_size = strlen(right_p) - strlen(right_end_p) + 1;
 if (alloc_size < 1) alloc_size = 1;
 string2 = (char *)malloc(alloc_size);
 strncpy(string2, right_p, alloc_size - 1);
 string2[alloc_size - 1] = '\0';


 dec_p = strstr(string1, ".");
 if (dec_p == NULL) dl1 = 0;
 else dl1 = strlen(dec_p) - 1;

 dec_p = strstr(string2, ".");
 if (dec_p == NULL) dl2 = 0;
 else dl2 = strlen(dec_p) - 1;

 if (dl1 > dl2)
 {
   cmp_alloc = strlen(string2) + dl1 + 1;
   cmp_str = (char *)malloc(cmp_alloc);
   strcpy(cmp_str, string2);
   free(string2);
   string2 = cmp_str;
   declen = dl1;
 }
```

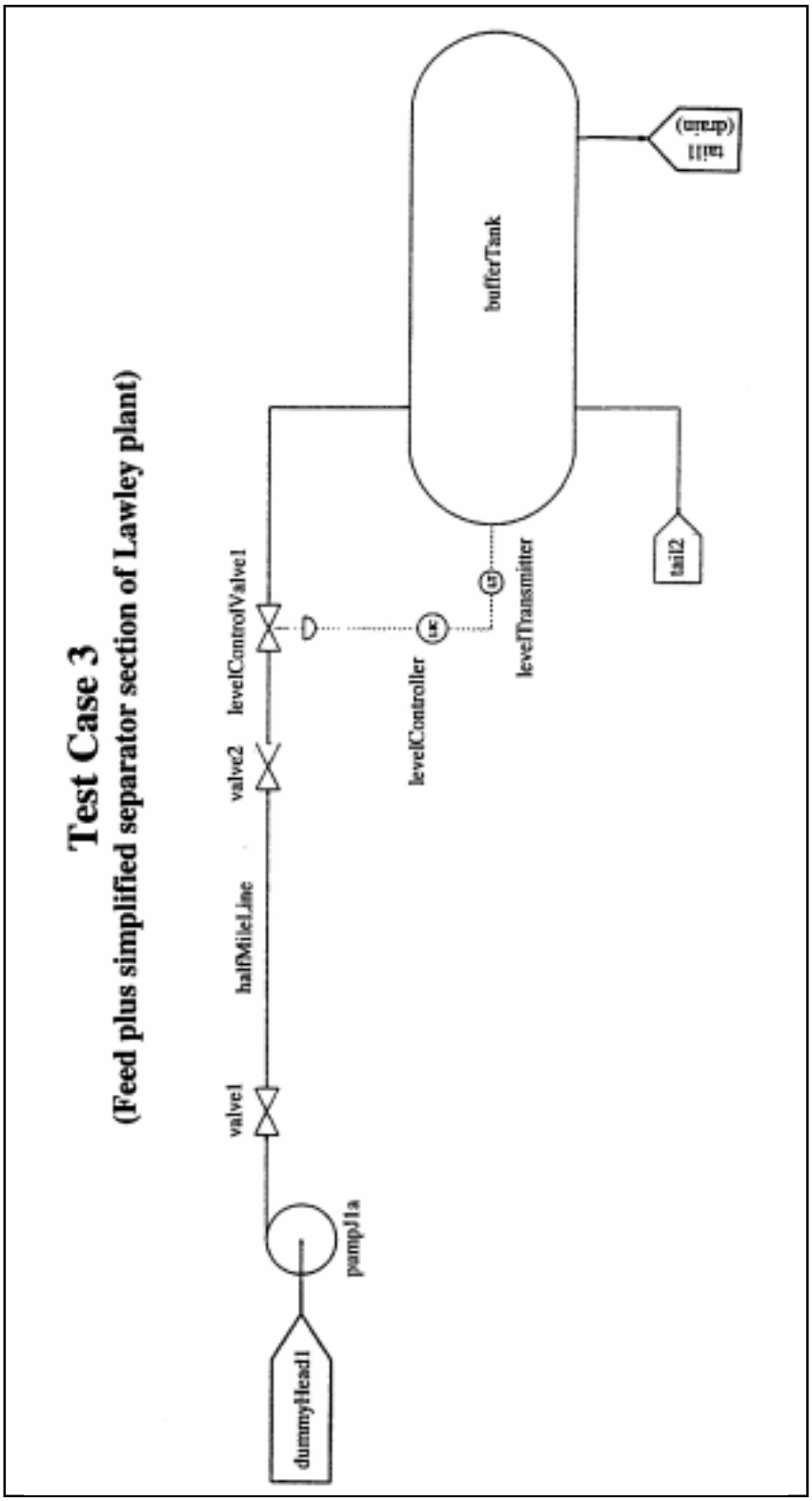
```
else if (dl2 > dl1)
{
  cmp_alloc = strlen(string1) + dl2 + 1;
  cmp_str = (char *)malloc(cmp_alloc);
  strcpy(cmp_str, string1);
  free(string1);
  string1 = cmp_str;
  declen = dl2;
}
else /* dl1 == dl2 */
  declen = dl1;

intforcomp(string1, declen);
intforcomp(string2, declen);

if (*string1 == '-')
{
if (*string2 == '-')
{
   if (strlen(string1) > strlen(string2))
      condTree->set_false();
   else
   {
      if (strlen(string1) < strlen(string2))
         condTree->set_true();
      else
      {
       if (*string1 < *string2) condTree->set_true();
       else condTree->set_false();
      }
   }
}
else condTree->set_false();
}
else
{
if (*string2 == '-') condTree->set_true();
else
 {
   if (strlen(string1) < strlen(string2))
      condTree->set_false();
   else
   {
      if (strlen(string1) > strlen(string2))
         condTree->set_true();
      else
      {
       if (*string1 > *string2) condTree->set_true();
       else condTree->set_false();
      }
   }
 }
}
```

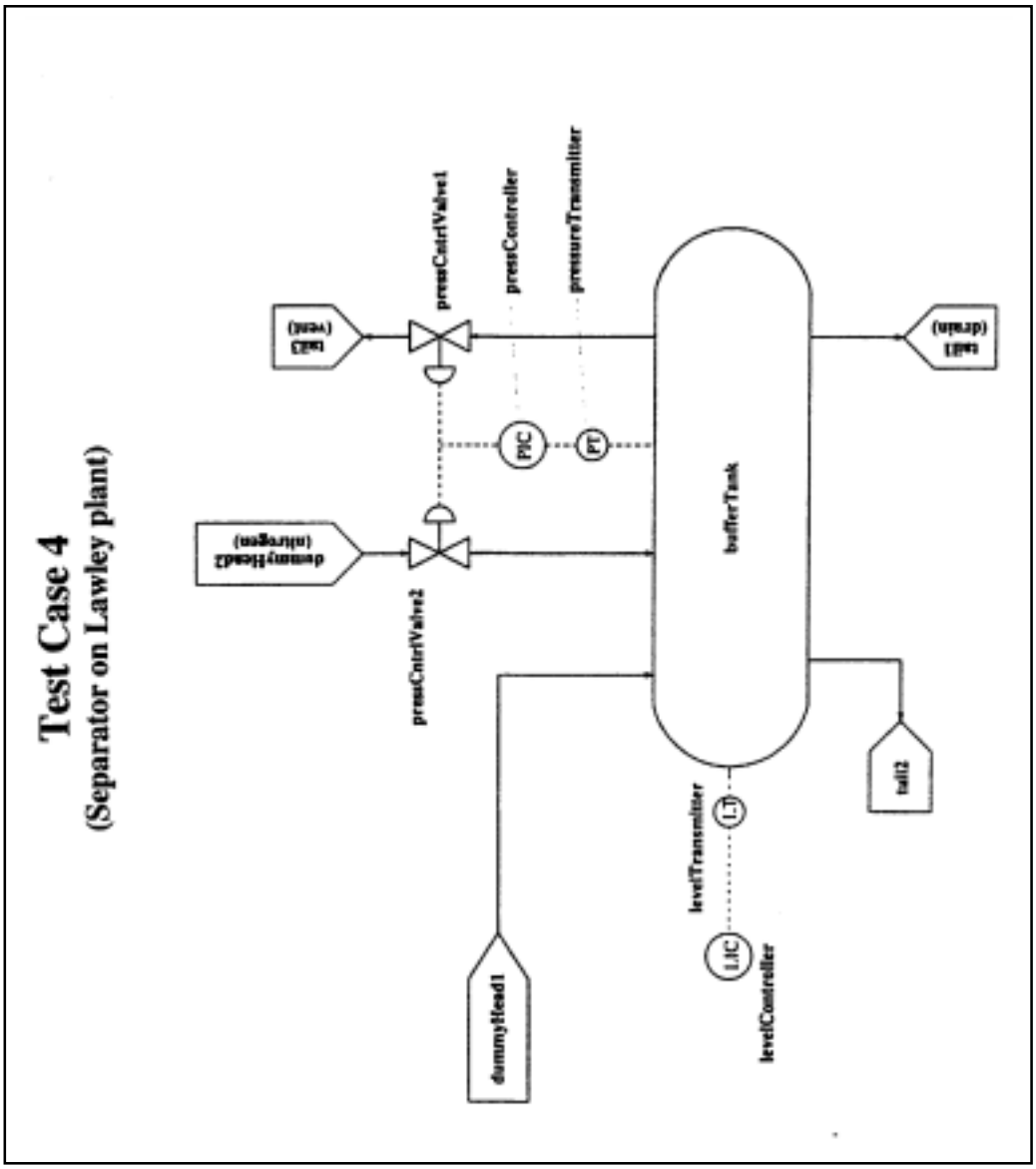
```
          free(string1);
          free(string2);
         }
         else condTree->set_true(); /* the terminal node does not
                    contain operators; Default to true, PMH 2.7.1996 */

        }

       }

     }
   }

   /* if the node is not a terminal node, evaluate it based on the
      childrens truth values */

   if (condTree->gate_type() == GTT_OR)
   {
      condTree->set_false();
      child_p = condTree->first_child();
      while (child_p != NULL)
      {
         if (child_p->true()) condTree->set_true();
         child_p = child_p->next_sister();
      }
   }
   if (condTree->gate_type() == GTT_AND)
   {
      condTree->set_true();
      child_p = condTree->first_child();
      while (child_p != NULL)
      {
         if (child_p->false()) condTree->set_false();
         child_p = child_p->next_sister();
      }
   }

    return 1;
}
```

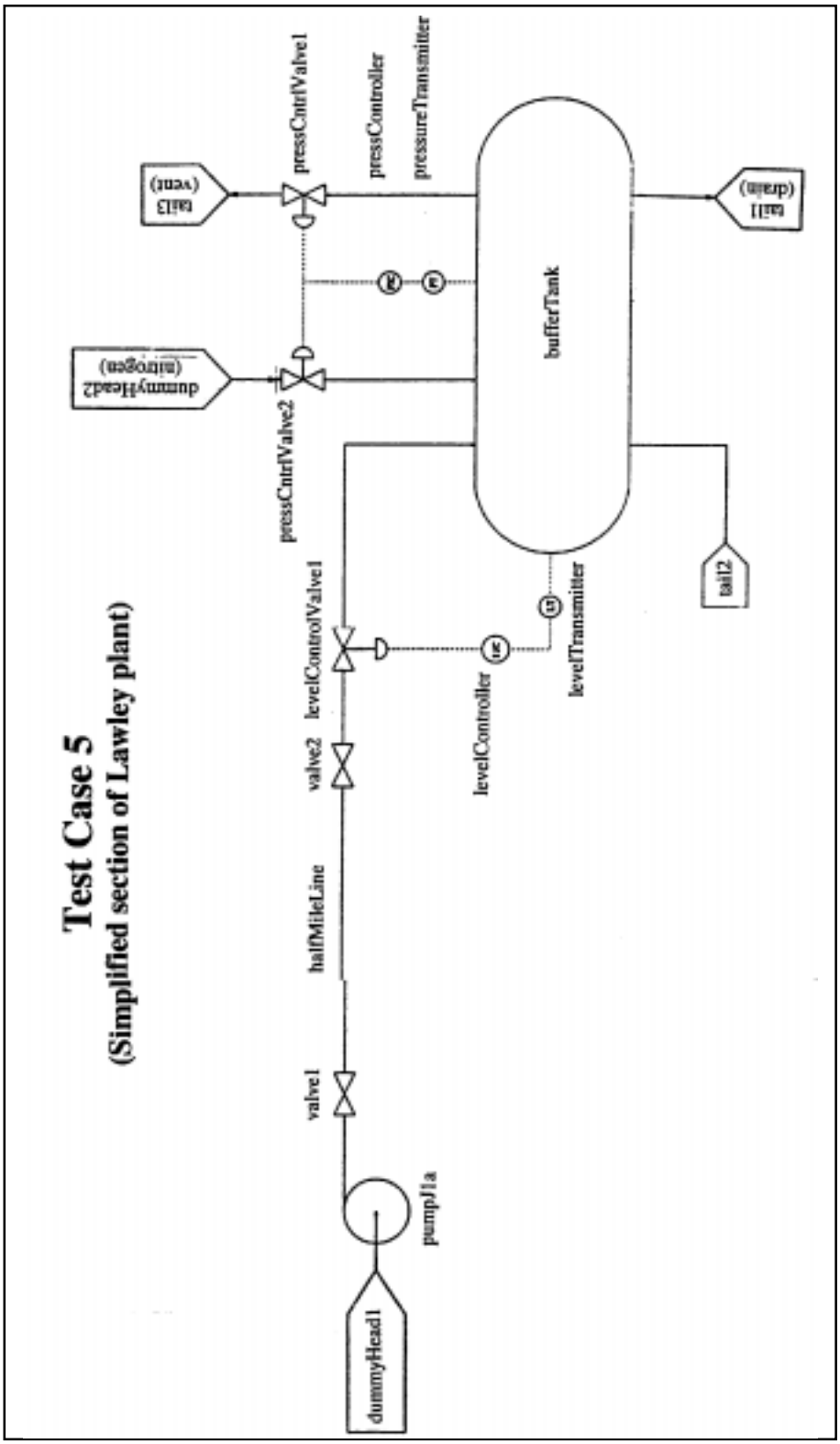
Appendix 6: AutoHAZID comparative evaluation test cases




Test Case 2
(Simplified separator on Lawley plant)
dummyHead1
LIC
levelController
levelTransmitter
LT
bufferTank
tail2
tail1
(drain)


Test Case 3
(Feed plus simplified separator section of Lawley plant)
dummyHead1
pump1a
valve1
halfMileLine
valve2
levelControlValve1
levelController
levelTransmitter
bufferTank
tail2
tail
(drain)


Test Case 4
(Separator on Lawley plant)
dummyHead1
dummyHead2 (nitrogen)
pressCntrlValve1
pressCntrlValve2
pressController
pressureTransmitter
levelTransmitter
levelController
bufferTank
tail1 (drain)
tail2
PIC
PT
LT
LIC


Test Case 5
(Simplified section of Lawley plant)
dummyHead1
pumpJ1a
valve1
halfMileLine
valve2
levelControlValve1
levelController
levelTransmitter
pressCntrlValve2
dummyHead2 (nitrogen)
pressCntrlValve1
pressController
pressureTransmitter
bufferTank
tail1 (drain)
tail2
tail3 (vent)

Appendix 7: Autohazid reports for test case 1

No fluid model

```
Report for FULL PLANT HAZOP.
HAZOP started at        Thu Feb 12 10:32:57 1998
HAZOP completed at      Thu Feb 12 10:34:40 1998

-----------------------------------------------------


Library Used            :           library2
Plant Used              :           plants\test1.pl
Results File            :           results\test1.no
Templates File          :           tlib
Fluids File             :           fluidlib


Flag Settings Used

display faults with no consequences             NO
display consequences with no causes             NO
filter out repeat faults                        YES
filter out repeat fault-conseq pairs            YES
display protections present                     NO
only display faults with no protections         YES
consequence rank threshold set at               1


-----------------------------------------------------
```

DEVIATION	CAUSE	CONSEQUENCE
pumpJ1 lessFlow in	pumpJ1 air ingress into pump	possible internal explosive atmosphere 4
	dummyHead1 leak to environment	toxic release 2, fire/explosion risk 2, fire or explosion risk 2, non-hazardous release 1
pumpJ1 moreFlow in	pumpJ1 leak to environment	fire/explosion risk 2, toxic release 2, non-hazardous release 1
pumpJ1 noFlow in	dummyHead1 no flow upstream, dummyHead1 blockage - solids, dummyHead1 blockage - frozen fluid	dry running - possible pump rupture 2
pumpJ1 revFlow in	dummyHead1 low pressure upstream, pumpJ1 loss of drive, tail1 high pressure downstream, pumpJ1 incorrect pump setup/installation	seal failure due to reverse impeller rotation 2, possible suction piping overpressure 2

pumpJ1 lessPressure in	dummyHead1 blockage - solids, pumpJ1 leak to environment, dummyHead1 blockage - frozen fluid, dummyHead1 leak to environment, dummyHead1 low pressure upstream	cavitation due to evolution of gases 3, cavitation - possible mechanical damage 3
pumpJ1 lessTemp in	dummyHead1 low temp upstream	seal failure - freezing of seal fluids 2, brittle fracture 2
pumpJ1 moreGas in	pumpJ1 air ingress into pump	vapour lock 3, pump damage - increased vibration 3, bearing overheat - loss of lubrication 3
pumpJ1 moreVapour in	dummyHead1 high temp upstream, pumpJ1 external fire	cavitation - possible mechanical damage 3
pumpJ1 moreFlow out	dummyHead1 high pressure upstream, pumpJ1 pressure surge at startup or shut-down, pumpJ1 lessPressure out	possible motor overload or trip 3
pumpJ1 noFlow out	levelControlValve fails closed, tail1 blockage - frozen fluid, valve1 closed, valve1 blockage - frozen fluid, levelControlValve blockage - solids, valve1 blockage - solids, valve3 blockage - solids, halfMileLine blockage - frozen fluid, levelControlValve blockage - frozen fluid, halfMileLine blockage - solids, tail1 complete blockage downstream, valve3 closed, tail1 blockage - solids, valve3 blockage - frozen fluid	possible pump casing overtemperature 2, possible seal overtemperature 2
pumpJ1 lessPressure out	tail1 leak to environment, valve1 leak to environment, levelControlValve leak to environment, halfMileLine leak to environment, valve3 leak to environment	toxic release 2, fire/explosion risk 2, non-hazardous release 1
pumpJ1 morePressure out	levelControlValve loss of control, valve3 partly closed, dummyHead1 high pressure upstream, tail1 blockage - frozen fluid, valve1 blockage - frozen fluid, levelControlValve blockage - frozen fluid, valve1 blockage - solids, valve3 blockage - frozen fluid, valve1 partly closed, levelControlValve blockage - solids, halfMileLine blockage - frozen fluid, pumpJ1 pressure surge at startup or shut-down,	possible pump casing or delivery pipework overpressure 2, possible seal overpressure 2

	tail1 high pressure downstream, halfMileLine blockage - solids, tail1 blockage - solids, valve3 blockage - solids	
pumpJ1 moreTemp out	dummyHead1 high temp upstream, pumpJ1 lessFlow out, pumpJ1 external fire	possible fluid decomposition 3, possible seal overtemperature 2, possible pump casing overtemperature 2
pumpJ1 maintenance	pumpJ1 no drains available	inadequate isolation and drainage 2
pumpJ1 startup	pumpJ1 no pressure sensor on pump delivery	cannot monitor pressure development at start-up 4
halfMileLine morePressure in	halfMileLine unit can be locked in	potential for liquid lock in and damage to unit by thermal expansion 3
halfMileLine moreTemp out	dummyHead1 high temp upstream, halfMileLine hot weather, pumpJ1 external fire, halfMileLine external fire, pumpJ1 lessFlow out	design temp exceeded 2
halfMileLine maintenance	halfMileLine no drains available	inadequate isolation and drainage 2
dummyHead1 revFlow out	pumpJ1 incorrect pump setup/installation, dummyHead1 low pressure upstream, tail1 high pressure downstream, pumpJ1 loss of drive	possible upstream contamination 3
tail1 lessTemp in	levelControlValve flashing across valve	possible valve damage 3

With limited fluid model

```
Report for FULL PLANT HAZOP.
HAZOP started at        Thu Feb 12 10:43:01 1998
HAZOP completed at      Thu Feb 12 10:44:34 1998

-----------------------------------------------------


Library Used:           library2
Plant Used  :           plants\test1.pl
Results File:           results\test1.unc
Templates File          :           tlib
Fluids File :           fluidlib


Flag Settings Used

display faults with no consequences             NO
display consequences with no causes             NO
filter out repeat faults                        YES
filter out repeat fault-conseq pairs            YES
display protections present                     NO
only display faults with no protections         YES
consequence rank threshold set at               1


-----------------------------------------------------
```

DEVIATION	CAUSE	CONSEQUENCE
pumpJ1 lessFlow in	pumpJ1 air ingress into pump	possible internal explosive atmosphere 4
	dummyHead1 leak to environment	fire or explosion risk 2, fire/explosion risk 2
pumpJ1 moreFlow in	pumpJ1 leak to environment	fire/explosion risk 2
pumpJ1 noFlow in	dummyHead1 no flow upstream	dry running - possible pump rupture 2
pumpJ1 revFlow in	dummyHead1 low pressure upstream, pumpJ1 loss of drive, tail1 high pressure downstream, pumpJ1 incorrect pump setup/installation	seal failure due to reverse impeller rotation 2, possible suction piping overpressure 2
pumpJ1 lessPressure in	dummyHead1 low pressure upstream, pumpJ1 leak to environment, dummyHead1 leak to environment	cavitation - possible mechanical damage 3
pumpJ1 lessTemp in	dummyHead1 low temp upstream	seal failure - freezing of seal fluids 2

pumpJ1 moreGas in	pumpJ1 air ingress into pump	vapour lock 3, pump damage - increased vibration 3, bearing overheat - loss of lubrication 3
pumpJ1 moreVapour in	dummyHead1 high temp upstream, pumpJ1 external fire	cavitation - possible mechanical damage 3
pumpJ1 moreFlow out	dummyHead1 high pressure upstream, pumpJ1 pressure surge at startup or shut-down, pumpJ1 lessPressure out	possible motor overload or trip 3
pumpJ1 noFlow out	levelControlValve fails closed, valve1 closed, tail1 complete blockage downstream, valve3 closed	possible pump casing overtemperature 2, possible seal overtemperature 2
pumpJ1 lessPressure out	tail1 leak to environment, valve1 leak to environment, levelControlValve leak to environment, halfMileLine leak to environment, valve3 leak to environment	fire/explosion risk 2
pumpJ1 morePressure out	levelControlValve loss of control, valve3 partly closed, dummyHead1 high pressure upstream, pumpJ1 pressure surge at startup or shut-down, tail1 high pressure downstream, valve1 partly closed	possible pump casing or delivery pipework overpressure 2, possible seal overpressure 2
pumpJ1 moreTemp out	dummyHead1 high temp upstream, pumpJ1 lessFlow out, pumpJ1 external fire	possible pump casing overtemperature 2, possible seal overtemperature 2
pumpJ1 maintenance	pumpJ1 no drains available	inadequate isolation and drainage 2
pumpJ1 startup	pumpJ1 no pressure sensor on pump delivery	cannot monitor pressure development at start-up 4
halfMileLine morePressure in	halfMileLine unit can be locked in	potential for liquid lock in and damage to unit by thermal expansion 3
halfMileLine moreTemp out	dummyHead1 high temp upstream, halfMileLine hot weather, pumpJ1 external fire, halfMileLine external fire, pumpJ1 lessFlow out	design temp exceeded 2
halfMileLine maintenance	halfMileLine no drains available	inadequate isolation and drainage 2

dummyHead1 revFlow out	pumpJ1 incorrect pump setup/installation, dummyHead1 low pressure upstream, tail1 high pressure downstream, pumpJ1 loss of drive	possible upstream contamination 3

Appendix 8: Autohazid reports for test case 2

No fluid model

```
Report for FULL PLANT HAZOP.
HAZOP started at        Tue Mar 10 13:17:15 1998
HAZOP completed at      Tue Mar 10 13:18:18 1998

-----------------------------------------------------


Library Used:           library2
Plant Used  :           plants\test2.txt
Results File:           results\test2.no
Templates File:         tlib
Fluids File :           fluidlib


Flag Settings Used

display faults with no consequences             NO
display consequences with no causes             NO
filter out repeat faults                        YES
filter out repeat fault-conseq pairs            YES
display protections present                     NO
only display faults with no protections         YES
consequence rank threshold set at               1


-----------------------------------------------------
```

DEVIATION	CAUSE	CONSEQUENCE
bufferTank1 lessFlow in1	dummyHead1 blockage - solids, bufferTank1 morePressure vapour, dummyHead1 blockage - frozen fluid, dummyHead1 low pressure upstream	solid settling 3, potential layering and rollover 2
	dummyHead1 leak to environment	solid settling 3, fire or explosion risk 2, toxic release 2, potential layering and rollover 2, fire/explosion risk 2, non-hazardous release 1
bufferTank1 moreFlow in1	dummyHead1 high pressure upstream, bufferTank1 lessPressure vapour	potential static fire/explosion hazard 3, increased port erosion - dissolved gas 3, increased port erosion 3, frothing 3

bufferTank1 noFlow in1	dummyHead1 no flow upstream	solid settling 3, potential layering and rollover 2
bufferTank1 moreTemp in1	dummyHead1 high temp upstream	increased vibration due to flashing liquid 3
bufferTank1 lessFlow out1	bufferTank1 liquid leak to environment	toxic release 2, flammable liquid release 2
bufferTank1 moreFlow out1	tail2 leak to environment	fire/explosion risk 2, toxic release 2, non-hazardous release 1
bufferTank1 lessPressure vapour	bufferTank1 vapour leak to environment	air ingress - explosion risk 4, air contamination 3, possible vacuum collapse 2, toxic vapour release 2, flammable vapour release 2
	bufferTank1 lessTemp vapour	air ingress - explosion risk 4, air contamination 3, possible vacuum collapse 2
bufferTank1 morePressure vapour	tail3 partly blocked, bufferTank1 moreTemp vapour, bufferTank1 moreTemp topLiquid	possible overpressure rupture 2
bufferTank1 moreTemp vapour	bufferTank1 moreTemp topLiquid, bufferTank1 external fire	design temp exceeded - structural weakening 2
bufferTank1 moreLiquid vapour	dummyHead1 high pressure upstream, bufferTank1 moreLevel topLiquid, bufferTank1 lessPressure vapour	liquid droplet entrainment 3
bufferTank1 moreVapour vapour	bufferTank1 moreTemp topLiquid	vacuum collapse - increased condensibles 2
bufferTank1 lessTemp topLiquid	bufferTank1 increased concentration of volatiles, bufferTank1 cold weather, dummyHead1 low temp upstream, bufferTank1 increased reaction	freezing 3
bufferTank1 moreTemp topLiquid	dummyHead1 high temp upstream, bufferTank1 hot weather, bufferTank1 external fire	crystallisation 3, design temp exceeded - structural weakening 2

bufferTank1 moreComposition topLiquid	dummyHead1 high composition upstream	increased corrosion 3
bufferTank1 contamination topLiquid	dummyHead1 upstream contamination	increased corrosion 3, liquid contents contaminated 3
bufferTank1 lessLevel topLiquid	bufferTank1 liquid leak to environment	gas breakthrough 3, flammable liquid release 2, toxic liquid release 2
	tail2 low pressure downstream, bufferTank1 liquid leak to environment, dummyHead1 low composition upstream, bufferTank1 morePressure vapour, dummyHead1 low pressure upstream, dummyHead1 blockage - solids, dummyHead1 leak to environment, dummyHead1 no flow upstream, tail2 leak to environment, dummyHead1 blockage - frozen fluid	gas breakthrough 3
bufferTank1 moreLevel topLiquid	dummyHead1 high pressure upstream, bufferTank1 lessPressure vapour	vessel overfilling 2
	tail4 blocked by frozen fluid, tail2 complete blockage downstream, dummyHead1 high composition upstream, tail2 blockage - solids, tail4 complete blockage downstream, tail2 blockage - frozen fluid, tail4 partly blocked, tail2 high pressure downstream	liquid droplet entrainment 3, vessel overfilling 2
bufferTank1 lessTemp botLiquid	bufferTank1 lessTemp topLiquid	freezing 3
bufferTank1 moreTemp botLiquid	bufferTank1 moreTemp topLiquid	crystallisation 3
bufferTank1 moreComposition botLiquid	dummyHead1 high composition upstream	corrosion 3
bufferTank1 lessLevel botLiquid	dummyHead1 low composition upstream, bufferTank1 liquid leak to environment	incorrect liquid out 3
bufferTank1 maintenance	bufferTank1 cannot isolate necessary lines	inadequate isolation and drainage 2
dummyHead1 revFlow out	dummyHead1 low pressure upstream	possible upstream contamination 3

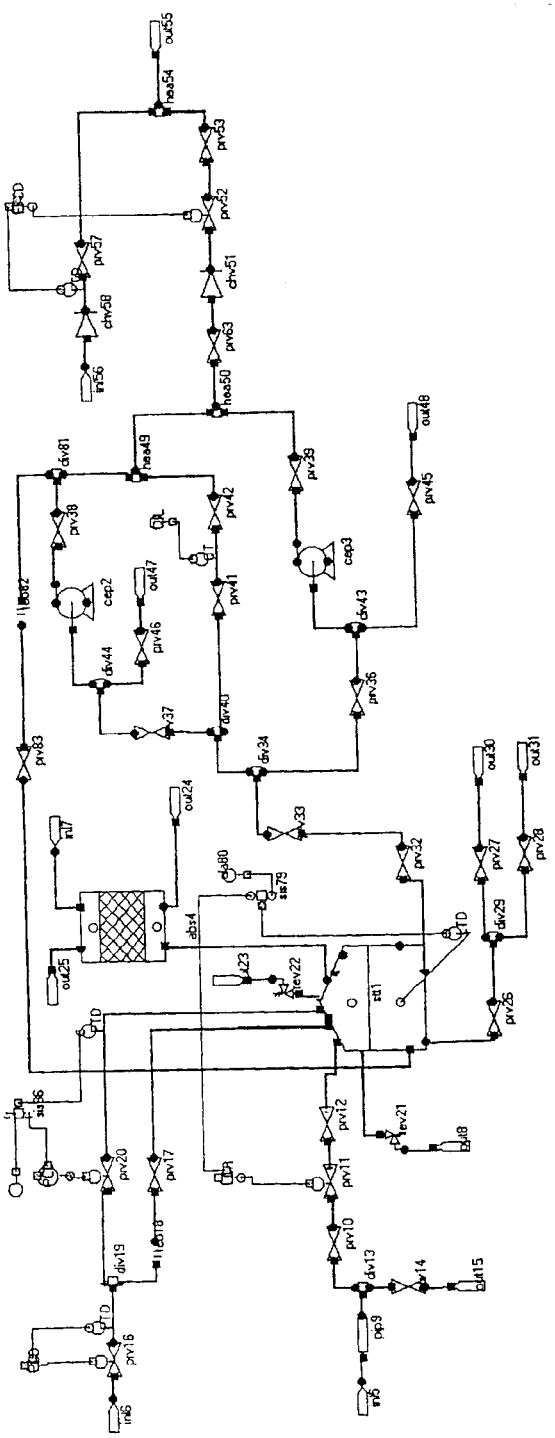
With limited fluid model

```
Report for FULL PLANT HAZOP.
HAZOP started at         Tue Mar 10 15:25:18 1998
HAZOP completed at       Tue Mar 10 15:26:20 1998

-------------------------------------------------------


Library Used :           library2
Plant Used   :           plants\test2.txt
Results File :           results\Test2.unc
Templates File:          tlib
Fluids File  :           fluidlib


Flag Settings Used

display faults with no consequences              NO
display consequences with no causes              NO
filter out repeat faults                         YES
filter out repeat fault-conseq pairs             YES
display protections present                      NO
only display faults with no protections          YES
consequence rank threshold set at                1


-------------------------------------------------------
```

DEVIATION	CAUSE	CONSEQUENCE
bufferTank1 lessFlow in1	dummyHead1 leak to environment	fire or explosion risk 2, fire/explosion risk 2
bufferTank1 moreFlow out1	tail2 leak to environment	fire/explosion risk 2
bufferTank1 lessPressure vapour	bufferTank1 lessTemp vapour, bufferTank1 vapour leak to environment	air contamination 3, possible vacuum collapse 2
bufferTank1 morePressure vapour	tail3 partly blocked, bufferTank1 moreTemp vapour, bufferTank1 moreTemp topLiquid	possible overpressure rupture 2
bufferTank1 moreTemp vapour	bufferTank1 moreTemp topLiquid, bufferTank1 external fire	design temp exceeded - structural weakening 2
bufferTank1 moreLiquid vapour	dummyHead1 high pressure upstream, bufferTank1 moreLevel topLiquid, bufferTank1 lessPressure vapour	liquid droplet entrainment 3
bufferTank1 moreVapour vapour	bufferTank1 moreTemp topLiquid	vacuum collapse - increased condensibles 2

bufferTank1 moreTemp topLiquid	dummyHead1 high temp upstream, bufferTank1 hot weather, bufferTank1 external fire	design temp exceeded - structural weakening 2
bufferTank1 contamination topLiquid	dummyHead1 upstream contamination	liquid contents contaminated 3
bufferTank1 lessLevel topLiquid	tail2 leak to environment, bufferTank1 (etc) liquid leak to environment, dummyHead1 leak to environment, dummyHead1 low composition upstream, bufferTank1 morePressure vapour, tail2 low pressure downstream, dummyHead1 no flow upstream, dummyHead1 low pressure upstream	gas breakthrough 3
bufferTank1 moreLevel topLiquid	dummyHead1 high pressure upstream, bufferTank1 lessPressure vapour	vessel overfilling 2
	tail4 partly blocked, tail4 complete blockage downstream, dummyHead1 high composition upstream, tail2 high pressure downstream, tail2 complete blockage downstream	liquid droplet entrainment 3, vessel overfilling 2
bufferTank1 lessLevel botLiquid	dummyHead1 low composition upstream, bufferTank1 liquid leak to environment	incorrect liquid out 3
bufferTank1 maintenance	bufferTank1 cannot isolate necessary lines	inadequate isolation and drainage 2
dummyHead1 revFlow out	dummyHead1 low pressure upstream	possible upstream contamination 3

Appendix 9: Trichloroethane storage system



Published by

Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
Phone internat. +358 9 4561
Fax +358 9 456 4374

Series title, number and report code of publication

VTT Publications 393
VTT–PUBS–393

Author(s)
Heino, Perttu

Title

Fluid property reasoning in knowledge-based hazard identification

Abstract

The study of serious accidents, which have occurred in the chemical process industry in recent times, highlights the need to understand fluid property related phenomena and the interactions between chemicals under abnormal process conditions or with abnormal fluid compositions. Consideration of these issues should be common practice in professional safety analysis work, and computer programs designed to support this work have to be able to deal with them.

The purpose of Hazard and Operability (HAZOP) study is to identify all possible deviations from the way a plant design is intended to be operated and all hazards associated with these deviations. Due to its systematic nature, the method is a good candidate for automation. Several research groups have developed embryonic knowledge-based HAZOP systems. However, no automated hazard identification features are included in current commercial software packages supporting HAZOP. The main problem of knowledge-based HAZOP systems is their poor performance in relation to the correctness and completeness of the resulting HAZOP study.

This thesis describes a novel methodology for fluid property reasoning in connection to knowledge-based HAZOP. Building on the earlier achievements of Loughborough University (LU) and Technical Research Centre of Finland (VTT) researchers, the methodology enables knowledge-based hazard identification programs to make a more intelligent assessment of the potential hazards and their causes.

In the first phase of the study, a rule-based fluid property and reaction property reasoning system was created for use in the HAZOPTOOL program. In the second phase, the LU fault propagation reasoning methodology implemented in the AutoHAZID HAZOP emulation program was extended with fluid property reasoning capabilities.

AutoHAZID was subjected to extensive evaluation which consisted of an evaluation workshop, a fluid model oriented study of the workshop results, and comparative testing based on a set of test cases. It was shown that it is possible and beneficial to extend knowledge-based HAZOP with a capability to reason about fluid properties and interactions. A framework for such a system is presented in this thesis together with some ideas for future work. Based on the results of the work reported here, it is recommended that fluid property reasoning is taken into use in any application of knowledge-based hazard identification.

Keywords
safety, hazard identification, HAZOP, computer-assisted hazard identification, physical and chemical properties, knowledge-based systems, process industry

Activity unit
VTT Automation, Risk Management, Tekniikankatu 1, P.O.Box 1306, FIN–33101 TAMPERE, Finland

ISBN	Project number
951–38–5395–0 (soft back ed.) 951–38–5396–9 (URL: http://www.inf.vtt.fi/pdf/)	61EUIST

Date	Language	Pages	Price
September 1999	English	167 p. + app. 53 p.	E

Commissioned by
National Technology Agency (Tekes), CTCI Corporation, Taipei, Taiwan

Series title and ISSN	Sold by
VTT Publications 1235–0621 (soft back ed.) 1455–0849 (URL: http://www.inf.vtt.fi/pdf/)	VTT Information Service P.O.Box 2000, FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374