IChemE
www.icheme.org/journals
doi: 10.1205/psep.04055

0957–5820/05/$30.00+0.00
© 2005 Institution of Chemical Engineers
Trans IChemE, Part B, November 2005
Process Safety and Environmental Protection, 83(B6): 509–532

PHASUITE: AN AUTOMATED HAZOP ANALYSIS TOOL FOR CHEMICAL PROCESSES

Part I: Knowledge Engineering Framework

C. ZHAO, M. BHUSHAN and V. VENKATASUBRAMANIAN*

Laboratory for Intelligent Process Systems, School of Chemical Engineering, Purdue University, West Lafayette, IN, USA

Hazard and operability analysis (HAZOP) is widely used in process hazard analysis of chemical processes. But it is time and effort consuming. To aid human experts in conducting HAZOP analysis in a more thorough and systematic way, a software system (PHASuite) for automated HAZOP analysis has been developed. PHASuite can greatly increase the efficiency of the HAZOP analysis, support best analysis practices, and provide foundation for reuse of the safety knowledge generated from the analysis. Given the knowledge intensive nature of HAZOP analysis, a knowledge engineering framework was developed. From the point of view of functionality, the framework consists of four main parts: information sharing, representation, knowledge base and reasoning engine. Ontology based information sharing schemes are developed to share process information and results with other systems. Coloured Petri Nets based representation, created from process information, is used to represent chemical processes as well as the methodology for HAZOP analysis. In this framework, a process is decomposed into two abstraction levels, operation level and equipment level, which are bridged using functional representation. Analysis is carried out on both the levels. Knowledge used in this system is stored in models. Case-based techniques are adopted for knowledge management. Knowledge base is stored externally in structured databases. A two-level, two-layer reasoning engine has been designed to operate on the Petri Nets representation of the process using the knowledge base to perform HAZOP analysis.

Keywords: automated HAZOP analysis; knowledge engineering; coloured Petri Nets; functional representation; model-based reasoning; case-based reasoning.

INTRODUCTION

Safety is an important issue in process design and operation in chemical industry. It is even more critical for modern chemical manufacturing processes, which are either operated under extreme conditions to achieve maximum economic profit, or are highly flexible, such as the specialty chemical or pharmaceutical processes. The importance of safety analysis in process operation is well recognized after occurrence of several tragic accidents that could have been avoided by adequate process safety analysis (Kletz, 1999). As a result, OSHA set the standard for process safety management (Title 29 CFR 1910.119), which requires the use of a process hazard analysis (PHA) technique for the identification of process hazards. It is estimated that 25 000 plant sites in US are covered by the OSHA standard.

Process Hazard Analysis

To ensure safe operation, process hazard analysis (PHA) is very important to proactively identify the potential safety problems and recommend possible solutions. There are several techniques for performing PHA, including What-If, Checklist, Hazard and Operability (HAZOP), Failure Modes and Effects Analysis and Fault Tree Analysis. A comparison of these methods and review of efforts in automating Checklist, FMEA, and Fault Tree Analysis, can be found in Vaidhyanathan (1995).

HAZOP is the method for identifying hazards and problems that prevent efficient and safe operation. It is easy to learn and use, and can be easily adapted to almost all the operations that are carried out within process industries. Hence it is widely accepted as the method for conducting PHA studies for chemical processes. A history of

**Correspondence to*: Professor V. Venkatasubramanian, Laboratory for Intelligent Process Systems, School of Chemical Engineering, Purdue University, West Lafayette, IN 47907, USA.
E-mail: venkat@ecn.purdue.edu

HAZOP analysis can be found in Kletz (1999). HAZOP studies are usually carried out by a team consisting of four to five members, who are experts on different aspects of the process, led by a team leader. The study requires the team to have detailed knowledge of the way the plant is intended to function. A typical team may consist of each of the following members: a chemical engineer, mechanical engineer, R&D chemist, production manager, and may be an instrumentation engineer and so on.

After defining the objective and scope of the study, selecting the team, and preparing the required documents, the study is carried out. Process is decomposed into manageable study units. Guidewords, including NONE, PART OF, MORE OF, MORE THAN, LESS OF, OTHER THAN, are applied to various aspects of the design intent to generate potential deviations. The effects of these deviations on the process or operation are analysed. The deviations resulting in hazardous or significant consequences are recorded along with the possible causes, needed actions, and questions/comments/recommendations. Detailed descriptions of the analysis procedure with illustrative examples have been given by CCPS (1985) and Kletz (1999). HAZOP analysis can be used in different stages of a process, including at design phase for early checking for major hazards, at design freeze phase, at pre start-up, prior to plant modifications, prior to taking a plant out of service and so on. The possible variances can be found in Knowlton (1981).

Automated HAZOP Analysis

HAZOP is, however, very time-consuming. According to one estimate (Freeman, 1992), for a process with eight P&ID consisting of simple, standard, complex and very complex drawings, a team of five people led by an experienced team leader, the completion of the HAZOP analysis needs 441 man hours, and the overall elapsed time is about 8 weeks. So an automated tool is definitely helpful. But a fully automated HAZOP analysis tool, which can totally replace the human team, is very difficult and nearly impossible to realize in the near future. The difficulties lie in the fact that the highly flexible reasoning mechanism and knowledge structure of human experts cannot be effectively simulated by computer systems. Nevertheless, as Kletz (1999) pointed out, an automated HAZOP analysis tool can certainly be used as an aid for human experts. Not only can automated HAZOP analysis tool, like PHASuite, help in making PHA study more thorough, systematic and consistent, improving the quality of the review, reducing the time and effort of the team, and documenting results for regulatory compliance, but can also be used in training new operators and for online abnormal situation management. The key functionality of automated HAZOP analysis tool is its reasoning capability. Most of the commercially available HAZOP tools, such as PHAWorks, and PHAPro, can be considered as documentation tools which provide workflow support for HAZOP analysis. Even though library of possible hazardous are usually provided, these tools do not use them to carry out reasoning autonomously.

Several attempts for automating HAZOP analysis have been made in the past two decades. Parmar and Lees (1987) represented the knowledge required for propagating faults in each process unit using qualitative propagation equations and event statements for initiation and termination of faults. The causes are generated by searching for the initial events and the consequences by searching the terminal events. However, the analysis is not complete since the causes and consequences are not propagated. The system is also very limited since the knowledge was hardwired for the given process. Waters and Ponton (1989) attempted to use a quasi-steady state qualitative simulation approach to automate HAZOP analysis. They found the approach to be highly combinatorial, thus restricting its practical usefulness. Karvonen *et al.* (1990) developed a rule-based expert system prototype using the KEE shell. The system's knowledge base consisted of the structure of the process and rules for searching causes and consequences. The generality of the system is limited due to the structure of the rules dependent on the process structure. Catino and Ungar (1995) developed a prototype HAZOP identification system which works by exhaustively locating possible faults, automatically building qualitative process models, simulating them and checking for hazards. This approach starts with faults whereas the HAZOP analysis starts with process deviations. Also this approach is too fine-grained for industrial scale process hazards reviews. A detailed survey of the literature on intelligent systems for HAZOP analysis can be found in Venkatasubramanian *et al.* (2000). Recently, Kang and Yoon (2001) proposed to use multiple models consisting of the unit function model, the unit behaviour model, the process structure model and the process material model to describe thechemical processes from a safety-oriented point of view.

Most of the above approaches to automated hazards analysis were demonstrated on small-scale processes or academic prototypes. Venkatasubramanian and coworkers have been working extensively in this area, and the systems developed by their group have been applied to a wide-variety of industrial scale process plants. Vaidhyanathan and Venkatasubramanian (1996) developed a model-based framework for continuous processes in which the knowledge required to perform HAZOP analysis is divided into process-specific and process-generic components. Process-generic knowledge consists of signed digraph models of the process units. Based on the framework an expert system HazopExpert was developed. Srinivasan and Venkatasubramanian (1998) adopted Petri Net representation and SDG based HAZOP analysis for batch processes, and developed the system BatchHazopExpert. At the same time, iTOPS, an intelligent tool for operating procedure synthesis was developed by Viswanathan *et al.* (1998). Since the operating procedure synthesis and safety analysis share a lot of information, a scheme for integrating iTOPS and BatchHazopExpert (BHE) was proposed by Zhao *et al.* (2000). All the above systems were developed in G2, an expert development shell (Gensym, 1997).

Although the above systems can be used to demonstrate the proposed methodologies for automating HAZOP analysis, they have some severe drawbacks, the important ones being:

(1) Most of the earlier attempts have focused on the reasoning methodologies. Issues regarding knowledge representation, storage and management are not sufficiently addressed, and this leads to poor scalability of those systems.

(2) All the systems are only prototypes developed with the aim of demonstrating different ideas. Most of them were developed in G2 or other expert shells, which are good for prototyping, but have severe limitations for further development in terms of portability, scalability, and computational speed.
(3) Information sharing with other software systems is not addressed.
(4) Among the systems developed at LIPS laboratory, HazopExpert and BatchHazopExpert share a lot of similarity in functionality and structure, but they are implemented as two separate systems.
(5) The integration between different systems is not well addressed. The integration of iTOPS and BatchHazopExpert is a straightforward translation between objects, which is time consuming and confusing.

The goal of this work is to develop PHASuite, as a high quality full-scale software system for automated HAZOP analysis, to address abovementioned issues with previous systems. PHASuite is a knowledge-based system, reasoning and analysing on chemical processes. The knowledge consists of models as well as experience related to process safety.

From the functionality point of view, PHASuite consists of five main components, as illustrated in Figure 1. Information necessary for the analysis is first input to the system. In order to enable the computer program to analyse the process, the information is then transformed to a proper representation suitable for automated analysis. Knowledge for safety analysis, probably wrapped in models, is used to model the process. Reasoning engine, consisting of the control knowledge which uses the knowledge to perform analysis on the representation of the process, is then invoked. The results generated from analysis are then reviewed by the user through results management facilities. Automated HAZOP analysis is a very difficult task due to the complex knowledge structure, flexible analysis depth, sophisticated user interaction, and requirement for learning through experience. The main difficulties of automated HAZOP analysis are related to:

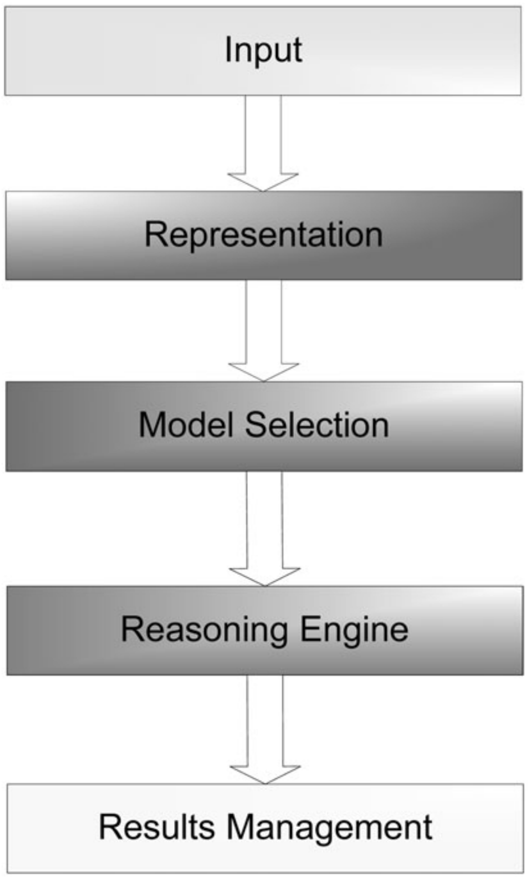


Figure 1. Main issues on automated HAZOP system.

- Representation of process specific information, i.e., operating procedures, P&ID and so on, which should be suitable for HAZOP analysis.
- Representation of process generic information, i.e., different kinds of knowledge for operation and equipments, representation of experience and knowledge acquisition.
- Reasoning methodologies to apply knowledge on the representation of the process to generate analysis results.

Development of PHASuite is divided into two steps. The first step is to design a knowledge engineering framework for automated HAZOP analysis using various artificial intelligence techniques and the second step involves using appropriate software engineering techniques to implement the system. Accordingly, this paper is divided into two parts. The first part focuses on the conceptual design of the system, i.e., the knowledge engineering framework behind the system. And the second part concentrates on the implementation guidelines and uses an industrial case study of a pharmaceutical process to illustrate the applicability and procedure of using PHASuite to assist human experts in HAZOP analysis.

Major components of the proposed knowledge engineering framework for PHASuite are discussed in the rest of this part of the paper. The next section presents the ontologies used in this work in order to develop the system in an open way and to share information with other software systems. Knowledge representation, storage, management and acquisition issues are addressed in the third section. Reasoning methodologies are discussed in the fourth section.

ONTOLOGY BASED INFORMATION SHARING

Ontology

To be successful, an automated HAZOP system should be constructed based on an open architecture, in which knowledge is explicitly represented and information is shared among different systems. Ontologies are the basis of such an open architecture and have an important role in the information sharing schemes presented in the knowledge engineering framework.

With the development of technology, software systems have been developed for different areas as well as different parts of the same area by different people and organizations. Each of them uses different jargon. The way to enable better communication between people, organizations and software, is to come to a shared understanding by reducing or eliminating conceptual and terminological confusion. As pointed out in Uschold and Gruninger (1996), such an understanding can serve as a unified framework for the different viewpoints, and can help to achieve (1) better communication between people, and (2) inter-operability among systems. The latter is achieved by translating between different models, paradigms, languages and software tools. The shared understanding is the basis for a formal encoding of the important entities, attributes, processes and their inter-relationships in the domain of interest. This formal representation may be a shared component that is reusable in a software system. Automation of consistency checking is possible and ensures the reliability of translation. In this paper, ontology is defined as an explicit specification of a conceptualization (Gruber, 1993).

An important motivation for ontologies is to integrate models of different domains into a coherent framework. Ontologies are used as inter-lingua for the purpose of inter-operability. In this work, the design of ontologies follows the steps discussed above. The main purpose for developing the ontologies is to use it as inter-lingua for external inter-operability between PHASuite and other software systems, and for internal inter-operability among components in PHASuite as well. The language used to define the ontologies is currently informal, i.e., mostly are defined in natural language. Although this kind of definition is sufficient for current implementation, definitions created using more formal language are desired for further implementation. Using KSL Ontology Server (Farquhar *et al.*, 1995), which is based on ontolingua, a mechanism for writing ontologies in a canonical format, seems to be promising.

Information Sharing Scheme for PHASuite

Process safety analysis is an information intensive process. Large amount of information about different aspects from different sources is necessary for analysis. The main required information can be divided into four types: material, P&ID, chemistry and operating procedures. Most of the information is very basic and would be required for specifying a process and hence would probably be available in formats used for other purposes. Re-entering that information into PHASuite can be time-consuming and error-prone. Thus the capability of information sharing with other information sources or providers is very crucial for successful deployment of PHASuite. The best way for information sharing is to share the ontologies among different information sources. It is, however, not the case currently since the common ontologies do not exist and all the systems have been developed without the guidance of the common ontologies. So the basic scheme applied in PHASuite consists of two steps as illustrated in Figure 2. The first step is to access the information source and translate the required information into a format suitable for PHASuite. This transformation is guided by a dictionary created from the ontologies defined for PHASuite. Interface storage is constructed to store this translated information. The second step is to generate the run-time file from the interface storage. The following sections demonstrate this scheme using examples of sharing operating procedure information with Batch Plus and safety analysis results with PHAPro. Approaches for using this methodology for sharing information with MSDS and P&ID are also proposed and discussed.

Sharing operation information with Batch Plus

For the operation related information, we need concepts for representing operating procedures, operating conditions, and parameters. The hierarchical level of ISA-S88.01 standard, which consists of procedure, unit procedure, operation, and phase, is followed in this work (ANSI/ISA-S88.01, 1995). To specify information related to operating procedure, parameters and operating conditions are also needed. Examples of parameter include 'Fraction' and 'Method' in *Transfer* operation. Examples of operating conditions include temperature, pressure, and time of the operation. Definition of process chemistry is straightforward and can be divided into reaction and separation. A reaction is specified by reaction type, reactants, products, excessive reactants, inhibitors, catalysts and solvents. A separation is specified by input materials, names of different phases, and materials involved in each phase. Equipment is specified by design properties, such as design temperature, design pressure and design capacity, as well as some structural specifications including whether the equipment has a jacket and so on.

Batch Plus is a product from Aspen Technology Inc. (2000). As a batch simulation tool, it needs information about the operating procedures, equipment and some properties of materials. Most of this information is also necessary for carrying out safety analysis. Thus it is very useful for PHASuite to share information with Batch Plus. To achieve this, dictionaries are designed based on the

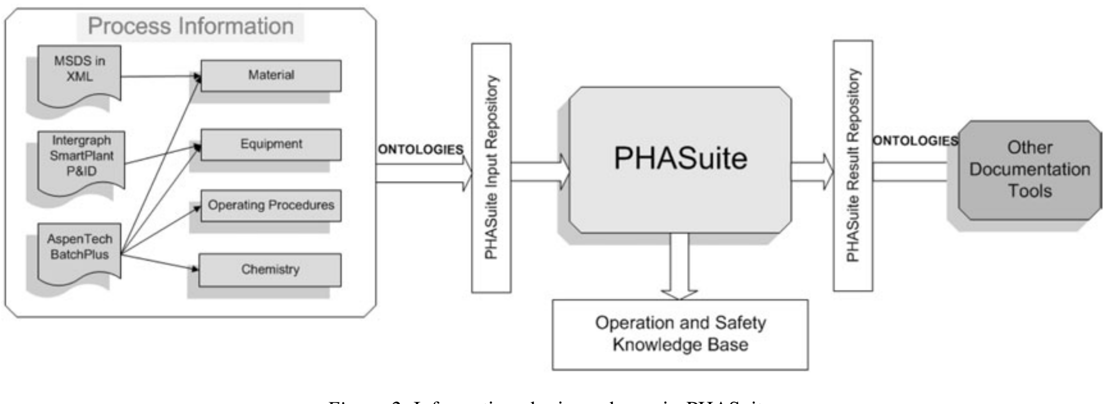

Process Information
MSDS in XML
Intergraph SmartPlant P&ID
AspenTech BatchPlus
Material
Equipment
Operating Procedures
Chemistry
ONTOLOGIES
PHASuite Input Repository
PHASuite
PHASuite Result Repository
ONTOLOGIES
Other Documentation Tools
Operation and Safety Knowledge Base

Figure 2. Information sharing scheme in PHASuite.

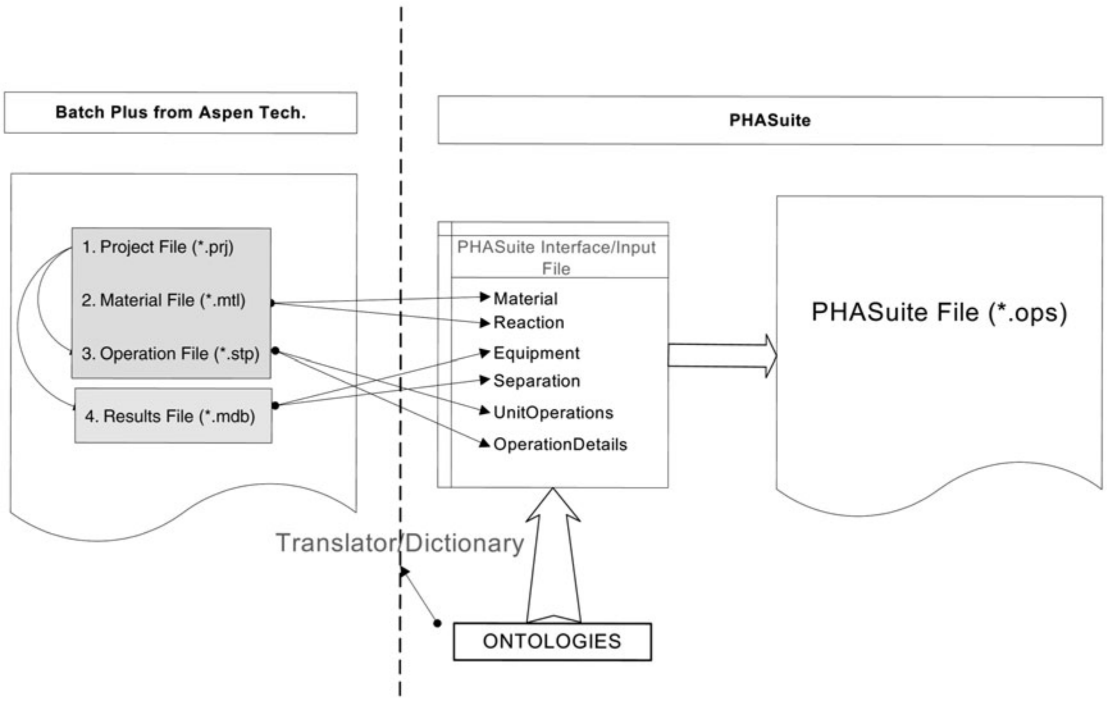


Figure 3. Sharing process information with Batch Plus.

ontologies defined in PHASuite for different levels of operations, equipment, and material. The dictionary includes corresponding operation name from PHASuite for each operation in Batch Plus, for example, OperationHoldFor BlockOperation corresponds to Age, and OperationClean-VesselForBlockOperation corresponds to Clean in Batch Plus. The dictionary also includes mapping for each parameter in each operation. For example, parameter 'Agitator Speed' in Age operation of Batch Plus is mapped to 'Rpm' in OperationHoldForBlockOperation in PHASuite.

Figure 3 shows the flow chart of importing information from Batch Plus. Process information in Batch Plus is stored in several separate database files. Each project in Batch Plus may contain one or more steps. Operation information for each of the steps is stored in separate files with file extension of *.stp. The step information is obtained from the project file with extension of *.prj. Material information is stored in a file with extension of *.mtl, and equipment information is stored in a file with extension of *.eqm. Besides the input files, simulation results are stored in a file with extension of *.mdb. The dictionaries constructed under the guideline of ontologies are used to guide in the translation of the information in Batch Plus to information accessible through PHASuite. Material and reaction information are imported from the material database, and the Unit Procedure and Operation Details are imported from operation database. Since separations are not specified explicitly in Batch Plus, this information is gathered through the stream information from simulation results. Equipment information is better organized in result database, so it is gathered from that source. The information thus gathered is stored in an interface/input interface storage (as a database) which is readily accessible by PHASuite. Then the corresponding object representation of that information is created in PHASuite and saved as a project file.

Sharing results with safety documentation tools

The safety related information includes the information which is not normally part of operation related information, such as the results generated from process safety analysis. A typical result from safety analysis includes the information on (1) location of the deviation, including the operation where the deviation is introduced; (2) the equipment; (3) deviation, including the parameters in which the deviation is introduced; (4) deviation type, such as high, low, none and so on; (5) consequence which is a text string to describe possible hazards or operation problems; (6) cause which is a text string; (7) location of consequence and location of cause; (8) safeguards which is a text string, and (9) recommendation which is a text string, as well as other useful information such as (10) cause causal path and consequence causal path.

The results analysed using PHASuite are stored in results database following the ontologies designed for the results. Although PHASuite itself provides facilities for results documentation and reporting, it may also be useful to export the results to other safety results documentation tools, such as PHAPro® (Dyadem, 2001). Figure 4 shows

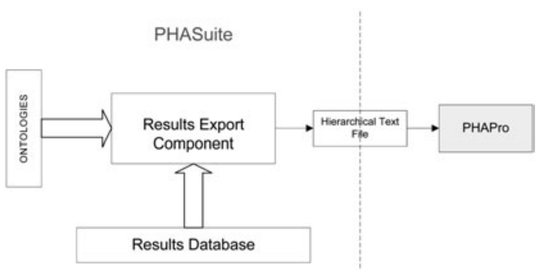


Figure 4. Exporting results to PHAPro from PHASuite.

the flow chart for PHASuite to export its results to PHAPro®. PHAPro® can access a hierarchical text file as input. Ontologies for the analysis results are used to guide the design of the import field mapping file used by PHAPro® to map the fields in the text file to the fields understandable by PHAPro®. The results export facility of PHASuite constructs the text file by reading results from results database and exporting the results to a hierarchical text file.

Sharing other information

Besides operating procedures, material hazardous properties and detailed process equipment diagram are two other important information. Similar to the scheme presented earlier, a two step approach is followed with the help of dictionaries created using appropriate ontologies. This implementation is in planning stage, while the sources of the information are to be determined.

Material properties, especially those related to safety, are important information for safety analysis. Material safety information is normally stored in the format of Material Safety Data Sheet (MSDS), which is typically a text file with a few sections devoted to different aspects of material properties and safety concerns. It is very hard to access the information stored in such a text format, although some parsing tools, such as Perl, can be used to find out certain information by string searching, even though the information obtained in this manner may be ambiguous. In recent years, XML has emerged as a popular file format for exchanging information (Harold and Means, 2001). Some efforts have been made to convert the MSDS to standardized XML format (Esoh.org, 2002). The central component will be a XML parser and a dictionary or translator defined using ontologies. Similar to other information, the material information is stored in an interface database of PHASuite, and the object representation for materials is constructed from this database at run-time.

P&ID is another very important information source for process safety analysis. For a modern chemical process, P&ID is normally very complex and it is a tedious process to recreate such a drawing in PHASuite. Most of the P&ID drawings are in some electronic format, such as AutoCAD (Autodesk, 2002), Intergraph (2002) and so on. The drawings created using older version of CAD tools are composed of lines or curves. In recent years, with the development of object oriented programming, newer CAD tools are object-based and the basic drawing components are blocks instead of lines, and some of them are data-centric. This data-centric approach makes it possible for PHASuite to share information with them. Appropriate Visual Basic code, using algorithms similar to those currently coded in PHASuite, can be embedded in the SmartPlant® P&ID to generate process flow diagram (PFD) from P&ID. The information can then be translated into an interface database which is accessible by PHASuite, guided by the dictionary created under the guidance of ontologies for equipment. It may also be possible to move the P&ID drawing facility in PHASuite to SmartPlant P&ID by using its drawing facility to further specify the stream information in PFD.

KNOWLEDGE REPRESENTATION, STORAGE, MANAGEMENT AND ACQUISITION

Automated HAZOP analysis system is knowledge intensive and the analysis capacity and quality depend extensively on the quality of domain knowledge. Knowledge acquisition and representation are very important to gather domain knowledge and to use it effectively. The performance and capability of the system is also greatly affected by the approach used for knowledge storage and management. The knowledge required for process safety analysis can be divided into two main types: operation related knowledge and safety related knowledge. These two kinds of knowledge have different characteristics and purposes. The main purpose of operation knowledge is to guide the generation of lower level operations based on the process information in order to decompose the process into a level suitable for applying general domain knowledge.

Knowledge Representation

As discussed in Rich and Knight (1990), a good system for the representation of knowledge in a particular domain should possess the following four properties:

- Representational adequacy—the ability to represent all kinds of knowledge that are needed in that domain.
- Inferential adequacy—the ability to manipulate the representational structure in such a way as to derive new structures corresponding to new knowledge inferred from old.
- Inferential efficiency—the ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions.
- Acquisitional efficiency—the ability to easily acquire new information.

Various methodologies have been proposed in literature to represent knowledge, which can be divided into two main categories, declarative and procedural knowledge. A *declarative representation* is one in which knowledge is specified, but the use to which that knowledge is to be put is not given. To use a declarative representation, we must augment it with a program that specifies what is to be done to the knowledge and how. There are several schemes for declarative representation, including: simple relational knowledge, inheritable knowledge, and inferential knowledge (Rich and Knight, 1990). A *procedural representation* is one in which the control information that is necessary to use the knowledge is considered to be embedded in the knowledge itself. To use a procedural representation, we need to augment it with an interpreter that follows the instructions given in the knowledge. Procedural knowledge can be represented in programs in many ways. The most common way is simply to code it in some programming language. The machine uses the knowledge when it executes the code to perform a task. Unfortunately the disadvantages of representing procedural knowledge are the inferential inadequacy (it is very difficult to write a program that can reason about another program's behaviour) and acquisitional inefficiency (the process of updating and debugging large pieces of code becomes difficult).

Another way to differentiate between different kinds of knowledge is to classify them as explicit, implicit, and tacit knowledge (Nickols, 2000). Explicit knowledge is formal and systematic. As the first word implies, explicit knowledge has been articulated, and captured in form of text, tables, diagrams and so on. Tacit knowledge is knowledge that cannot be articulated. An example is that a person is able to recognize another person's face, but is only vaguely able to describe how that is done. Implicit knowledge is knowledge that can be but has not yet been articulated. The existence of such knowledge is implied by or inferred from observable behaviour or performance. Implicit knowledge can be identified by task analyst or knowledge engineer and becomes explicit knowledge.

In this system, declarative representation schemes, including semantic networks, frames, predicate logic, and production rules, are used to represent knowledge based on knowledge primitives such as properties of material, properties of equipment, specifications for separation and reaction, and details on different level of operating procedures. Separately, control knowledge on using the declarative knowledge to perform analysis is represented using procedural representation. This clearly distinct separation of declarative knowledge and its control knowledge is crucial for the system as will be seen in the following sections. Most of the knowledge used in this system is in the form of explicit knowledge. The structure of the knowledge base helps to articulate implicit knowledge into explicit knowledge which can then be incorporated into the knowledge base.

Knowledge Storage

After the schemes to represent knowledge are determined, the next logical question is how and where to store the knowledge. The requirements for knowledge base systems to be effective for large-scale applications are to be able to support efficient retrieval and update operations.

Over the last few years, database management systems (DBMS) have been developed to provide an open, structured and external storage mechanism for efficiently accessing and managing large data sets. They also provide data management facility through a variety of functionalities, including efficient storage and data sharing and so on (Ramakrishnan and Gehrke, 1999). DBMS is very important to the computer age and is used in almost every application. Among the several available data models in DBMS, relational data model is the most popular, since it is based on mathematically sound theory and has the important characteristics of physical data independence and logical data independence. As information processing begins to extend the need for database support into many areas outside the traditional fields like business, more expressive data models than the relational model are needed to capture the semantics of the applications. The object-oriented data model is one such concept that has been extensively researched. However, this concept has still not matured and none of the objected-oriented DBMS (OODBMS) are available for general use. So a lot of research has been conducted to provide object-oriented storage and access to a relational database.

Norrie *et al.* (1994) proposed architecture for large-scale knowledge base systems based on relational databases using three levels of semantic constructs—frames, objects and relations. The intermediate object level retains the structural semantics of the frame level and bridges the semantic gap between the frame and relational levels. This approach has been adopted in a hybrid knowledge base system. Ramanathan (1994) presented an approach to store and retrieve huge quantities of geophysical data using an object-oriented layer which was constructed based on the schema of the relational database. This approach was used to access a legacy database. Abernethy and Altman (1998) presented SOPHIA, a frame-based knowledge architecture which is built upon a relational DBMS to provide basic frame access capabilities.

All of the previous HAZOP analysis prototypes developed in LIPS lab store knowledge as procedures using programming languages. Stored procedures are easy to create, and can freely access all the information available in the system. The most severe drawback of this approach is, however, that the expansion or even modification of the knowledge base is very difficult, since even a small change or expansion in knowledge requires several steps including modifying the source code, and recompiling whole system or major part of the system before the change can take effect. As a consequence, learning or applying other knowledge management techniques is very difficult to achieve. The stored procedure scheme also results in other disadvantages including poor software structure due to no clear interface being defined between the model and other components of the system, which makes the system difficult to scale up.

The approach taken in this work is to use Microsoft Access as the DBMS media for knowledge storage for declarative knowledge. The reason for choosing Access is due to its wide availability as a component in Microsoft Office family. It is also easy to export the database in Access to other popular DBMS, such as Oracle, and even other open data standard formats, such as XML. Also the programming language adopted in this work, Visual C++ has embedded class libraries for accessing the Access database, which makes creating an interface between the internal storage and object layer easier. The schema for each table in each database follows ontology designed in a manner similar to those used for information sharing in Ontology Based Information Sharing section. The ontology is also used to construct the object layer as the interface between the knowledge base and the reasoning engine. This approach is similar to SOPHIA, where the slot-and-filler scheme is followed and each table corresponds to an individual knowledge class. As opposed to declarative knowledge, the control knowledge is rarely changed after being deployed. To increase the performance, control knowledge is not stored in Access, but has been hard coded in Visual C++. So the knowledge base which stores declarative knowledge and the control knowledge which has been embedded as source code are separate parts. The user can modify, create and expand the knowledge base without the existence of control knowledge as long as some specifications are observed. With the help of designed ontologies, it is also possible to use the knowledge base for purposes other than safety analysis. Changes to the control knowledge, mainly in the reasoning engine, and the interface between knowledge storage and object layer, is more complex, since the changes must be made in the source code after understanding the code clearly although extensive care has been taken to ensure easy understanding of the source code. After changes have been made, the code must be recompiled and redeployed before the modifications can take effect.

Operation knowledge

To represent a batch process, the ISA-S88.01 standard recommends the information to be divided into a hierarchical structure where the details about the process are specified at four levels. From bottom to top, at the lowest level is the phase, which describes tasks such as 'open valve', 'start pump' and so on. A set of phases can be logically grouped into an operation. Examples of operations are 'charge a reactor' or 'heat a reactor'. A sequence of operations makes up a unit procedure such as a 'reaction', 'distillation', or 'filtration'. At the highest level is the procedure that consists of a sequence of unit procedures. Viswanathan (2000) proposed an operating procedure synthesis framework to generate operating procedures using Grafcet chart based hierarchical planning approach. Starting with process specific knowledge including the process description and plant information, the nominal operating procedures are incrementally generated over four stages of planning. The stages involved are path search and equipment assignment, preliminary sequencing, final sequencing, and operating procedure synthesis. Each stage adds a layer of batch process operation information. In current implementation of PHASuite, the Unit Procedure level corresponds to the *BlockOperation*, and the Operation level to the *OperationForBlockOperation.*

It is assumed here that the information about the first three levels of the hierarchy, i.e., procedure, unit procedure, and operations are available. Operating procedures are part of the process information. The operating procedures for a process may have been generated manually and can be either manually input to PHASuite or automatically imported from other software where the operating procedures are already stored.

Although the hierarchical structure recommended by ISA-S88.01 standard is sufficient to describe a process, automated safety analysis needs further decomposition of the operation. The operation level is further decomposed into two intermediate levels, called Intermediate Operation Level 1 (InterOp-1) and Intermediate Operation Level 2 (InterOp-2). The decomposition of InterOp-1 from Operation level in ISA-S88.01 standard is based on the equipment, i.e., each InterOp-1 occurs in exactly one major equipment. For example, consider a typical filtration operation as:

> Transfer contents of unit 100-1 to 100-2. Transfer 100% of vessel contents. Transfer using FI-100. The transfer time is 25 min. The filter separates 100% of all solids.

Three equipments, two vessels 100-1 and 100-2 and one filter FI-100, participate in this filtration operation. Based on the decomposition criteria, the InterOp-1 level consists of three operations, i.e., *Transfer* with main equipment 100-1, *FilterFeed* with main equipment FI-100, and *Receive Filtrate* with the main equipment 100-2. This decomposition makes it possible to determine the materials involved in each InterOp-1 level operation. The operations in InterOp-1 and their relations can be shown graphically in Process Sequence Diagram (PSD). As will be shown later, the PSD is a very important tool for creating proper process representation for safety analysis.

The InterOp-2 level is used solely for the purpose of safety analysis and is used internally only. The user is not aware of the existence of this level of operation. The decomposition from InterOp-1 to InterOp-2 depends on the functionality. For example, the FilterFeed operation of the previous example is divided into Receive and Filter, where Receive operation describes the functionality of receiving material from 100-1 to FI-100, and Filter operation describes the functionality of filtration taking place in FI-100. There is a safety model corresponding to each type of InterOp-2 operation.

The knowledge for guiding the decomposition of operation to lower levels and the mapping between parameters is wrapped in the corresponding models for operations. The model structure for each level of operations is similar. The model consists of two main parts: operating conditions or parameters used to define the operation, and the relations with lower level operations. Every operation in any level has a corresponding model. The models are stored in Access databases.

In order to have a common object interface for different kinds of operations in each level, each operating condition or parameter is wrapped in *Parameter*, which contains four variables: (1) parameter-label; (2) parameter-type, which can be string, boolean, double, or set; (3) parameter-value; and (4) unit-of-measurement. For example, the operating condition with the label *Temperature*, can have parameter-type double, specified value 25 and unit-of-measurement '°C'. As another example, the *Method* parameter in *Charge* operation, has a parameter-type set, with possible values {*Vacuum, Pressure, Pump, Solid, Vacuum through line filter, Pressure through line filter*}. With such a unified object interface, it is possible for us to move from the current dialog based interface for user to access and modify operation specifications to a unified properties editor type of user interface. At the same time, this approach makes it possible to expand the operation without changing the source code.

The details on specifying the relation part of the model include the types of the lower level operations, the sequence of those operations, the condition for the decomposition, and the mapping of the operating conditions between the related levels of operations. The types of the operations are indicated by their label/name directly. Integer numbers are used to indicate the sequence of the operations. The conditions are written in terms of first order logic operators on the operating parameters. A mapping table has been created for each corresponding higher-level operation to lower level operation.

In the current version of PHASuite, nine BlockOperations have been defined: BlockCrystallization, BlockDrying, BlockHydrogenation, BlockMilling, BlockReaction, BlockCentrifugation, BlockDistillation, BlockExtraction and BlockFiltration. Fifty OperationForBlockOperations, for example, OperationAddForBlockOperation, OperationCentrifugeForBlockCentrifugation and so on, have been created. Further seventy operations in InterOp-1, including charge, transfer, receive, DryLod, Atmospheric-Distillation, and so on, have been created.

Safety knowledge

While operation knowledge guides the decomposition of the process, thereby enabling a better representation of the process, safety knowledge consists of the knowledge about what and how hazards could occur in chemical processes. The safety knowledge is sophisticated and important for the quality of the analysis. Rather than using shallow knowledge for the whole process, deep knowledge for each reusable process primitive is wrapped into safety models. Use of deep knowledge inside models is necessary for applying model-based reasoning techniques.

A safety model consists of several major components including digraph to represent the causal relationships between process variables, ports to define the connectivity between models, and rules for digraph functions and local causes and local consequences. For operation model, function description and equipments to carry out the operation are also parts of the model.

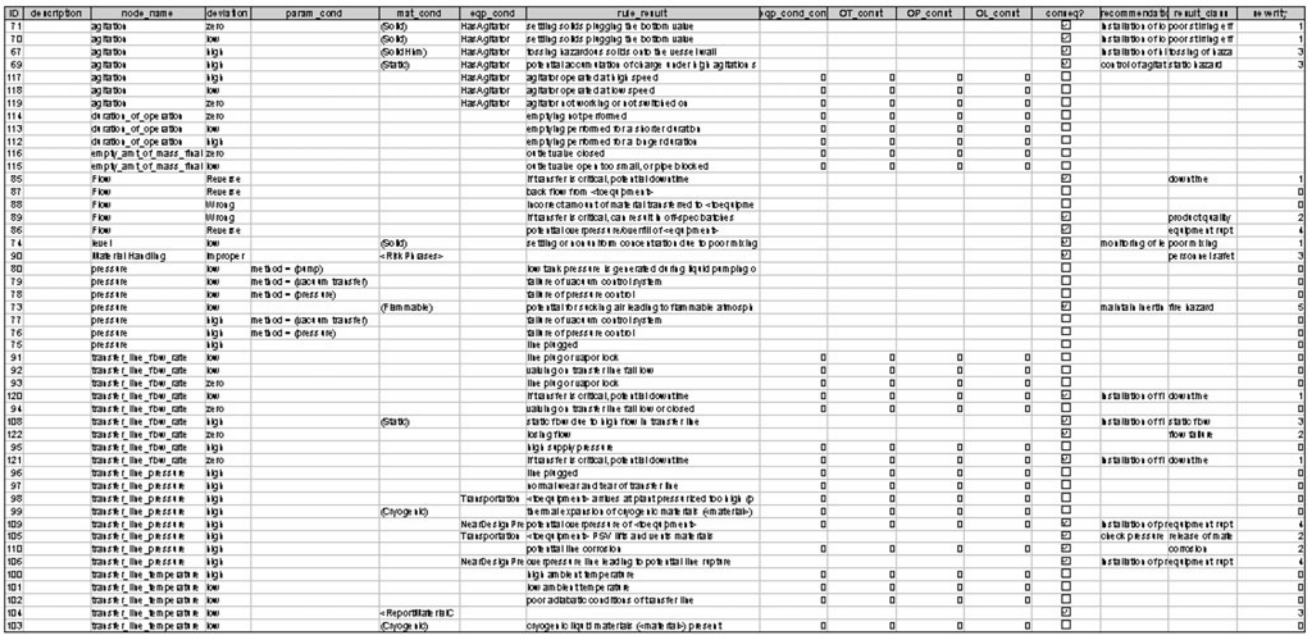
Different aspects of the model are stored within tables in a database file. Tables of *DgArc*, *DgCauseNode*, *DgConseqNode*, *DgFuncs*, and *DgNode* are used to describe the relations between process variables. *ModelView* table stores information about the position of the process variables which are used during graphical construction of the digraph.

Materials involved in each operation can be categorized into one or several sets. Table *MaterialSet* stores the knowledge of sets of materials involved in the operation. The default set is *materials_in_operation*, which describes all the materials involved in the operation. Other material set is described by the *SetName* and the *PortID* which determines the type and location of the input or output stream. For example, in the model for *charge* operation, material set *materials_from_source* describes the material charged into the main equipment during the charge operation. PortID of 0 describes that the materials flows in from an external source. As another example, there are two material sets specified in the model for distillation operation. The *materials_in_bottom* describes the bottom materials of the distillation operation and the flow out of the distillation column from the port with ID 101. The *materials_in_distill* describes the distilled materials of the distillation operation and the flow out of the distillation column from the port with ID 102. For operations with reaction taking place, *materials_as_reactant*, *materials_as_products*, *materials_as_catalyst*, *materials_as_solvent*, *materials_as_inhibitor* are material sets to describe different kinds of materials. The materials described by these sets are self explanatory based on the names of the sets.

Parameters table stores the knowledge about the parameters which are used to instantiate the model, besides the normal operating conditions including temperature, pressure, and level. Each parameter is defined by ParamName, ParamType which is same as the type definition in operation knowledge, and aMust which is a Boolean value to determine if this parameter must be present before analysis can be carried out. For example, the *method* parameter in *charge* operation has ParamType of 2, which represents a string, and aMust value of *false.* The *reaction* parameter for irreversible endothermic reaction operation has a type of 2, which is a string to specify the reaction name, and aMust value of *true* which means the analysis is meaningless without specifying the type of reaction for this reaction operation. The completeness of information is ensured by performing consistency check before analysis.

The *Ports* table stores the knowledge about the connectivity of the model. Usually, an operation model has one input stream and one or two output streams. The equipment model could have more complex structure for input or output streams. Each port in the *Ports* table is determined by the PortID, PortName, InputPhase which is a descriptor to connect the stream connected to the port to the process variables in the model, and OutputPhase which is also a descriptor to connect the stream connected to the port to the process variable in the model connected with this model. For example, the port with name *empty* in empty operation has ID of 102 which identifies that the stream connected to the port is an output stream, and the InputPhase is *empty* which identifies that the stream is connected to the process variables with prefix *empty_*, such as *empty_amt_of_mass_final*, *empty_amt_of_heat_final* and so on. The OutputPhase of the port is *fill*, which means the stream is connected to 'next' model with the process variables with prefix *fill_*.

The RuleSet table contains the knowledge about the possible local causes and local consequences for a specified deviation. Here 'local' means the direct effect (consequence) or reason (cause) of the deviation, without any consideration of the propagation of the effect or reason to other process variables in this model and without any consideration of the propagation to other models (both downstream and upstream). The concept of local knowledge makes it possible to construct the knowledge base independently without any specific process information and reuse the knowledge for different processes. Figure 5 shows a screenshot of the *RuleSet* table for Empty operation model. Each rule in the table is either a rule for cause, or a rule for consequence, distinguished by the setting of *consequence* field. Each rule consists of two main parts: the conditions, and the assessments. The fields *node_name* and *deviation* specify the deviation type on the process variable. The conditions include (1) *param_cond*: conditions on parameter, such as a specific charging method for charge operation; (2) *mat_cond*: conditions on materials or material sets; (3) *eqp_cond*: conditions on equipment properties; and (4) *OT_const*, *OP_const*, *OL_const*: real valued constants to specify thresholds on operating temperature, pressure and level. The grammars on the conditions are in the format of first order logic. For example, the material condition, *(Static && Cryogenic)(materials_in_distill)*, consists of two parts bounded by two parenthesis. The symbols and the relations between the symbols in the first parenthesis specify the material conditions. The material sets and the relations between material sets specify the materials to which this condition is applied. The above condition means that the cause or consequence becomes valid only when one or more materials, which are involve in the material set of *materials_in_distill* of this operation, posses the hazardous properties of Static *and* Cryogenic. Other valid operators on the symbols include '||' between two symbols to represent the relation of OR, and '!' in front of the symbol to specify 'not' operation. For example, '!Static' is specified to check whether there exist materials which are not static.



ID	description	node_name	deviation	param_cond	mat_cond	eqp_cond	rule_result	eqp_cond_con	OT_const	OP_const	OL_const	conseq?	recommendati	result_class	severity
71		agitation	zero		(Solid)	HasAgitator	settling solids plugging the bottom valve					☑	installation of lo	poor stirring eff	1
70		agitation	low		(Solid)	HasAgitator	settling solids plugging the bottom valve					☑	installation of lo	poor stirring eff	1
67		agitation	high		(SolidHm)	HasAgitator	tossing hazardous solids onto the vessel wall					☑	installation of hi	tossing of haza	3
69		agitation	high		(Static)	HasAgitator	potential accumulation of charge under high agitation s					☑	control of agitat	static hazard	3
117		agitation	high			HasAgitator	agitator operated at high speed	0	0	0	0	☐			
118		agitation	low			HasAgitator	agitator operated at low speed	0	0	0	0	☐			
119		agitation	zero			HasAgitator	agitator not working or not switched on	0	0	0	0	☐			
114		duration_of_operation	zero				emptying not performed	0	0	0	0	☐			
113		duration_of_operation	low				emptying performed for a shorter duration	0	0	0	0	☐			
112		duration_of_operation	high				emptying performed for a longer duration	0	0	0	0	☐			
116		empty_amt_of_mass_final	zero				outlet valve closed	0	0	0	0	☐			
115		empty_amt_of_mass_final	low				outlet valve open too small, or pipe blocked	0	0	0	0	☐			
85		Flow	Reverse				If transfer is critical, potential downtime					☑		downtime	1
87		Flow	Reverse				back flow from <toequipment>					☐			0
88		Flow	Wrong				incorrect amount of material transferred to <toequipme					☐			0
89		Flow	Wrong				If transfer is critical, can result in off-spec batches					☑		product quality	2
86		Flow	Reverse				potential overpressure/overfill of <equipment>					☑		equipment rupt	4
74		level	low		(Solid)		settling or non uniform concentration due to poor mixing					☑	monitoring of le	poor mixing	1
90		Material Handling	improper		<Risk Phrases>							☑		personnel safet	3
80		pressure	low	method = (pump)			low tank pressure is generated during liquid pumping o					☐			0
79		pressure	low	method = (vacuum transfer)			failure of vacuum control system					☐			0
78		pressure	low	method = (pressure)			failure of pressure control					☐			0
73		pressure	low		(Flammable)		potential for sucking air leading to flammable atmosph					☑	maintain inertin	fire hazard	5
77		pressure	high	method = (vacuum transfer)			failure of vacuum control system					☐			0
76		pressure	high	method = (pressure)			failure of pressure control					☐			0
75		pressure	high				line plugged					☐			0
91		transfer_line_flow_rate	low				line plug or vapor lock	0	0	0	0	☐			0
92		transfer_line_flow_rate	low				valving on transfer line fail low	0	0	0	0	☐			0
93		transfer_line_flow_rate	zero				line plug or vapor lock	0	0	0	0	☐			0
120		transfer_line_flow_rate	low				If transfer is critical, potential downtime	0	0	0	0	☑	installation of fl	downtime	1
94		transfer_line_flow_rate	zero				valving on transfer line fail low or closed	0	0	0	0	☐			0
108		transfer_line_flow_rate	high		(Static)		static flow due to high flow in transfer line					☑	installation of fl	static flow	3
122		transfer_line_flow_rate	zero				losing flow					☑		flow failure	2
95		transfer_line_flow_rate	high				high supply pressure	0	0	0	0	☐			0
121		transfer_line_flow_rate	zero				If transfer is critical, potential downtime	0	0	0	0	☑	installation of fl	downtime	1
96		transfer_line_pressure	high				line plugged	0	0	0	0	☐			0
97		transfer_line_pressure	high				normal wear and tear of transfer line	0	0	0	0	☐			0
98		transfer_line_pressure	high			Transportation	<toequipment> arrives at plant pressurized too high (p	0	0	0	0	☐			0
99		transfer_line_pressure	high		(Cryogenic)		thermal expansion of cryogenic materials (<materials>)	0	0	0	0	☐			0
109		transfer_line_pressure	high			NearDesignPre	potential overpressure of <toequipment>	0	0	0	0	☑	installation of pr	equipment rupt	4
105		transfer_line_pressure	high			Transportation	<toequipment> PSV lifts and vents materials					☑	check pressure	release of mate	2
110		transfer_line_pressure	high				potential line corrosion	0	0	0	0	☑		corrosion	2
106		transfer_line_pressure	high			NearDesignPre	overpressure line leading to potential line rupture					☑	installation of pr	equipment rupt	4
100		transfer_line_temperature	high				high ambient temperature	0	0	0	0	☐			0
101		transfer_line_temperature	low				low ambient temperature	0	0	0	0	☐			0
102		transfer_line_temperature	low				poor adiabatic conditions of transfer line	0	0	0	0	☐			0
104		transfer_line_temperature	low		<ReportMaterialC							☑			3
103		transfer_line_temperature	low		(Cryogenic)		cryogenic liquid materials (<materials>) present	0	0	0	0	☐			0

Figure 5. Structure of local causes and consequences and their storage.

Digraph representation of safety knowledge

As discussed in previous section the safety models for operation and equipment are stored as an Access database with different tables storing different parts of the model. One of the important parts in the model is the relation between process variables involved in the operation or equipment. The relations can be visually rendered as a signed directed graph (SDG), as shown in Figure 6. SDGs have been used extensively as a tool for graphically representing the qualitative causal models of chemical process systems. Digraphs of process units can be derived from the material and energy balances or from experience based on expert knowledge.

In this system, the basic symbols for the graphical representation are nodes and arcs. Nodes can then be further categorized into *dg_node, dg_vector_node, dg_cause_node*, and *dg_conseq_node*. *dg_node* represents a process variable which can be applied to all the materials, such as temperature, pressure, reaction_extent, level and so on. *dg_vector_node* represents the concentration information for one or more materials. *dg_node* and *dg_vector_node* are related to the ports and the material sets which those process variables are associated with. *dg_cause_node* and *dg_conseq_node* indicate the local cause and consequence of the specific process variables they are connected to. The nodes are specified by several properties, including (1) *standard_variable* which specifies if deviation can be applied to the process variable the node represents; (2) *manipulated_variable* to specify if the process variable is involved in a control loop and is the manipulated variable; (3) *controlled_variable* to specify whether or not the process variable is involved in control loop and is controlled; (4) *node_zeroable* to specify whether or not the deviation of *none* can be applied to the process variable. During

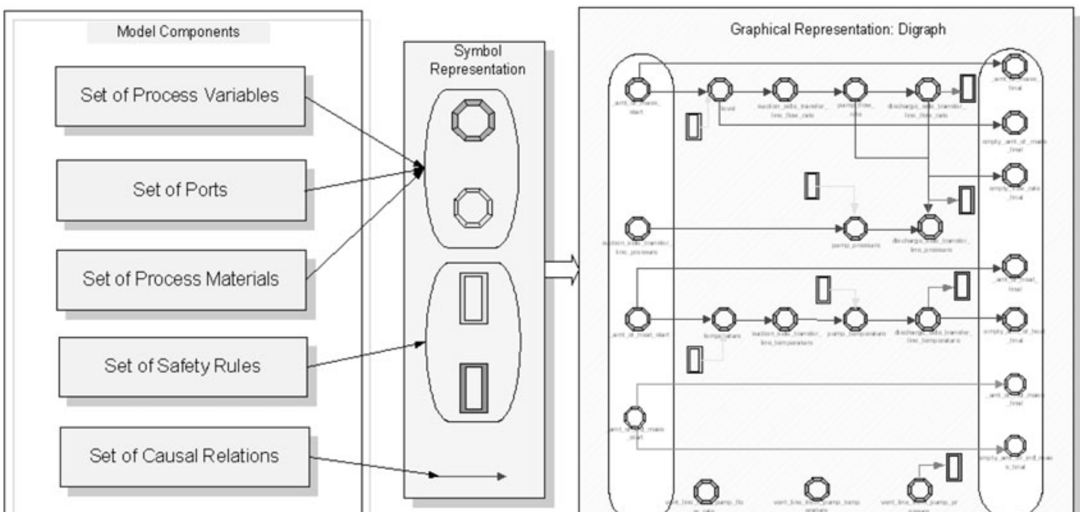


Figure 6. Digraph structure and its relation with other information.

HAZOP analysis, a deviation in a controlled variable is not propagated further unless the original deviation is from a manipulated variable. For example, in a heat subtask, temperature is a controlled variable and manipulated variables are 'jacket flow rate' and 'duration of operation'. A temperature deviation not resulting from the manipulated variables is not propagated further downstream. The consideration of control loops helps in avoiding the generation of hazards with low possibility.

The *dg_node* and *dg_vector_node* are stored in the table *DgNode. dg_cause_node* and *dg_conseq_node* are stored in the tables *DgCauseNode* and *DgConseqNode* respectively. Each record in the tables represents a node, with the properties of the node being stored in fields.

Nodes are connected by arcs. Depending on the different combinations of types of the nodes the arcs connect, the arcs can be divided into (1) *dg_arc*, which link *dg_node* with *dg_node*, or *dg_node* with *dg_cause_node* or *dg_conseq_node*; and (2) *dg_vector_arc*, which link *dg_vector_node* with *dg_node* or *dg_vector_node.* The causal relations between process variables are represented by the direction of arcs and the properties of the arcs. The simple signs are *positive* or *negative.* The positive (negative) sign on an arc represents a positive (negative) influence of the process variable represented by the source node on the process variable represented by the target node connected by the arc. The properties of the arc, *arc_highable* and *arc_lowable*, capture the conditional interactions between process variables. Deviation propagation is constrained based on conditional relationships between variables. For example, in a digraph where there is an arc connecting agitation to temperature, by setting the 'highable' property of the arc connecting the variable agitation and the variable temperature to false, a high agitation, then, cannot cause temperature going high. By using conditional deviation propagation, the accuracy of HAZOP analysis is significantly improved. In current implementation, all the arcs are stored in *DgArc* table with different fields storing the properties of the arc.

Complexity arises as not every relation can be represented as simple positive or negative relation, especially with the *dg_vector_arc.* For these relations, the gain of the arc is specified by an identifier which links to an arc-function. For example, the arc which links dg_node of *reaction_extent* and dg_vector_node *amt_of_ind_mass_final* needs to specify how the deviation on the reaction_extent can affect the concentration of the materials involved in the operation. However the effects of the reaction_extent are different for different material sets. Low reaction extent can cause low concentrations for the materials as products, but can cause high concentrations for the materials as reactant. The cause-effect relationships between the related process variables are captured in the arc gain methods mentioned above, and are stored in *DgFuncs* table. The function is identified through the identifier in the gain field in the *DgArc* table. Each record in the *DgFuncs* table represents an assessment, so an arc function may need more than one record to be represented completely. The properties for each assessment include what material set it is applied to, what is the deviation state, what is the physical state of the materials, and what the assessment is. The material set is specified by the identifier of the name of the material set, and assessment specifies if the deviation caused by the source process variable on the target process variable is the same as the original deviation or the opposite of it.

From the point of view of connectivity, a digraph consists of three node layers. The first layer is inlet layer, corresponding to inlet streams, inlet mass, heat, pressure, and flow. The second layer consists of process variables, including normal variables, such as temperature, pressure, flow rate, level, and variables for operation intent, such as separation extent and so on. The last layer is the outlet layer, including mass, heat, pressure, and flow corresponding to outlet streams. The variables are represented using *dg_node* or *dg_vector_node*, and are connected with *dg_arc*, or *dg_vector_arc. dg_cause_node* and *dg_conseq_node* are usually connected with *dg_node* or *dg_vector_node* in the second layer. The reasoning engine is constructed to be compatible with this kind of digraph structure. To incorporate the different naming conventions, marshalling and demarshalling procedures for the process variables are included in the reasoning engine and these are discussed in Reasoning Methodologies section.

Knowledge Management Using Case-Based Techniques

The techniques for representing knowledge and storing knowledge as models in database have been discussed in previous sections. Use of models is the fundamental requirement for adopting model-based reasoning techniques as will be discussed in Reasoning Methodologies section. However, the problems in the model-based reasoning lead to adaptation of the case-based techniques as knowledge management methodology in this framework.

Case-based techniques

Model-based reasoning refers to the structuring and use of general causal knowledge, usually to describe well-understood causal devices such as circuits, electrical generators, and pumps. Explicit models or deep models of the subject which are used by the knowledge based system for reasoning, are used. A basic assumption of model-based reasoning is that a single general model of a class of objects or devices is sufficient for representation. This is the way model-based programs are generally implemented. A single model is represented that includes in it every contingency for every exceptional situation that the model builder can think of (Kolodner, 1993).

However, it is very hard to use a general model for every situation and it is very hard to manage. Other problems with this approach include how to use experience and learn from experience. As is usually the case in AI research, research on reasoning using experience is based on the studies by psychologists on memory and cognitive processes of human beings. Psychological research has found that memory has a complex *structure* and *indexing* that allows people to relate a new episode to prior cases through thematic and abstract categories. Based on this, Schank (1982) proposed a theory of learning based on reminding. The psychological assumptions of memory structure and learning from experience are summarized in Slade (1991). Besides the knowledge structure for memory and storing experience, the processes involved in acquiring and accessing these structures are specified in Slade (1991).

Applying these cognitive research results to real life problems simulates the research of case-based reasoning (CBR). The central idea of CBR research is *learning through experience.* The first case-based reasoning system was CYRUS, developed by Kolodner (1983). CYRUS is basically a question-answering system with knowledge of the various travels and meetings of former US Secretary of State Cyrus Vance. Several systems, including MEDIATOR, PERSUADER, and CHEF (Riesbeck and Schank, 1989) have been developed using similar techniques. CBR has been successfully applied in several industrial applications such as customer relation management, help desk, and management of industrial knowledge (Bergmann *et al.*, 1999). Recently, researchers in chemical engineering applied CBR to solve design problems (Xia and Rao, 1999; Pajula *et al.*, 2001).

All CBR methods have similar operations, as shown in Figure 7 (Aamodt and Plaza, 1994), from the processes point of view and task structure point of view respectively. The major steps in CBR are similar to that of process flow in cognitive science research. These steps are:

(1) RETRIEVE the most similar case or cases.
(2) REUSE the information and knowledge in those cases to solve the problem.
(3) REVISE the proposed solution.
(4) RETAIN the parts of this experience likely to be useful for future problem solving.

HAZOP analysis performed by human experts depends largely on their experience. Experience plays two important roles. Firstly experience contributes to refinement and modification of reasoning process. Successful experience is solidified into causal relationships between process variables and rules for cause/consequence analysis. Experience's second role is equally important. Experience helps

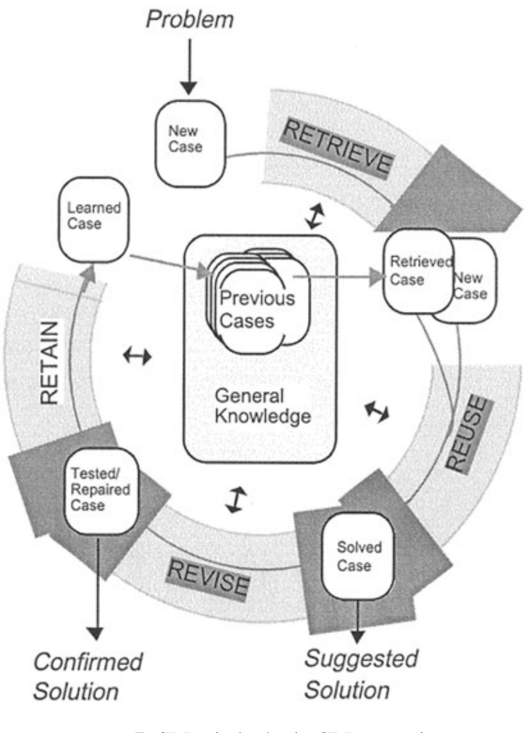


Figure 7. CBR circle: basic CBR operations.

in analysis of new processes by recalling similar situation encountered during earlier HAZOP analyses. Case-based reasoning techniques provide a formal way to organize the different kinds of experience into a formally organized knowledge base, which is easy to access, modify, and expand. In case-based reasoning, new problems are solved by retrieving and adapting solutions of similar problems encountered in the past. Once a new solution is created, it can be stored for potential reuse in future.

Representation of cases

In this framework, the *dynamic* model, which was proposed by Kolodner (1983), is adopted. The basic idea is to organize specific cases which share similar properties under a more general structure. Initial base is the starting point for constructing the case base. It consists of generic models. Specific knowledge in different processes may be added to these generic models to create new models and thus extend the case base. The models, created during the development of this system which have been tested on industrial sized processes, are used as initial case base in this system. There are about 50 operation models and 50 equipment models. These generic models are also used as the generalized episodes.

Case organization

In this framework, the features used to index operation models are:

- function type;
- subfunction;
- physical properties of the processing materials, with values liquid, gas, solid;
- processing conditions:
  - pressure, with values of high, low, normal;
  - temperature, with values of high, low, normal;
- components (equipments) and the corresponding process variables.

The types of functions for device can be divided into: ToMake, ToMaintain, ToPrevent, ToControl (Chandrasekaran, 1994). Similarly for operations, where the functionality is defined on the materials or substance, the functions may be divided into the following types: *ToMake*, such as reaction related operations; *ToSeparate*, such as separation related operations; *ToMaintain*, such as hold; *ToMove*, such as Transfer; *ToChange*, such as heat, cool; *ToClean*: clean, purge, and so on.

The equipment models are indexed through types, subtypes, and structural difference. The cases are hierarchically organized in the knowledge base. The features used include:

- type;
- subtype;
- processing material physical properties, with values of liquid, gas, solid;
- processing conditions:
  - pressure, with values of high, low, normal;
  - temperature, with values of high, low, normal;
- components and the corresponding process variables;
- *values*: no fixed values, specific for different type of equipment.

Based on the work by Walas (1988), equipments can be divided into fluid transport equipment, solid transfer equipment, heat exchangers, dryers and cooling towers, and so on.

In this framework, operation models are selected for operations, and equipment models are selected for equipments. The top two features, function and subfunction for operation or type and subtype for equipments, have been included in the specification of operation and equipment. The index structure, which is a discrimination network, has a tree structure and consists of features and their values as well as the corresponding model identifier. The tree structure is defined as containing non-leaf node and leaf node. Class of *NotLeafNode* and *LeafNode* are inherited from *TreeNode* class, which has following variables: *node_name, node_features, node_value*, *parentnode_IDs, childnode_IDs.* The whole index structure is externally saved into database and maintained by knowledge builder. The organization scheme is very flexible. Other types of index features can be easily added on, for example, if there is a need for a specific model for operation or equipment in a specific equipment and/or in a specific facility, the ID of the facility and/or the ID of the equipment can be used as index features.

Case retrieval

The case retrieval starts when a problem description is input to the case memory, and ends when a best matching case has been found. The steps in case retrieval include identifying features generating initial match, searching and selecting. The identification subtask needs to understand the problem within its context to generate a set of relevant problem descriptors. The initial matching process is to retrieve a set of relevant candidates by using problem descriptors (input features) as indexes to the case memory in a direct or indirect way. Three principal ways that are used by different systems are: following direct pointers from problem features, searching an index structure, or searching in a model of general domain knowledge. A best match is chosen from the set of similar cases. This may be done by using the system's own model of general domain knowledge, and/or by asking the user for confirmation and additional information.

Case retrieval is one of the main steps when selecting models from knowledge base. The task includes selection of candidate models and ordering of candidate models based on their similarity to the current situation. The process is first assessed and the features that are used to index the models are determined. For example, charge operation needs an operation model with function ToCharge, and the things to be charged are amount of materials. Hence the generic operation 'add material' is located. The specific models are determined by process information. Here the index feature is physical state of material added. So the process information about the material added is assessed. If the material added contains solid material, load model is selected. This process is shown in Figure 8.

If the functional or structure-behaviour specification of the desired case and a stored case match at least partially, then the stored case is judged as potentially useful and is selected as a candidate case. For example, consider a model to be selected for tank, whose component list contains agitator. However, after searching through the

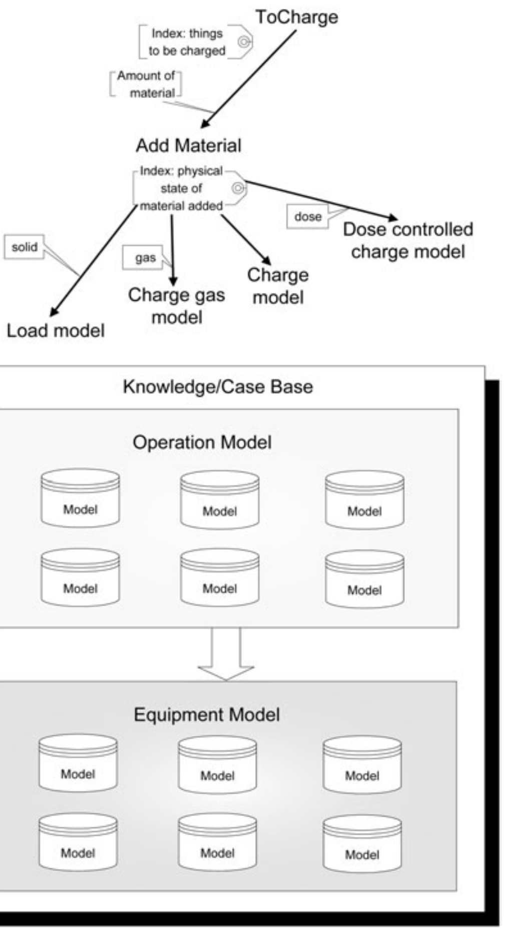


Figure 8. Retrieve a case from case base according to process conditions.

knowledge base, there is no index feature for component agitator. The closest model, which is model for a tank without agitator is selected as candidate.

Case reuse and adaptation

The simplest case reuse strategy is to transfer the retrieved case to the new case as its solution. When the retrieved case cannot be directly transferred to the new case, the retrieved case must be adapted. Two main techniques for case adaptation are: corrective modifications and compensatory modifications. By the location of modifications, the adaptation can be divided into parametric, componential and topological modification.

Retrieved operation and equipment models are analysed to see if they are suitable for the present situation. Appropriate modification methods are applied to those models if modification is necessary before they are used.

The modifications which can be made to the model include:

- process variables: add, delete;
- causal relations: modify, delete, add;
- rules for cause/consequences: modify, delete, add;
- rules for digraph propagation: modify, add.

Case modification starts after the easiest to adapt candidate model is selected from the ordered set of candidate models. Though this system supplies aids to this modification task, user interaction is heavily involved, which can be called as 'on-site model modification'. The aids supported by this system include: commonsense knowledge, and rule-guided repair. Considering the tank example described in last section, when a new component is present in the tank, commonsense knowledge tells that a new process variable should be added to the model to represent the effect of this new component. In the tank example, a process variable *agitation_speed* is added to the model. The relations between this new added variable and other variables should be decided. The system searches the knowledge base to see if there is index for component agitator and finds the corresponding model. Analysis is then carried out to find out the part of the model which gives information which may help in the construction of relations between *agitation_speed* and other variables. Similarly, this approach helps modifications of other parts of the model. This corresponds to recalling the experience of only those parts which are useful for the new situation. In the implementation of this module, knowledge builder will be called up for the 'on-site model construction'. Further research on this topic will help in reducing the involvement of user.

Learning

The knowledge or experience learned from solving new problem should be incorporated into the existing knowledge. The learning process in CBR, which is case retainment, involves selecting the appropriate information from the case to retain, the form to retain it, indexing the case for later retrieval for similar problems, and integrating the new case in the memory structure. Thus, a CBR system may incrementally extend and refine its general knowledge model as well as its memory of past cases, in the course of problem solving.

While examining the HAZOP analysis results, user may add, delete or modify the results, which acts as a feedback. User needs to specify the location of the deviation, the deviation that relates to the cause/consequence, and location and description of the causes/consequences. The user input may be shallow knowledge. To acquire and reuse this kind of knowledge, corresponding causal paths need to be created. Three situations may be encountered:

- If there is a causal path and it can represent the logic, the new local causes or consequences are the only changes need to be made.
- If there is no such causal path, user will be presented with the paths that are present in the same model, and suggested sign of the path from those variables to the consequence deviation. User then needs to specify the path which needs to be added.
- If the path exists, but the cumulative sign of the causal path is not consistent with the newly added result, the inconsistency needs to be resolved.

Finally, the case-base is modified or expanded depending on whether the change is generally applicable or only applicable to the current situation.

Knowledge Acquisition

Knowledge acquisition is a very important step in constructing a knowledge system. The purpose of knowledge acquisition is to identify the relevant technical knowledge, record it, and translate it to a proper knowledge representation form for the purpose of understanding, supporting or automating complex problem solving behaviour.

Currently, interviewing a domain expert with questions regarding the domain of interest is still the primary technique for knowledge acquisition. The basic model for knowledge acquisition has been that the knowledge engineer mediates between the expert and knowledge base, eliciting knowledge from the domain expert, encoding it for knowledge base, and refining it in collaboration with the expert to achieve acceptable performance. Based on the involvement of knowledge engineer in the knowledge acquisition process, the major approaches for knowledge acquisition can be divided into two categories: the model-based incremental knowledge engineering approach (MIKE) (Angele *et al.*, 1993), and the configurable role-limiting method approach (CRLM) (Poeck and Gappa, 1993). MIKE is strongly influenced by work in the domain of software engineering and information system design. It is based on the distinction between different phases like analysis, design, and implementation, similar to the software development process. On the other hand, CRLM is based on implementations of strong problem-solving methods and greatly simplifies knowledge acquisition through guidance by the fixed knowledge representation scheme. In the process model of CRLM, since the predefined knowledge representation formalism is used to guide the acquisition of domain knowledge, the domain expert only has to instantiate given generic concepts with the support of sophisticated graphical user interface. The knowledge acquisition tools allow domain experts to develop the knowledge base by themselves without further guidance by a knowledge engineer, after a training phase.

The knowledge in PHASuite was initially gathered manually based on the knowledge about the chemical processes of developers (or knowledge engineers). The knowledge base was then modified and improved with experience gained from analysing more than a dozen real industrial processes. Since it is necessary for the knowledge base of this system to be expanded and modified by domain experts themselves, the CRLM knowledge acquisition model is adopted. Knowledge Builder is designed and implemented as the knowledge acquisition tool.

Unified knowledge base

As discussed earlier in this section, knowledge base is a very important if not the most important component in this system. A suitable knowledge base is critical for the success of the system. Knowledge is present ubiquitously in HAZOP analysis, and different knowledge is related with each other.

At the level of equipment and operation, knowledge is present in the specifications, the structural information, the behaviour of the equipments and operations, as well as the causal relationship between process variables. The different knowledge components interact with each other as illustrated in Figure 9. For example, the structural information about equipment affects the definition of process variables, thus affecting the causal relationship between process variables. The design and implementation of the knowledge base for automated HAZOP analysis are very

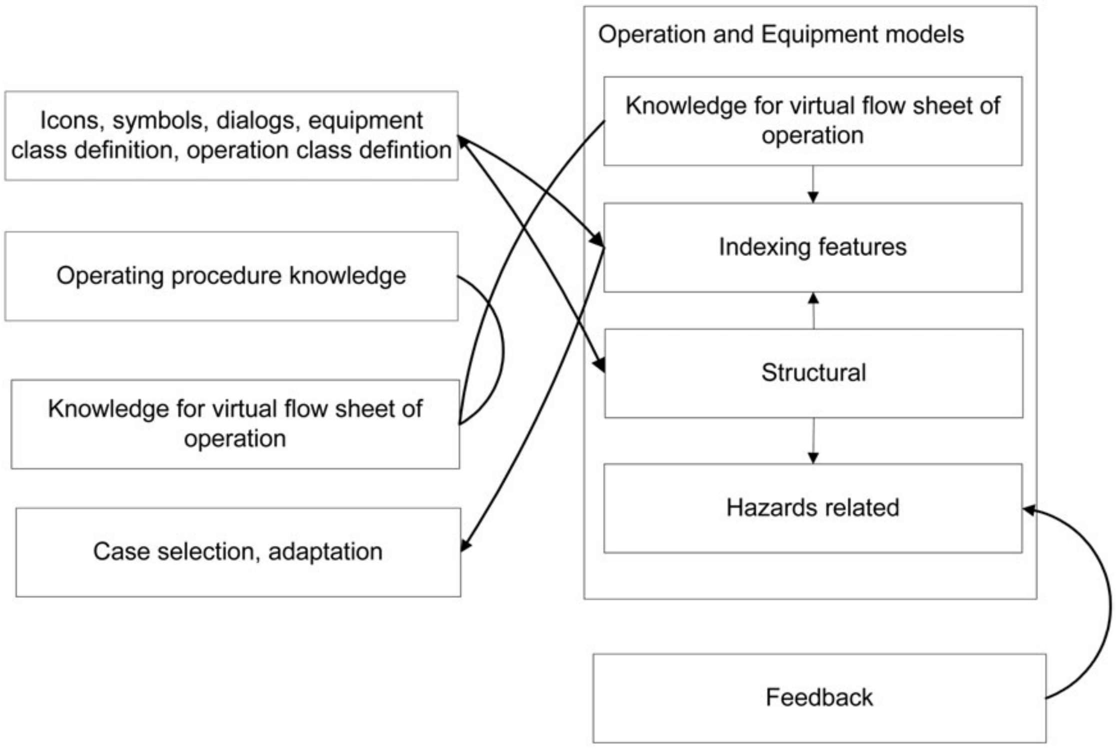

Operation and Equipment models
Icons, symbols, dialogs, equipment class definition, operation class defintion
Knowledge for virtual flow sheet of operation
Indexing features
Operating procedure knowledge
Structural
Knowledge for virtual flow sheet of operation
Hazards related
Case selection, adaptation
Feedback

Figure 9. Interaction between different parts of knowledge base.

complex tasks. The basic elements of the knowledge base are the models for equipment and operation. Different layers of the model represent the different views/aspects of the model. For example, the topmost layer, i.e., the interface layer, interacts with user through dialogs and tables for gathering specifications and structural information about the model. The different layers of knowledge are connected with each other. The index features used in CBR include types (functional intent) of the equipment or operation, subtypes, physical properties of the process materials, process conditions, and components and so on.

Due to the complex and inter related nature of the knowledge contained in the knowledge base, it is very hard for knowledge engineer and the users to work directly on the knowledge base, and even when it is possible, it is very hard to ensure the correctness and the consistency of the knowledge base. Hence to simplify the knowledge modification process, it is desirable to create a software tool as an interface between the users and the knowledge base. It is also necessary to have a sophisticated knowledge acquisition tool to supplement the role-limiting knowledge acquisition methodology. Knowledge Builder, which contains all the facilities for knowledge acquisition and management, has been constructed based on the structural and layered design, as an important supporting system for PHASuite.

The overall knowledge structure can be divided into several layers. The bottom layer consists of the properties of material, equipment and operations. For example, the hazardous properties of materials, such as corrosive, cryogenic, flammable and so on, for equipment, the properties of design temperature, design pressure, volume of the vessel, and material of construction and so on, and constants such as EquipmentPressureMargin, FlashPointTemperatureMargin etc. are included in bottom layers. Based on this, several criteria could be defined, such as for material condition of *LiquidFlammable*, several lower level primitives are combined to achieve this condition, i.e., the material should be in the liquid, and should have flammable nature, i.e., *(physical_state = 'Liquid' && flammable_nature)*. For equipment, equipment conditions are generated from basic equipment properties. For example, NearDesignPressure is defined as (operating pressure)/DesignPressure > Pressure_Margin. The local cause/consequence rules represent a higher level above the condition level, in that the conditions to define the rules are material, parameter, and equipment conditions defined in the lower level. This layered structure of the knowledge is adopted in the design of the knowledge storage, and the Knowledge Builder.

Knowledge Builder

As described above, the main functionality of the Knowledge Builder is to act as an interface for the end user to access and modify the knowledge base. All its operations are based on the storage of the knowledge base, which in this system is the database. So the main components of the software include (as shown in Figure 10):

- Layer to read and write to the database of the knowledge storage.
- An easy to use GUI for user to work with.
- Presentation of the knowledge content through the GUI.

The lowest layer contains the facilities for reading from the knowledge base, writing to the knowledge base, and

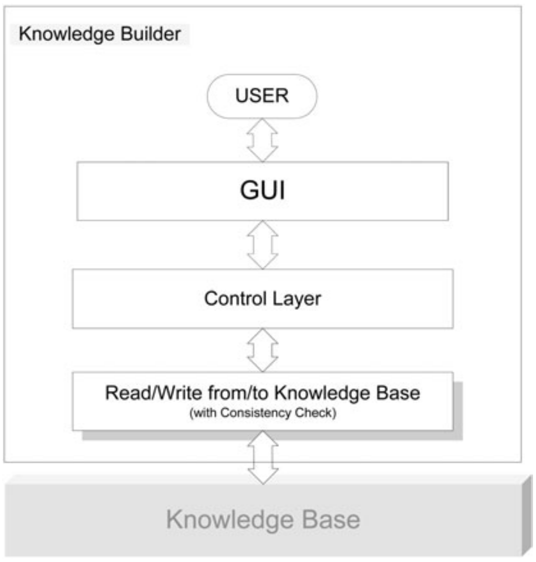
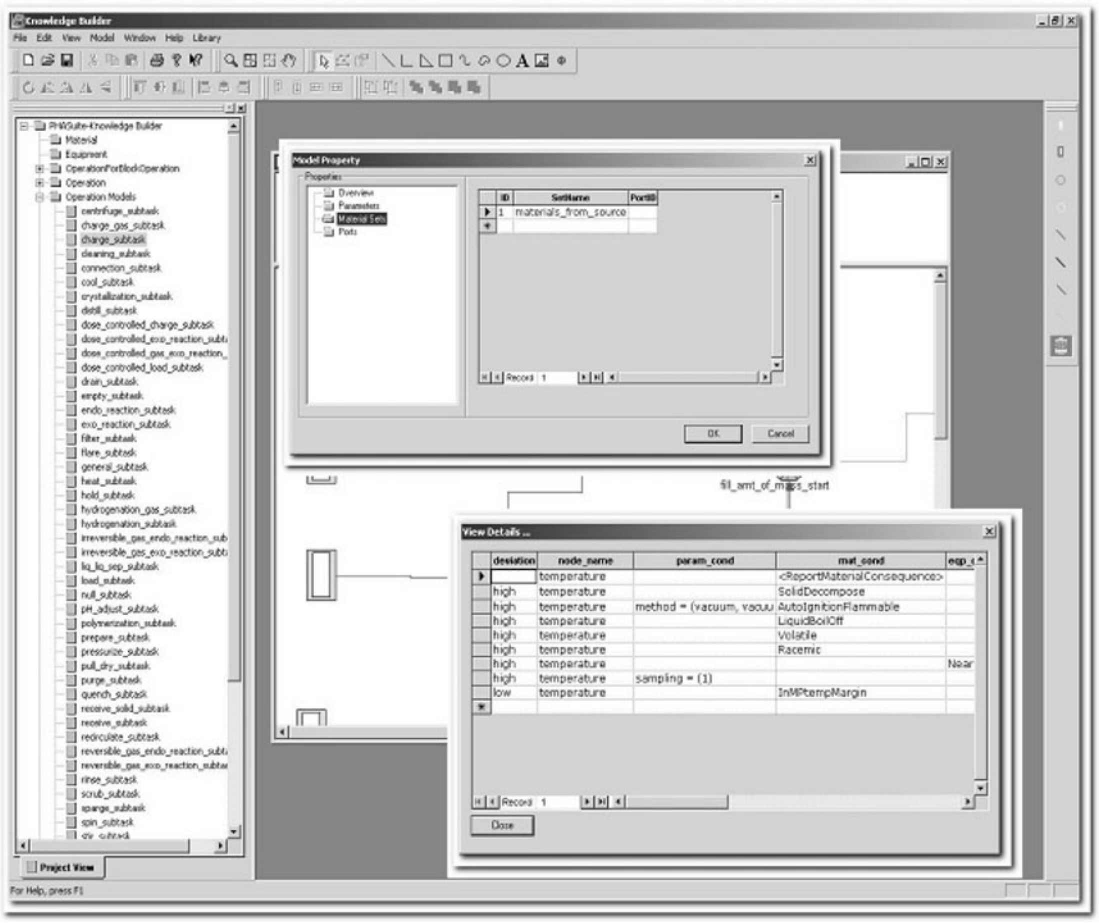


Figure 10. Overall architecture of the Knowledge Builder.

consistency checking during the reading and writing operations. The major part of it is SQL queries. There are two types of SQL queries in this system, based on their storage location. Some queries are stored with the knowledge base, and accessed by the knowledge builder by their name. Although the name is fixed, the details on the queries can be modified as long as the required output information is assured. This type of queries gives more flexibility. The other queries are embedded in the source code of the Knowledge Builder. The source code must be modified, and the program must be recompiled, before the modification can be put in effect. The queries which are closely related to the structure of the knowledge base and rarely changed are of this type.

As shown in Figure 11, the main window of the Knowledge Builder is divided into two parts. The left hand side shows the organization of the knowledge base using a tree type view. The knowledge is categorized into Material, Equipment, OperationForBlockOperation, Operation, Operation Models, and Equipment Models. On expanding a tree node, the corresponding individual models are listed. Different type of knowledge has different representation schemes, and all can be displayed in the workspace on right hand side of the tree type view. The view of the knowledge components can be divided into two types. The knowledge with no need for graphical representation is presented in worksheet like tables in a FormView or a Dialog. The different parts of the knowledge are further organized and accessed through a tree view in the dialog. For the models of operations and equipments, it is best to graphically display the causal relations between process variables. To make the navigation easier, the digraph nodes are categorized by their type and shown in a tree type view in the upper frame of the digraph. The tools



ID	SetName	PortID
1	materials_from_source	

deviation	node_name	param_cond	mat_cond	eqp_c
	temperature		<ReportMaterialConsequence>	
high	temperature		SolidDecompose	
high	temperature	method = (vacuum, vacuu	AutoIgnitionFlammable	
high	temperature		LiquidBoilOff	
high	temperature		Volatile	
high	temperature		Racemic	
high	temperature			Near
high	temperature	sampling = (1)		
low	temperature		InMPtempMargin	

Figure 11. Main GUI components of Knowledge Builder.

for creating or modifying a digraph are placed in a palette toolbar when the digraph is shown. The relations between the different knowledge components are also shown and created graphically.

The control layer determines where to access the knowledge components, and how to display them to user. The operation models and equipment models are stored individually. The other knowledge components including the location and directory information for the operation and equipment models are all stored in another database file. For example, on double clicking the 'charge_subtask' under the 'Operation Models' in the left hand side tree view, the control layer first looks into a specified table for the file location of the model from the central database, and then renders the digraph from the operation model storage.

Although the current implementation of Knowledge Builder has all the basic functionalities necessary for accessing and modifying the knowledge base, it is nonetheless still a prototype version. Further testing and implementation are needed to make it a full fledged and deployable system.

REASONING METHODOLOGIES

After the process information is gathered manually or imported automatically using the approaches discussed in Ontology Based Information Sharing section, reasoning methodologies are then applied to analyse on the chemical process to perform safety analysis and generate results using the knowledge base constructed using techniques presented in Knowledge Representation, Storage, Management and Acquisition section. Before the automated analysis can be carried out by the software program, it is necessary to have a proper representation of the process. Petri Nets are used as a representation for HAZOP analysis in this system. Operation-centred analysis and equipment-centred analysis have been integrated using function representation and are discussed in this section. A two-level, two-layer reasoning engine has been developed and is also discussed.

Representation of Process

For an automated HAZOP analysis tool to perform reasoning on chemical processes, the process information needs to be transferred to a representation that is suitable for such a task. The representation provides an abstraction layer above the underlying process descriptions. The reasoning engine works on the representation layer rather than the process description layer.

Batch chemical processes can be naturally modeled as discrete event systems (Srinivasan & Venkatasubramanian, 1998). The event is the execution of operation for the execution of a batch sequence. For continuous processes, researchers have adopted state transition diagram (Kinoshita *et al.*, 1981) to represent the sequence of and relationships between equipments (units). Furthermore, the procedure to perform HAZOP analysis by human experts on a chemical process can be characterized as a discrete event system. Equipment and operation are basic elements of analysis. Analysis on a basic element starts when all the streams input to that element have been analysed. Process variables in different elements interact through the conditions of streams. When a deviation is introduced in a piece of equipment or operation, the effects of the deviation on other process variables in the same elements are analysed first, and this is referred to as the local propagation. Conditions of the streams at the outlet of the element are then determined. The analysis then progresses to next equipment or operation downstream during consequence analysis. Petri Nets, which is a widely used modelling tool for discrete event systems, has been adopted to represent the process to be analysed by the reasoning engine of PHASuite.

Petri Nets are bipartite directed graph with two kinds of nodes that are defined as places and transitions. Places and transitions are connected with arcs. Tokens that reside in the places indicate the net's state. The execution of Petri Nets is controlled by the number and distribution of the tokens. More discussion about Petri Nets can be found in Peterson (1981). When a Petri Nets graph is used to model a physical system, events, which change the state of a physical system, are explicitly modeled by transitions. In a simple Petri Nets graph, the input places form preconditions of an event to a corresponding transition that is connected by an arc from the input place to the transition. Accordingly, the output places are the post conditions of an event for the corresponding transition that is connected by an arc to the place.

The ordinary Petri Nets are suitable to model manufacturing processes and have several applications in that area. For use in HAZOP analysis for chemical processes, however, a more powerful modelling language is needed. The reason is that tokens need to be differentiated, such as the material propagation token is different from the consequence analysis token. Also tokens need to be associated with complex data structure, e.g., the deviation propagation token represents a data structure for the states of the process. As an extension of the ordinary Petri Nets, Coloured Petri Nets, which are suitable for the above purpose are adopted in this framework.

Coloured Petri Nets (CPNets) are more powerful modelling tool created by extending the structure of ordinary Petri Nets (Jensen, 1996). The relationship between CPNets and ordinary Petri Nets is analogous to the relationship between high-level programming languages and assembly code. Coloured Petri Nets have got their name because they allow the use of tokens that carry data values and can hence be distinguished from each other, in contrast to the tokens of low-level Petri Nets, which are drawn as black, 'uncoloured' dots. Tokens can also carry a complex data structure. More detailed information on CPNets can be found in Jensen (1996). In this framework, Coloured Petri Nets are used to represent the chemical processes for HAZOP analysis.

Petri Nets representation for HAZOP analysis

The way human experts perform HAZOP analysis of a process is modelled by Coloured Petri Nets. Analysis of a basic analysis node starts when all the streams input to that node have been dealt with. Transition to next analysis node takes place after finishing analysis of one equipment or operation. The token contains information about the current location of the analysis. During consequence analysis token is propagated downstream, and during cause analysis, token is propagated upstream.

Petri Nets representation for HAZOP analysis of batch processes

The execution of a batch sequence can be modelled by Coloured Petri Nets. The event here is the execution of operation. The token contains the information about the progress of the process. We also define *recipe_transition* and *pn_transition.* According to the ISA-S88.01 standards for batch control (Instruments Society of America, 1995), *recipe_transition* represents the unit operations, and *pn_transition* represents the operation. *pn_transition* are also used to represent the equipments.

Petri Nets representation for HAZOP analysis of continuous processes

During HAZOP analysis of continuous processes, equipment is the basic element of analysis. The process variables in different equipments interact through the conditions of streams. During HAZOP analysis, when a deviation is introduced in a piece of equipment, the effects of the deviation on other process variables are first analysed. After this local propagation is performed, the conditions of the streams at the outlet of the equipment are determined. The analysis then progresses to next equipment downstream during consequence analysis. The token here contains information about the location of the effect of the deviation. Operation here is defined upon one or more main equipments and their auxiliary equipments that carry out a functionality. Each analysis node is defined as a *pn_transition.* The whole process or the combination of several analysis nodes is defined as a *recipe_transition.*

In a summary, the proposed Coloured Petri Nets representation contains the following parts:

- *Transition:* represents a basic analysis element, i.e., a piece of equipment or operation which can change the process states when it is fired.
- *Place:* contains coloured tokens to represent the states of the process itself and the HAZOP analysis status before the transition connected to it is fired.
- *Arc:* represent a real flow, such as the pipe in continuous process or information flow in batch process.
- *Transition function:* contains the logic about what condition must be satisfied in order to fire a transition, and how to change the place states after it is fired.
- *Tokens:* represent the process states and the HAZOP analysis status. They are used to control the propagation in the Petri Nets.
- *Arc functions:* determine the process states after a transition connected from it is fired jointly with the transition function. Here, we combine the arc functions into the transition for simplicity.

In this work, several coloured tokens have been used. Material propagation token carries process stream status on temperature, pressure, level, and composition of material. The consequence token and cause token carry information about current analysis location and the effects of/on upstream/downstream analysis nodes from the current node. As will be shown later in this section, use of same representation scheme for both continuous and batch processes based on the HAZOP methodology lays the foundation for integrated operation-centred and equipment-centred analysis, and further enables one single system to analyse both continuous and batch processes.

Integrated Operation-Centred and Equipment-Centred Analysis

The major activity in HAZOP analysis is causal and qualitative reasoning. The key to a successful automated HAZOP analysis depends on the reasoning capability of the system. Applying causal and qualitative reasoning into device diagnostic is a very active research area since 1980s in AI community. Chemical engineers have also applied these techniques on chemical processes, especially for diagnosis of chemical processes. Due to its close relation with the reasoning applied in HAZOP analysis, research on diagnostic reasoning is reviewed and summarized in this section.

Diagnostic reasoning

The first generation expert systems, which performed diagnosis in different disciplines, used rules to associate the symptoms with malfunctions. The limited success of those systems stimulated the discussion on 'deep' versus 'shallow' representation of knowledge for diagnosis, starting around 1983. Rules are 'shallow' representation since they provide no indication of how the malfunctions caused the symptoms. In contrast, deep representations, included in a model, provide explanations for the associations. Since then, model-based reasoning has become an important area in AI research.

The causal models, which consist of deep knowledge, are divided into two types. Weiss *et al.* (1978) proposed causal nets as a model-based method for computer-aided medical decision-making. In causal nets, a network of causal relations between device variables is used to describe the working of the device. As Chandrasekaran (1994) pointed out, the causal nets theories do not propose explicit criteria for the levels of abstraction that should be used for the variables, or for organizing the causal network.

Another stream of idea is to use qualitative device models to describe device (or physical systems). Based on their representation scheme, these qualitative models can be divided into three main types: componential, process, and state variable relations.

The componential approach was proposed by de Kleer and Brown (1984). The components are specified as satisfying certain input-output relations. The components and their relations are defined as structure of the device. The device can be simulated, i.e., the values of all the variables can be derived using knowledge about the components' input–output properties and about the component interconnections, which is the behaviour of the device. Davis (1984) described a system that reasons using knowledge of structure and behaviour. Among chemical engineering applications, MODEX and MODEX 2 followed this approach, where diagnostic tree is generated based on physical links between various equipments, from causal models that capture processing unit's function and structure (Rich and Venkatasubramanian, 1987, 1988).

Forbus (1984) characterized the events that can cause changes in objects over time, such as moving, heating up, cooling down, compressing and so on, as processes. He introduced the idea of modelling physical systems as a set of processes. Qualitative process theory (QPT) as developed by Forbus defines a simple notion of physical process that appears useful as a language for writing dynamical theories. Reasoning about processes also motivated the concept of quantity space, which is used as qualitative representation for quantity in terms of inequalities.

Abstraction of process: functional representation

The device or physical systems can be abstracted into different levels of abstraction. For example, an electronic amplifier is composed of different components including transistors, resistors, capacitors and so on. Each component behaviour can be described in terms of their currents and voltages. Using the techniques described above, a description of device behaviour can be generated in terms of currents and voltages. However, the behaviour of the device as a whole, which is an amplifier, is also of interest. And it is a higher level description. Sembugamoorthy and Chandrasekaran (1986) proposed the functional representation (FR) framework, for representing goal-directed, flexible reasoning that bridges the device-level behaviour with the descriptions at the component level.

In FR, the function of the overall device is described first and the behaviour of each component is then described in terms of its contribution to the function. In the FR framework, the structure is represented by listing the component names and their functions and indicates how the components are put together to make the device, i.e., describe the relations between the components. The basic idea in describing how a device achieves its function is that of a *causal process description* (CPD). A CPD can be thought of as a directed graph whose nodes are predicates about the states of the device, and the links are the causal transitions. The explanatory annotations include By-CPD, By-Function-Of and By-Domain-Law, and so on.

The FR framework has been applied to solve different problems. Sticklen and his group (Pegah *et al.*, 1993) applied FR to simulation. They describe a simulation of the fuel transport system of an F-18 aircraft using this technique. They also (Sticklen *et al.*, 1991) describe integrating qualitative and quantitative simulations in a FR framework. The FR scheme helps the simulation systems focus on those specific quantitative equations which are to be used in the actual computation. Goel and Stroulia (1996) describe a functional model called structure-behaviour-function (SBF) to describe a device and applied it for adaptive design. FR is also applied in representing problem solvers, and representation of scientific theories. A summary of FR application can be found in Chandrasekaran (1994). The work on creating generic device libraries, where device classes at different levels of system description are represented along with parameterized structural representation and the corresponding CPD, can be found in Chandrasekaran *et al.* (1998).

Current HAZOP analysis can be divided into two types, operation-centred analysis and equipment-centered analysis, based on the emphasis of the analysis and basic analysis node. Equipment-centred analysis is mainly applied to continuous processes, where the analysis progresses from upstream to downstream, starting from a deviation in a piece of equipment. However there is no need to consider each line and every single minor item of equipment, instead, smaller items are grouped into logical units according to their functionality. Operation-centred analysis is mainly applied to batch processes, since operational sequence of steps is important for batch processes. Deviation starts from one operation and is propagated along the operational sequence.

In fact, from functional representation point of view, operation and equipment are two different abstraction levels of the process. Compared to the device and its components, operation can be analogous to device and the equipments used to carry out the operation can be analogous to components. The terminology of operation is used to describe a function carried out by one or more equipments. Equipments are real entities, while operations are not. For example, the function or the goal of the operation 'TransferWithPump' is to transfer materials (substances) from one place (source) to another place (sink) using a pump through several pipelines. From this description, the equipments used in this operation can be identified to include a pump, a source, a sink, pipes between those equipments, valves on pipe, controllers and so on. Since operation emphasizes on functionality, it is possible that the functionality is carried out by different kinds of equipments. As an example, a centrifuge pump may be used in the operation of 'TransferWithPump' to transfer normal materials, but cryogenic pump is needed when transferring cryogenic materials. If abstraction of the different level of the process is not explicitly presented, different operations must be defined when any of its components are different. For example, we will have operations for TransferWithCentrifugePump and TransferWithCryogenicPump. The complexity of the combinations in that case will be exponential.

So the operation and equipment represent the different levels of the process. Functional representation is adopted as the methodology to bridge the gap between these two levels. Operation-centred analysis emphasizes on overall functionality, and overall analysis for the equipments carrying out the operation as a group. Equipment-centred analysis emphasizes on hazard analysis at equipment level. Decomposing the process representation to two abstraction levels based on functional representation gives us the capability to describe the process in more detail, and thus produces better results. Operation-centred analysis supplies the hazard results as well as operability results, which are hard to achieve without the abstraction. By connecting the operation and equipment level, more detailed results can be obtained, which are hard to achieve through only the operation level analysis.

Basic design

Operation model emphasizes on functionality, and is created according to functional representation. Equipment model emphasizes on relationships between the local process variables. For example, the functional representation for TransferWithPump consists of:

- *Intended function:* transfer materials {m1, m2, m3} from Source A to Sink B, using pump; pump can be centrifuge pump, or cryogenic pump and so on.

- *Components:* source, sink, pump, and transfer line.
- *Causal process description:* SDG to model the causal relations between process variables and the intended functions in operation TransferWithPump as well as in the components of this operation.

Operation model includes variables from the equipments to carry out the intended functions, the variables specifically related to the functionality and the variables not belonging to any equipment, e.g., *duration_of_operation.* The variables in both the operation and equipment model establish connections between the different levels of abstraction. During analysis, propagation is carried out in both levels.

Establishing relationships between operation and equipment

In order to reason on the two different abstract levels of process, connection needs to be established between operations and the equipments that are used to carry out the operation. Process information and miscellaneous information about a project, such as the name of the results file, are stored in a central repository. The process information consists of (1) chemistry and material involved in the process; (2) the block operations; (3) the process sequence diagram; (4) the process flow diagram, and (5) the array of equipments and the connections between equipments.

The PSD is generated in the following steps: (1) initialization by clearing the existing equipment relations and operations; (2) generating operations in InterOp-1 level for each operation specified in the process information; (3) after the generation, operation numbers are set, and the relations are specified; (4) a linear programming solver is then executed to optimize the graphic representation of the PSD. The algorithm for generating PFD from P&ID is carried out in three steps: simplification, identification and specification.

The detailed operations are collected in process sequence diagram (PSD). Some equipment information are specified during specification of PSD, such as the filtrate collection vessel and vessel for filtration operation. So PSD contains the detailed operations and the relations between the operations, which may be parent–child relation, or sibling relations. P&ID can be input using P&ID editing facility within the system, or imported from other information sources. P&ID can either be directly used, or simplified to PFD using the generating tool supported by the system. The major part of generating PFD is to identify major equipments and major pipelines connecting the major equipments. So PFD contains all the equipments and the connections between those equipments. The procedures for generating PSD and PFD are summarized in a diagram shown in Figure 12.

Each *Operation* object contains a FlowSheetKnowledge object which is an object of FlowSheet. The FlowSheet-Knowledge object acts as the connection between the PSD and PFD. The connections between operation level and equipment level are established between PSD and PFD. A variable SITUATION, with possible values PSD, PFD, and PSDPFD, is determined by the information available for analysis. For example, for usual batch processes, PSD may be the only available information source, and

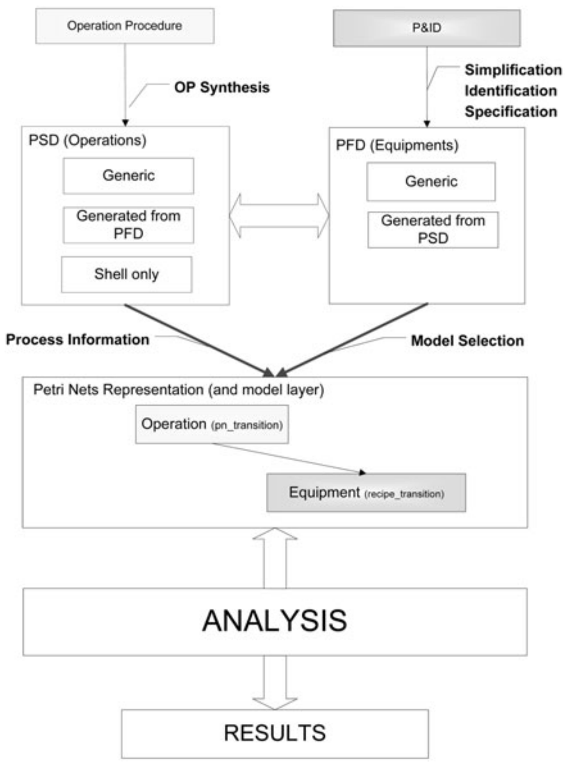


Figure 12. Layered architecture.

SITUATION is specified to be PSD. The value of SITUATION determines the possible analysis mode, which is specified in ANALYSISMODE.

Table 1 lists the possible analysis mode for each kind of situation, as well as the tasks needed to connect these two levels. The first part of the value corresponds to the situation to which the analysis mode can be applied, and the second part of the value specifies the levels at which the analysis is performed on. The following methods are available in PHASuite:

- GeneratePFDFromPSD: if no PFD information is available for analysis, but two levels analysis needs to be carried out, a virtual PFD needs to be created based on the knowledge stored in each operation.
- GenerateEmtpyPSDforPFD: if no PSD information is available for analysis, an empty shell of PSD needs to be generated so the analysis can be performed.
- GeneratePSDfromPFD: if no PSD information is available, but analysis is required to be performed on two levels, a virtual PSD is generated by appropriately aggregating equipments in PFD.
- PatternMatching: if both PSD and PFD are available for analysis, and analysis is required on both layers, the connection between PSD and PFD are created by matching the knowledge stored in each operation in PSD to PFD.

Different methods in OPS engine are called to connect PSD and PFD depending on the different combinations of the values of SITUATION and ANALYSISMODE.

Creation of Petri Nets representation for PSD and PFD

After the appropriate connection is established between PSD and PFD, next task is to create the Petri Nets

Table 1. Situation assessment and analysis mode.

Analysis mode	Situation		
	PSD	PFD	PSDPFD
PSD_OP	No extra functions needed		
PSD_OPEQ	GeneratePDFfromPSD()		
PFD_EQ		GenerateEmptyPSDforPFD()	
PFD_OPEQ		GeneratePSDfromPFD()	
PSDPFD_OP			No extra functions needed
PSDPFD_EQ			No extra functions needed
PSDPFD_OPEQ			PatternMatching()

representation, based on the generated PSD and PFD as well as the connections between them, as illustrated in Figure 13. This task includes following subtasks: (1) creating recipe_transition, pn_transition for Petri Nets layer; (2) transferring information from PSD and PFD to Petri Nets; (3) selecting models for operation and equipment. The outcome of this task is a two level representation for the process, and each level contains two layers, which are the Petri Nets layer and safety model layer. The model selection task basically involves use of process information to find appropriate models for operation and equipment for current process, from knowledge base managed using case-based techniques as discussed in Knowledge Representation, Storage, Management and Acquisition section.

The creation of Petri Nets representation results in generation of an information repository for automated safety analysis. The main methods are divided as (1) higher level methods to control the creation; (2) methods to connect PSD and PFD; (3) methods to create Petri Nets representation based on PSD; and (4) methods to create Petri Nets representation based on PFD. After proper initialization, material information, equipment information, and chemistry information are transferred to the central information repository. The recipe_transition are then created for corresponding higher-level operation, including block operation for operation level, and aggregation of equipments for certain operations for equipment level. *pn_transition* is created for each operation in PSD and each equipment in PFD. The main steps include (1) generating InterOp-2 level operation if necessary; (2) selecting proper model for each operation; (3) creating *pn_transiton* by generating model object; and (4) specifying the model properties. *pn_transition*s are connected with *pn_place*s in between. ID of the *recipe_transition* in equipment level is associated with a *pn_transition* in operation level, based on the connections between PSD and PFD.

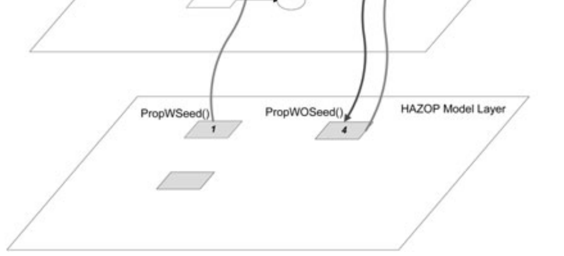


Figure 13. Analysis on coloured Petri Nets layer and safety model layer.

The HazopModel is composed of objects dg_node, dg_vector_node, dg_cause_node, dg_conseq_node, dg_arc, dg_vector_arc, ModelPort, and MaterialSet. The model objects are constructed during run-time from the corresponding knowledge stored in Access database.

Connecting Petri Nets according to process connectivity information

After pn_transition are created for each equipment and operation, they are then connected based on the process connectivity information available in either PSD or PFD. Each *pn_transition* has several (at least two) connection points which can be used to connect with other *pn_transitions.* These connection points are identified using integral *PortID.* The ports are divided into input ports and output ports. The Ids for input ports occupy the range of [1, 100] and Ids for output ports occupy the range of [101, 200]. Two *pn_transitions* are connected through a *pn_place.* Each *pn_place* has one and only one input port and output port for connecting to input *pn_transition* and output *pn_transition.* Each *pn_place* stores a pointer to the *pn_transition*, as well as the PortID of port in the pn_transition it connects to.

The ports information is part of the model for operations and equipments. Normally for batch processes, most of the operations have only one input port and one output port, for example, cool, heat, evacuate and so on. Some operations, especially the separation related operations may have more than one output ports. For example, distillation operation has an output port for bottom materials, and another output port for distilled materials. The data structure for ports also contains information to guide the material propagation.

To summarize, all the steps involved in the process for creating the Petri Nets representation of the process for analysis can be represented by a layered architecture, as shown in Figure 12. As a first step, process information is input to the system either manually or is imported automatically from other available electronic formats. There are two main parts of information: operating procedures and P&ID. Operating procedure synthesis is applied to generate PSD from operating procedures, while PFD is generated through the process of simplification, identification, and specification from P&ID. Different methods are invoked to connect the PSD and PFD level according to the general knowledge stored in the operation, the availability of process information and the user specification. Then the Petri Nets representations are created from PSD and PFD, and are connected based on the relation between PSD and PFD. Beneath the Petri Nets representation, safety models are selected for each operation and equipment from the knowledge base. Models are constructed from the model storage in Access database, and initialized according to the process information.

Two-Level, Two Layer Reasoning Engine

The procedures for creating Petri Nets representation create two representation levels for the process, namely operation level and equipment level. Each Petri Nets representation level is further divided into Petri Nets layer and safety model layer. The functionalities required for analysis on the two layers are same regardless of the analysis that would be performed on the operation level or on equipment level.

Two-layer reasoning engine

The two-layer reasoning engine is a very important component of the system. The two layers are Petri Nets layer and safety model layer. All the connection information is present in Petri Nets layer and there is no connection between the safety models in the model layer, as shown in Figure 13. The analysis task contains two main subtasks: material propagation and deviation propagation. The intent of material propagation is to specify the materials for each operation and equipment. The material propagation is only carried out in Petri Nets layer. The outcomes of material propagation, which are the materials in specified material sets of each operation, are saved within pn_place. The main methods related to material propagation include (1) method to control the propagation of the materials; (2) method to deal with chemistry related operations, i.e., operation or equipment involved with different kinds of separations and reactions; and (3) method to calculate and determine the range of the operating conditions, i.e., pressure, temperature, level, and so on. Material propagation starts from pn_transition which do not have parent pn_transition, and propagates downstream following the connection established in the Petri Nets layer. The propagation stops only when the current path hits a dead end.

For deviation propagation, the Petri Nets layer is used to control the propagation between operations or equipments. The model layer is used to generate HAZOP analysis results given the process operating conditions and materials involved. The basic steps for analysis of a specific deviation include:

(1) Initialization and preparation for analysis.
(2) Local propagation to find consequences for the deviation inside the same pn_transition (shown as step 1 in Figure 13).
(3) Generation of current process conditions, and transferring the control to Petri Nets layer (shown as step 2 in Figure 13).
(4) If a pn_transition downstream still exists in Petri Nets layer, following the connection to propagate the deviation to next pn_transition (shown as step 3 in Figure 13).
(5) Repetition from step 2, until reaching the last pn_transition in Petri Nets layer.
(6) Similar steps are taken for cause analysis, with only change being that the direction of the propagation is upstream rather than downstream.

The methods for Petri Nets layer propagation are stored within the class HighSimEng, and the methods for safety model layer propagation are stored within the class LowSimEng. Analysis on single deviation starts from a call to HighSimEng. After verifying that the deviation is valid by a call to the HAZOPModel embedded in the pn_transition, HighSimEng is then invoked with the information on the exact location of the deviation in the digraph representation of the relations between process variables. The tasks for initialization and housekeeping, such as cleaning the status variables, recording deviation location, and so on, are carried out by HighSimEng. There are two methods of LowSimEng which can be called in HighSim Eng to invoke the LowSimEng for reasoning on safety model layer. One is used when the exact location of the deviation in the digraph is known, that is, when the deviation is the starting point of the analysis. The other is used when there is no specified starting point, i.e., the deviation is propagated down/up from upstream/downstream pn_transitions. In fact, the latter one calls the previous method with all the possible starting points in a pn_transition. The iteration is controlled by HighSimEng. After the local propagation in the model layer, the process conditions which may change due to introduction of the deviation are generated. The process conditions are stored in a *pn_token* with parameters for status of the mass, heat, flow rate, pressure and the concentration for each individual material involved in the operation. Status of process variables for the downstream/upstream pn_transition are specified according to the information stored in the token, and then a call to LowSimEng transfers the control to LowSimEng.

There is no connection in the model layer, and the LowSimEng works only on a single safety model. Depth first search methods are used for the local propagation. Following the connections on dg_node as represented by the arcs, the node states of the next node or previous node are determined. The local propagation continuous until the next node is a consequence node, or a cause node. Then the consequences or the causes for the deviation are assessed. The rules for assessment are evaluated by current pn_transition. As discussed in Knowledge Representation, Storage, Management and Acquisition section, the conditions, including parameter conditions, material conditions, equipment conditions and process conditions are parsed by a parser to determine the symbols and the relations between the symbols. The symbols correspond to basic process properties including material properties, material sets, parameters, and process conditions. Combination of material properties can also be used, and the criteria of assessing the combined condition are also part of the knowledge base. If all the conditions are satisfied, the consequences or causes are generated by specifying the material and equipment involved. In current implementation, in the text strings for consequence and cause, some identifiers have been used to determine what process specific information is to be inserted in. For example, in the cause of high pressure in Empty operation,

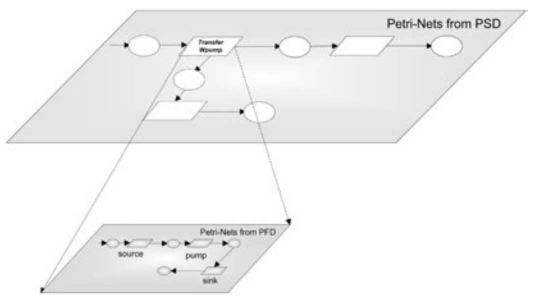


Figure 14. Analysis on operation level and equipment level.

'thermal expansion of cryogenic materials (⟨material⟩)', '⟨material⟩' is an identifier. When generating the cause, the identifier '⟨material⟩' is replaced by a string containing names of the materials satisfying the material conditions. Other identifiers include ⟨equipment⟩ for specifying main equipment of the operation, ⟨toequipment⟩ and ⟨fromequipment⟩ for specifying information about special equipments. Other identifiers can also be added and the corresponding process information can be defined in knowledge base. A typical result contains information of the location of the deviation specified by the ID of the operation and unit operation, the deviation type, cause, consequence, the location of the cause, and the location of the consequence, the safeguard, severity and recommendation. Some specifications for the result are generated from information recorded during analysis, while some information is generated from the rules for assessment of the consequence and the cause. Each result also contains a causal path to record every node the analysis path goes through from starting deviation to the end deviation. The result is then stored in current pn_transition.

Reasoning on two levels

As shown in previous sections, the process could be abstracted into two levels, operation level and equipment level. Both the levels are represented using Petri Nets with a model layer beneath each of them. The two-layer reasoning engine is used to carry out the analysis on these two levels, as illustrated in Figure 14. The reasoning on operation level is first carried out, and the analysis on equipment level is controlled by operation level. In the Petri Nets representation layer, the connection between these two levels is through ID of the equipment recipe_ transition embedded in the operation pn_transition. In the model layer, the connection is through the mutual process variables. During the method call to HighSimEng, after the local deviation propagation in operation level, analysis is carried out in equipment level.

CONCLUSIONS

As a long-term vision, for PHASuite to become a successful process safety analysis tool, it should possess several properties. First and most important of all, it should have an open structure, within which users are able to add knowledge to the system with little help from knowledge engineers. The open structure also makes it possible to create standards for causal model. Information sharing with other tools is also important to reduce the effort of reentering process information and to improve the use of the safety analysis results. The system is an information consumer since it shares process information with other software packages, and also an information producer since it generates safety analysis results and shares the results with other software packages including abnormal situation management tools, operator training tools and so on. The tool could also be used as a company-wide repository for knowledge on operating procedures and safety.

This vision is important not only for the progress of this tool, but it also serves as a guideline for designing and implementing the current version. Based on this vision, a knowledge engineering framework for automated HAZOP analysis for chemical processes has been proposed. The framework acts as the guide for the development of current version of PHASuite and also lays out a good foundation for future development. The main components of the framework include: (1) ontologies based information sharing; (2) model-based knowledge representation for operation knowledge as well as safety knowledge; (3) relational database based knowledge storage; and (4) case-based technique for knowledge management and for incorporating experience into the system. Powerful reasoning methodologies based on various artificial intelligence techniques have also been developed. Coloured Petri Nets representation is created based on process specific information. A process is abstracted to two levels, operation level and equipment level. These two levels are bridged using functional representation. Analysis is carried out on both the levels.

PHASuite, a sophisticated software system for automated process safety analysis using HAZOP technique, has been constructed based on the above framework. Software engineering methodologies are applied to ensure the quality, flexibility, and easy maintenance of the system. Component programming strategies are used to construct the two-level and two-layer reasoning engine that performs reasoning on the Coloured Petri Nets representation using the knowledge base to generate HAZOP results. Auxiliary system components, including graphical user interface, results management, and object layers to process information and knowledge base, have been built to make the system ready-to-use. The work done here is ready for real-life applications (some parts of the work have already been extensively tested), and its open structure ensures scope for further improvement. A tool for knowledge acquisition is also implemented as an important facility.

The framework has the following characteristics: (1) appropriate representation of HAZOP analysis procedure for chemical processes; (2) supporting abstraction and analysis on different level of the process; (3) incorporating general domain knowledge, as well as experience; (4) learning ability, i.e., the analysis capability and quality should increase as more processes are analysed; and (5) easy to understand and implement. From application point of view, this system can: (1) improve quality of HAZOP analysis; (2) save time and effort of human experts involved in performing HAZOP analysis; (3) easily incorporate further enhancement; and (4) integrate with other related software systems.

REFERENCES

Aamodt, A. and Plaza E., 1994, Case-based reasoning: foundational issues, methodological variations, and system approaches, *AI Communications*, 7(1): 39–59.

Abernethy, N.F. and Altman, R.B., 1989, SOPHIA: providing basic knowledge services with a common DBMS. *Proceedings of the 5th KRDB Workshop*, Seattle, WA, USA.

Angele, J., Fensel, D., Landes, D., Neubert, S. and Studer, R., 1993, Model-based and incremental knowledge engineering: the MIKE approach. *Knowledge Oriented Software Design* (Elsevier, Amsterdam).

ANSI/ISA-588.01, 1995, Batch Control – part 1: models and terminology.

Aspen Technology Inc., 2000, *Batch Plus User Manual*, Version 2.2.

Autodesk Inc., 2002, AutoCAD 2002, http://www.autodesk.com.

Bergmann, R., Goker, M., Manago, M. and Wess, S., 1999, *Developing Industrial Case-Based Reasoning Applications: the INRECA Methodology* (Springer-Verlag, Berlin Heidelberg, Germany).

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

CCPS (Center for Chemical Process Safety), 1985, *Guidelines for Hazard Evaluation Procedures* (American Institute of Chemical Engineers, New York, USA).

Chandrasekaran, B., 1994, Functional representation and causal processes, *Advances in Computers*, 38: 73–143.

Chandrasekaran, B., Josephson, J.R., Davis, J.F. and Rizzoni, G., 1998, Functional and diagrammatic representation for device libraries, *Annual Report*, Ohio State University, USA.

Davis, R., 1984, Diagnostic reasoning based on structure and behavior, *Artificial Intelligence*, 24: 347–410.

de Kleer, J. and Brown, J.S., 1984, A qualitative physics based on confluences, *Artificial Intelligence*, 24: 7–83.

Dyadem International Ltd, 2001, *PHA-Pro 5 Document*.

Farquhar, A., Fikes, R., Pratt, W. and Rice, J., 1995, Collaborative ontology construction for information integration, *Technical Report of Knowledge Systems Laboratory*, Stanford University.

Forbus, K.D., 1984, Qualitative process theory, *Artificial Intelligence*, 24: 85–168.

Freeman, R.A. ,Lee, R. and McYamana, T.P., 1992, Plan HAZOP Studies with an expert system, *Chemical Engineering Progress*, 88(8): 28–32.

Gensym, 1997, *G2 Reference Manual Version 5.0* (Gensym Corporation, Cambridge, MA, USA).

Goel, A.K. and Stroulia, E., 1996, Functional device models and model-based diagnosis in adaptive design, *AI EDAM*, 10(4): 355–370.

Gruber, T.R., 1993, Toward principles for the design of ontologies used for knowledge sharing, *International Workshop on Formal Ontology*, Padova, Italy.

Harold, E.R. and Means, W.S., 2001, *XML in a Nutshell*, (O'Reilly & Associates, Inc., USA)

Intergraph Corporation, 2002, SmartPlant P&ID, http://www.intergraph.com/ppo/smartplant

Jensen, K., 1996, *Colored Petri Nets: Basic Concepts, Analysis Methods, and Practical Use* (Springer-Verlag, London, UK).

Kang, B. and Yoon, E.S., 2001, Application of automated hazard analysis by new multiple process-presentation models to chemical plants, *Industrial Engineering Chemistry Research*, 40: 1891–1902.

Karvonen, I., Heino, P. and Suokas, J., 1990, Knowledge-based approach to support HAZOP studies, *Technical Research Center of Finland: Research Report*.

Kinoshita, A., Umeda, T. and O'Shima, E., 1981, An algorithm for synthesis of operational sequences of chemical plants, 4th European symposium on computerized control and operation of chemical plants, CHEMCONTROL, Vienna, Austria.

Kletz, T., 1999, *Hazop and Hazan: Identifying and Assessing Process Industry Hazards*, 4th edition (The Institution of Chemical Engineers, Rugby, UK).

Knowlton, R.E., 1981, *An Introduction to Hazard and Operability Studies: The Guide Word Approach*, (Chemetics International Ltd. Vancouver, Canada).

Kolodner, J.L., 1983, Maintaining organization in a dynamic long-term memory, *Cognitive Science*, 7: 243–280.

Kolodner, J.L., 1993, *Case-Based Reasoning* (Morgan Kaufmann Publishers, San Mateo, CA, USA).

Nickols, F., 2000, The knowledge in knowledge management, *Knowledge Management Yearbook* (Butterworth-Heinemann, Boston, USA).

Norrie, M.C., Reimer, U., Lippuner, P., Rys, M. and Schek, H.J., 1994, Frames, objects and relations: three semantic levels for knowledge base systems reasoning, *Proceedings of First Workshop of Knowledge Representation Meets Databases*, Saarbrücken, Germany.

Pajula, E., Seuranen, T., Koiranen, T. and Hurme, M., 2001, Synthesis of separation processes by using case-based reasoning, *Computers & Chemical Engineering*, 25: 775–782.

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

Pegah, M., Sticklen, J. and Bond, W., 1993, Functional representation and reasoning about the F/A-18 aircraft fuel system, *IEEE Expert*, 8(2): 65–71.

Peterson, J., 1981, *Petri Net Theory and the Modeling of Systems* (Prentice-Hall Inc., NJ, USA).

Poeck, K. and Gappa, U., 1993, Making role-limiting shells more flexible, *Proceedings of the 7th European Knowledge Acquisition Workshop*, Toulouse and Caylus, France.

Ramakrishnan, R. and Gehrke, J., 1999, *Database Management Systems*, 2nd edition (McGraw Hill, USA).

Ramanathan, R., 1994, Providing object-oriented access to a relational database, In Cordes, D. and Vrbsky, S. (eds), Proceedings of the 32nd Annual Southeast Conference, Tuscaloosa, Alabama, USA.

Riesbeck, C.K. and Schank, R.C., 1989, *Inside Case-Based Reasoning* (Lawrence Erlbaum Associates Publishers, NJ, USA).

Rich, E. and Knight, K., 1990, *Artificial Intelligence*, 2nd edition (McGraw-Hill, NY, USA).

Rich, S.H. and Venkatasubramanian, V., 1987, Model-based reasoning in diagnostic expert systems for chemical process plants, *Computers & Chemical Engineering*, 11(2): 111–122.

Schank, R., 1982, *Dynamic Memory: A Theory of Reminding and Learning in Computers and People* (Cambridge University Press, NY, USA).

Sembugamoorthy, V. and Chandrasekaran, B., 1986, Functional representation of devices and compilation of diagnostic problem-solving systems, In *Experience, Memory, and Reasoning* (Lawrence Erlbaum Associates, NJ, USA).

Slade, S., 1991, Case-based reasoning—a research paradigm, *AI Magazine*, 12(1): 42–55.

Srinivasan, R. and Venkatasubramanian, V., 1998, Automating HAZOP analysis of batch chemical plants: Part I. The knowledge representation framework, *Computers & Chemical Engineering*, 22(9): 1345–1355.

Sticklen, J., Kamel, A. and Bond, W.E., 1991, Integrating quantitative and qualitative computations in a functional framework, *Engineering Applications of Artificial Intelligence*, 4(1): 1–10.

Uschold, M. and Gruninger, M., 1996, Ontologies, principles, methods and applications, *The Knowledge Engineering Review*, 11(2): 93–136.

Vaidhyanathan, R., 1995, *A Model-Based Framework for Automating HAZOP Analysis of Continuous Process Plants*, PhD thesis, LIPS Lab, Purdue University.

Vaidhyanathan, R. and Venkatasubramanian, V., 1996, HAZOPExpert: an expert system for automating HAZOP analysis, *Process Safety Progress*, 15(2): 80–88.

Venkatasubramanian, V. and Rich, S.H., 1988, An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis, *Computer & Chemical Engineering*, 12(9–10): 903–921.

Venkatasubramanian, V., Zhao, J. and and Viswanathan, S., 2000, Intelligent systems for HAZOP analysis of complex process plants, *Computers & Chemical Engineering*, 24: 2291–2302.

Viswanathan, S., 2000, A hierarchical model-based intelligent systems framework for synthesis of safe batch operating procedures, PhD Thesis, Purdue University, USA.

Viswanathan, S., Mockus, L. and Venkatasubramanian, V., 1998, iTOPS: an intelligent tool for operating procedures synthesis, *Computers & Chemical Engineering*, 22: S601–S608.

Walas, S., 1988, *Chemical Process Equipment: Selection and Design* (Butterworth-Heinemann, MA, USA).

Waters, A. and Ponton, J.W., 1989, Qualitative simulation and fault propagation in process plants, *Trans IChemE, Chemical Engineering Research and Design*, 67(4): 407–422.

Weiss, S., Kulikowski, C., Amarel, S. and Safer, A., 1978, A model-based method for computer-aided medical decision making, *Artificial Intelligence*, 11: 145–172.

Xia, Q. and Rao, M., 1999, Dynamic case-based reasoning for process operation support systems, *Engineering Applications of Artificial Intelligence*, 12: 343–361.

Zhao, J., Viswanathan, S., Zhao, C., Mu, F. and Venkatasubramanian, V., 2000, Computer-integrated tools for batch process development, *Computers & Chemical Engineering*, 24: 1529–1533.

The manuscript was received 1 March 2004 and accepted for publication after revision 4 April 2005.