Prototype for Integrated Hazard Analysis

Dong-Hsiung Kuo, Ding-Shang Hsu, Chuei-Tin Chang, and Dong-Han Chen
Dept. of Chemical Engineering, National Cheng Kung University, Tainan, Taiwan 70701, ROC

The prototype of an integrated hazard-analysis system (IHAS) was developed. Essentially any process can be analyzed with this software if the system topology is correctly supplied by the user. Since all algorithms adopted are digraph-based, the system digraph must be constructed first with IHAS. The embedded feedforward and feedback loops are then identified and classified. On the basis of this information, three widely accepted hazard-assessment procedures—FTA, ETA, and HAZOP—can be performed automatically. From the results obtained in practical applications, one can see that the quality of hazard analysis is indeed improved if IHAS can be used as an aid to the human experts.

Introduction

Hazard analysis is an important step in designing or revamping any chemical plant. Fault-tree analysis (FTA), event-tree analysis (ETA), and hazard and operability study (HAZOP) are three of the most effective and widely recognized methods used for such a purpose in the industry. Since, in general, these techniques are implemented manually by experts in brainstorming sessions, the need for manpower and time is often overwhelming. Thus, there are real incentives to automate these safety-analysis processes.

In the past two decades, the research on automatic fault-tree analysis has advanced significantly. Several efficient tools have been developed to synthesize fault trees, such as, transfer function (Fussell, 1973), decision table (Salem et al., 1977; Kumamoto and Henley, 1979), digraph (Lapp and Powers, 1977), reliability graph (Camarda et al., 1978), and minifault tree (Kelly and Lees, 1986a–d; Kahn and Hunt, 1989). In essence, the approaches taken in these studies are the same, that is, to build qualitative system models for use in the automatic fault-tree construction algorithms. Generally speaking, digraph is one of the most popular models. Numerous publications exist in the literature (Shaeiwitz et al., 1977; Chamow, 1978; Lambert, 1979; Lapp and Powers, 1979; Allen and Rao, 1980; Cummings et al., 1983; Allen, 1984; Andrews and Morgan, 1986; Andrews and Brennan, 1990). Although it has been shown in these works that the digraph-based techniques are useful in practical applications, the original method proposed by Lapp and Powers (1977) is still deficient under certain conditions (Andow, 1980, 1981; Galluzzo and Andow, 1984). Based on qualitative steady-state analysis, Chang and Hwang (1992) proposed new fault-tree structures and improved the original algorithm. After further studies (Chang and Hwang, 1992, 1994; Chang et al., 1993, 1994; Hwang, 1993) the digraph-based approach has not been only developed into a practical tool for building fault trees, but also extended to solve problems in event-tree synthesis.

On the other hand, research projects that aimed to automate HAZOP began to emerge in the recent literature, for example, Weatherill and Cameron (1989), Venkatasubramanian and Vaidhyanathan (1994), Leone (1996), and Srinivasan and Venkatasubramanian (1996). It should be noted that the analysis performed in HAZOP format tends to be less rigorous and comprehensive than that with fault trees and event trees. Further, part of the results generated with the latter two approaches can be adopted as the conclusions of HAZOP. Specifically, the contents of a HAZOP report can be produced with the information sources listed in Table 1. Thus, it is the intention of this work to integrate FTA, ETA, and HAZOP into one software. The prototype of such a system has already been developed and tested with a variety of practical examples in our research. From the results generated so far, one can conclude that this software can be used not only in real applications but also as a tool for training purpose.

Framework of an Integrated Hazard-Analysis System

To perform FTA, ETA and HAZOP for any given process, the generic software should at least be able to perform the following tasks:

Correspondence concerning this article should be addressed to C.-T. Chang.

Table 1. **Information Sources for Building a HAZOP Report**

Deviation	Causes	Consequences	Safeguards	Actions
System default and/or user input	Cut-sets of the fault tree using "deviation" as the top event	Accident sequences of the event trees using the events in cut-sets as the initial events	Same as "consequences"	Results inferenced by an expert system based on "causes" and "consequences"

1. Transformation of piping and instrumentation diagram (P&ID) into machine-processable input codes.
2. Construction of system digraph according to the input codes and component-digraph database.
3. Identification of the feedback loops (FBLP) and feed-forward loops (FFLP) embedded in system digraph.
4. Fault-tree analysis
   (a) synthesize fault trees using user-specified top events.
   (b) determine the cut sets.
5. Event tree analysis
   (a) synthesize event trees using user-specified initial events.
   (b) determine the accident sequences.
6. Generation of HAZOP reports
   (a) select possible deviations.
   (b) use the deviations as top events and construct the corresponding fault trees.
   (c) Determine the cut-sets of each fault tree.
   (d) Use a basic event in every cut-set as the initial event and build the corresponding event tree.
   (e) Determine the accident sequences of each event tree.
   (f) Translate the symbols in cut-sets and accident sequences into colloquial descriptions.
   (g) Use the preceding results as the conditions of IF/THEN rules to identify the remedial actions required.
   (h) Fill the HAZOP report with results generated in steps (a), (f), and (g).

The prototype of an integrated hazard-analysis system (IHAS) has been developed in our study. This system consists of 14 tool programs, a component-digraph database, and a remedial-action knowledge base. The system's framework, depicting the relationships among tool programs, database, and knowledge base, is presented in Figure 1.

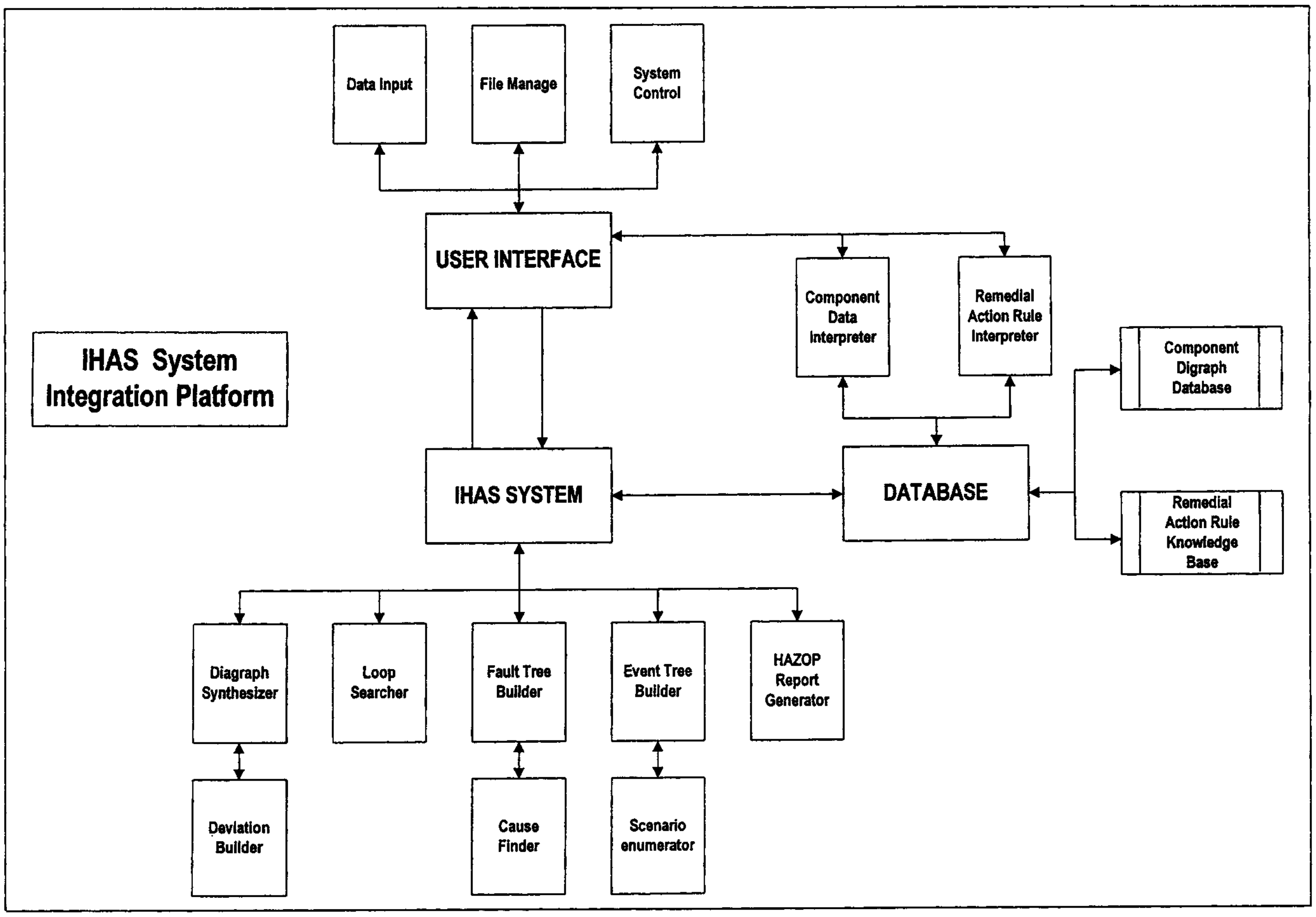

Figure 1. System framework of IHAS.

Table 2. Classification of Common Process Equipments

Equipment Class	Equipment Subclass
Control elements	Valve Sensor Controller Other
Vessels	Column Buffer tank Storage tank Reactor Other
Heat exchangers	Shell and tube Double pipe Furnace Boiler Cooling tower Other
Rotating machineries	Pump Compressor Other
Protective devices	Trip, interlock Pressure relief valve Rupture disc Other

Component-Digraph Database

Since all algorithms adopted in IHAS are digraph-based, it is necessary to first construct a component database that contains a collection of small digraphs. Each of these digraphs is a component model and is used as a building block of the system model. There are, of course, a large number and a wide variety of equipment in any chemical plant. In the proposed system, the common ones are classified according to Table 2. Each is represented with a standard digraph and stored in the database. If there is a need to incorporate process-specific information, the user can make use of one of the tool programs, that is, *component-data interpreter*, to modify the system-supplied digraphs or even add new ones into the database.

The model-building process of the component digraphs is inevitably manual. A simple example is presented in the sequel to illustrate the systematic model-construction procedure adopted in this work.

Example 1. Let us consider the control valve presented in Figure 2. The following steps can be taken to determine the corresponding component digraph:

1. Label all pipelines and signal lines with numbers.

2. Combine the symbols representing process variables with the labels specified in step 1 to create a set of basic nodes.

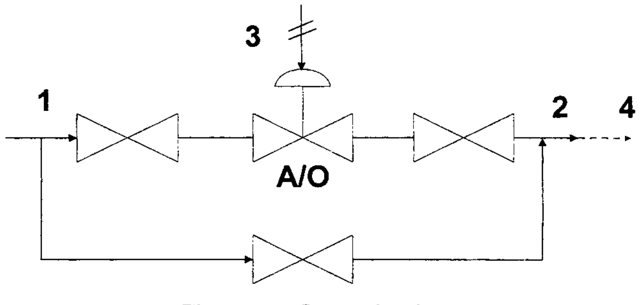

Figure 2. Control valve.

In IHAS, the default process variables are mass flow rate, temperature, pressure, and pressure/electrical signals. The symbols m, t, p, and s are used to represent these variables, respectively. The definitions of all process variables used in this article can be found in the "Notation" section. Seven basic nodes can be identified in the present case: $t1$, $p1$, $m1$, $t2$, $p2$, $m2$, and $s3$.

3. Introduce additional nodes to represent equipment failures of types A and B.

In previous research (e.g., Chang and Hwang, 1992), the equipment failures were classified into four types— A, B, C, and D—according to their digraph configurations and propagation patterns of their effects. For the sake of completeness, the definitions of these four types of failures are included in the Appendix of this article. Among the four, only types A and B are represented by nodes. For example, one may use the nodes presented in Table 3 to characterize the failure modes of the control valve system in Figure 2. Notice that the two reserved letters a and b are prefixed to the node symbols for the purpose of distinguishing type A and B failures from the process variables. The meanings of all such symbols used in this article can be found in the Notation section.

4. Connect the nodes according to their cause-and-effect relations and then lump all nodes representing the flow rates in the same pipeline into a node denoting the "final" output flow rate.

In a typical P&ID, it is not uncommon to find more than one component installed on the same pipeline. If the corresponding component digraphs are connected directly, the resulting system digraph may contain nodes representing flow rates at different locations of the same pipeline. This causes difficulties in analyzing the digraph structure, since feedback loops are formed among these nodes. In fact, there is no need to consider flow-rate variation along the same pipeline in hazard analysis. Thus, in building the component digraphs, a "final" output flow rate is always reserved. If there are several components installed on the same pipeline, the same label should be assigned to the node representing final output flow rate in each component digraph.

In the control valve example, the resulting component digraph can be found in Figure 3. To facilitate storage of this information in a computer, the digraph configuration is translated into a *node list* as follows:

$t2$	$t1$					
$p2$	$p1$					
$m4$	$s3$	$alk0$	$bvfcdn0$	$bvfcup0$	$byfo0$	$bcvfc0$

Notice that, in each row of this list, the first element is the output node of the other elements.

Table 3. Type A and B Failures in the Control Valve System

Node	Failure Mode
$alk0$	Control valve leaks
$bvfcup0$	Inlet isolation valve fails closed
$bvfcdn0$	Outlet isolation valve fails closed
$byfo0$	Bypass valve fails open
$bcvfc0$	Control valve fails closed

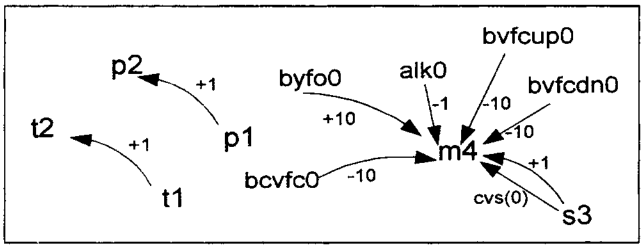

Figure 3. Digraph model of the control valve in Figure 1.

5. Specify the possible deviation values associated with every node and the gain corresponding to every edge. Also, use conditional edges to represent failures of type C and D.

The preceding information is stored in a *gain list*. For the present case, this list can be found below:

$t2$	$t1$	$+1$	$+1$	$+10$	-1	-10	
$p2$	$p1$	$+1$	$+1$	$+10$	-1	-10	
$m4$	$s3$	$+1$	$+1$	$+10$	-1	-10	$cvs0+0$
$m4$	$alk0$	-1	$+1$	$+10$			
$m4$	$bvfcdn0$	-10	$+1$				
$m4$	$bvfcup0$	-10	$+1$				
$m4$	$byfo0$	$+10$	$+1$				
$m4$	$bcvfc0$	-10	$+1$				

Notice that the first two elements of each row are output node and input node, respectively, and the third element is the corresponding gain value. The other values in the row are the allowable input deviations. For example, the allowable deviations of $t1$ are the last four values listed in the first row. Finally, if one wishes to consider type C and/or type D failures in the component digraph, then conditional edges should be added. For example, the conditional edge between $m4$ and $s3$—$cvs(0)$—in Figure 3 denotes a type C failure "control valve stuck." These edge conditions are placed at the end of each row of the gain list. In this case, the condition appears at the end of the third row in the preceding list. Again, the definitions of all symbols used to denote types C and D failures are presented in the Notation section.

Assemblage of the System Digraphs

After modifying the existing component digraphs and adding new ones to the component database, a system digraph can be constructed according to the topology of P&ID. The equipment and their interconnections in P&ID can be specified interactively using the tool program *process reader*. Another tool program *digraph synthesizer* can then be utilized to build the system model automatically. In essence, three tasks are performed in *digraph synthesizer*:

1. Retrieving the component digraphs from database;
2. Relabeling them according to user specifications obtained with the *process reader*;
3. Combining the node and gain lists of all components and then removing repeated elements from the resulting lists.

The aggregated node and gain lists obtained with the *digraph synthesizer* should be a complete representation of the system model.

Loop Identification and Classification

As mentioned before, the system digraph is obtained by connecting component digraphs corresponding to all units in the system. Due to interaction between units, complex "loops" are often formed in these system models. From a purely structural viewpoint, two types of loops are important for implementing the digraph-based safety assessment techniques: feedforward loops (FFLP—two or more paths from one node to another in a digraph) and feedback loops (FBLP—a path through the nodes in a digraph that starts and terminates at the same node). These loops can be further classified according to their functions: control loops, protection loops, and process loops. The control loops can be identified by searching for the nodes that represent the variables in control systems. Similarly, the nodes on protection loops are variables in the protective systems, that is, the trips, interlocks, and pressure relief valves. On the other hand, the process loops are usually caused by complex process flow configurations or by complex physical and chemical relations amont process variables. In order to ensure the validity of the digraph model, it is obvious that the effects of any external disturbance to these different loops must be evaluated correctly. As a result, the task of identifying and classifying all embedded loops becomes an indispensable step of any credible digraph-based hazard analysis.

In general, the embedded FFLPs and FBLPs can be easily identified in a digraph by inspection if it is associated with a simple stand-alone unit. However, a large number of *tangled* loops may exist even when the system P&ID is only moderately busy (Chang and Hwang, 1992). Manual enumeration of these loops is therefore laborious and error-prone in practice. An algorithm has already been developed in a previous study (Chang et al., 1997) to automatically identify all loops in any digraph. The tool program *loop searcher* in IHAS was written based on this algorithm.

The results obtained by executing the *loop searcher* are lists of FFLPs and FBLPs. These loops must be further analyzed according to their functions. Notice that a digraph model contains only a collection of cause-and-effect information between adjacent nodes. In a treelike digraph, the effect of a failure on any variable can be easily determined by identifying the path between the two cooresponding nodes and evaluating the product of edge gains on the path. However, this approach is no longer valid if loops exist. The variables associated with nodes on an FBLP or at the end of an FFLP may be affected by an external disturbance along two or more paths. Furthermore, the product of edge gains along different paths may be opposite in sign. Thus, without additional information about the system, the corresponding deviations in the loop variables cannot be determined.

If the FFLPs and FBLPs are control loops, the net effects of external disturbances on the loop variables can be predicted (Chang and Hwang, 1992). In the cases of protection loops and process loops, these effects are process-specific and must be provided by the user in a data file. The format of this data file and the procedure for its preparation are illustrated in the following two examples:

Example 2. Let us consider the storage system in Figure 4 and the corresponding digraph in Figure 5. If the liquid level is too high, the switch will be triggered to set the solenoid

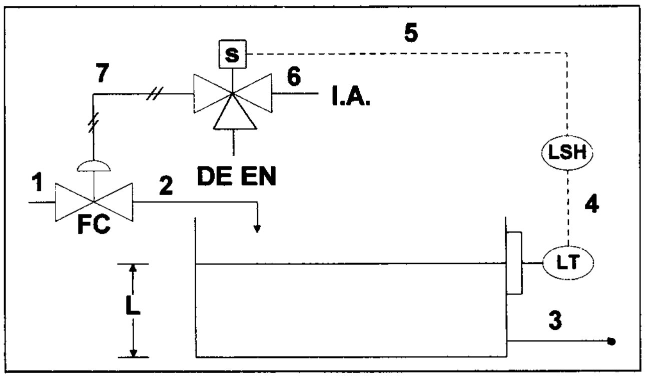

Figure 4. Storage system.

valve at the deenergized position. Consequently, the instrument air will be vented and control valve closed.

An FBLP can be identified with the *loop searcher*, that is, $L \leftarrow m2 \leftarrow s7 \leftarrow s5 \leftarrow ss4 \leftarrow L$, and no FFLP can be found. Since the FBLP is a protection loop, the user must provide the data file presented below:

$p1$	$+10$	L	-10	$m2$	-10	$s7$	-10	$s5$	$+10$	$ss4$	-10
$s6$	$+10$	L	-10	$m2$	-10	$s7$	-10	$s5$	$+10$	$ss4$	-10
$bsfc$	$+1$	L	-10	$m2$	-10	$s7$	-10	$s5$	$+10$	$ss4$	-10
atd	$+10$	L	-10	$m2$	-10	$s7$	-10	$s5$	$+10$	$ss4$	0
btd	$+10$	L	-10	$m2$	-10	$s7$	-10	$s5$	$+10$	$ss4$	$+10$
$m3$	-10	L	-10	$m2$	-10	$s7$	-10	$s5$	$+10$	$ss4$	-10

Notice that in each row of this list the outcomes caused by a particular off-FBLP input are specified. The first two elements represent the input node and its deviation value. There are also another five pairs of elements remaining in the same row. Each pair denotes a loop variable and the corresponding deviation.

Example 3. Next, let us consider the heat exchange system in Figure 6 and the corresponding digraph in Figure 7. In this system, the switch will be actuated when the flow rate

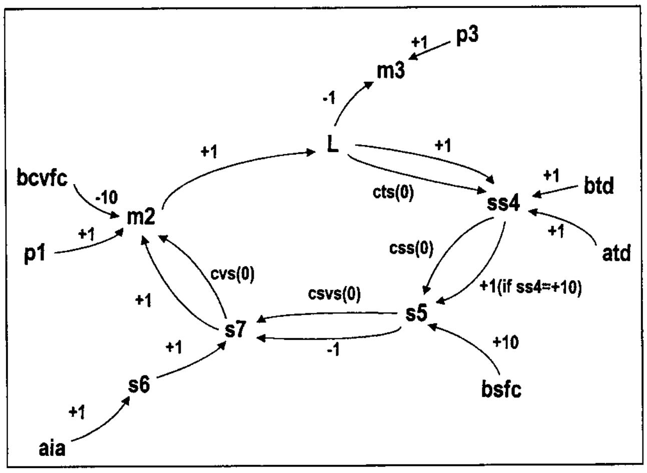

Figure 5. System digraph of the storage system in Figure 4.

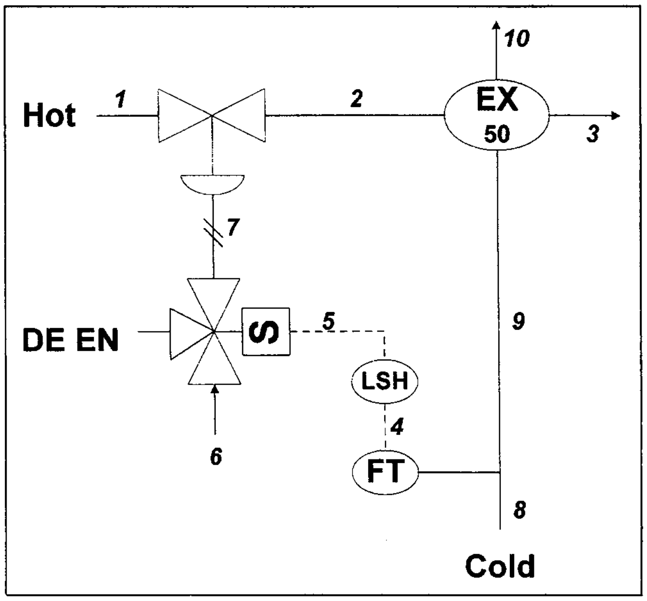

Figure 6. Heat-exchange system.

of cold stream is too low. Again, the solenoid valve will then be set at the deenergized position to vent air and, consequently, the control valve will be closed to cut off the flow of hot stream.

Two FFLPs can be found with the *loop searcher*:

$$\begin{Bmatrix} t3 \leftarrow m9 \leftarrow m8 \\ t3 \leftarrow m2 \leftarrow s7 \leftarrow s5 \leftarrow ss4 \leftarrow m8 \end{Bmatrix}$$

and

$$\begin{Bmatrix} t10 \leftarrow m9 \leftarrow m8 \\ t10 \leftarrow m2 \leftarrow s7 \leftarrow s5 \leftarrow ss4 \leftarrow m8 \end{Bmatrix}.$$

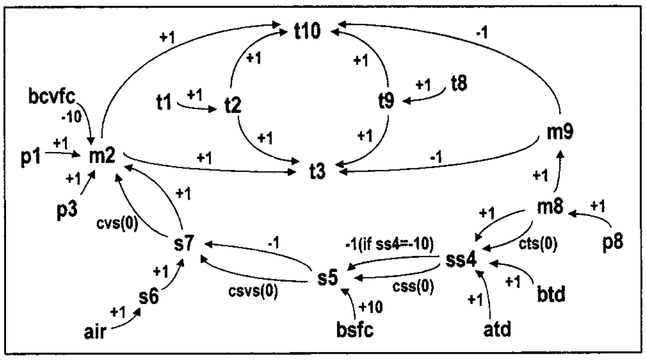

Figure 7. System digraph of the heat-exchange system in Figure 6.

These two feedforward loops are formed by the variables in the protective system. The net effects of disturbances entering the starting nodes must be provided by the user. In IHAS, this process-specific information should be specified in a standard data file as follows:

$t3$	$m8$	0	$cvs(+0)$	-1	$csvs(+0)$	-1	$css(+0)$	-1	$cts(+0)$	-1
$t10$	$m8$	1	$cvs(+0)$	-1	$csvs(+0)$	-1	$css(+0)$	-1	$cts(+0)$	-1

In this file, the net effects of inputs entering the starting node of a particular FFLP are recorded in the same row. The first two elements are the terminating and starting nodes of the FFLP, respectively. The third element is the net gain between these two nodes. The rest of the elements should be given in pairs. The first in each pair is the condition of a type C or type D failure and the second the resulting net gain value.

It should be noted that, if the user is not familiar with the digraph models and also knowledgeable in process-specific information, assessing the net effects of external disturbances on protection and process loops is probably not a trivial task. This difficulty can be overcome with more friendly interfaces. The corresponding revisions in computer codes are currently still in progress. Consequently, the software IHAS described in this article should only be considered as a prototype, but not the final commercial product.

Fault-tree Analysis

At the present time, two important steps in fault-tree analysis can be carried out automatically with IHAS: fault-tree synthesis and cut-set identification. Since the latter task is performed on the basis of an existing standard algorithm (CCPS, 1992) and its implementation is straightforward, a discussion of this procedure is thus omitted for the sake of brevity. The flow chart of the fault-tree construction algorithm has also been published (Chang and Hwang, 1992). Following are some of the practical issues in carrying out this algorithm.

Representation and ratification of the events in fault trees

In IHAS, all events are represented as $X(v)$, where X denotes the node symbol and v is the deviation value. Notice that a node may be used to represent either a process variable or a component failure. In the former case, the deviation values can be $+10$ (too high), $+1$ (high), 0 (normal), -1 (low), and -10 (too low). If X represents an equipment malfunction, then only the positive deviations are needed to reflect the severity of failure.

Every event in a fault tree must be checked against the system digraph. First of all, the node symbol X must exist in the corresponding node list. Second, the deviation v should be an allowed value, that is, a value specified in the gain list. In some cases, the top event is associated with a node without outputs. Then it is only necessary to impose the first requirement since only the input values are provided in the gain list.

Identification of primal nodes

The process of fault-tree synthesis is terminated when all inputs of the tree are found to be basic events. A basic event is associated with a primal node, that is, a node without inputs, in the system digraph. Consequently, there is a need to identify all primal nodes in advance. In IHAS, this task is accomplished by examining the elements in the node list. Specifically, a list of primal nodes can be obtained by removing the elements in the first column of the node list from a list of all nodes in the system digraph.

Calculation of input deviations

In developing a fault tree, the inputs to a nonbasic event can be determined by identifying the corresponding input nodes in digraph. The deviation value of the input can be calculated according to the following equation:

$$\text{Input deviation} = \frac{\text{output deviation}}{\text{gain}}.$$

If the resulting deviation value cannot be found in the gain list, then the corresponding input event should not be included in the fault tree.

Compilation of fault-tree structures

Different fault-tree structures may be connected to events associated with nodes at different locations in the digraph. As mentioned before, a node may be related to its inputs in a treelike digraph or located on a loop. In the latter case, these loops can be classified into FFLPs and FBLPs according to their structural characteristics. The *loop searcher* can be used to identify all of them automatically. The FFLPs and FBLPs can be further classified into control loops, protection loops, and process loops according to their functions. A collection of standard fault-tree structures have been developed for all cases mentioned earlier (Chang and Hwang, 1992, 1994; Chang et al., 1994). In implementing these structures, the net effects of external disturbances must be determined first. Since the behaviors of control FFLPs and FBLPs can be predicted, default values have been built in the IHAS system already. However, for protection loops and process loops, the needed process-specific data must be supplied by the user in the form of the data file and standard data file shown earlier.

Construction procedure

Although a complete flow chart can be found in Chang and Hwang (1992), an example is still provided here to illustrate the fault-tree construction procedure.

Example 4. Let us consider the fictitious digraph presented in Figure 8a. If the top event is selected to be $X1(+1)$, the corresponding construction steps should be:

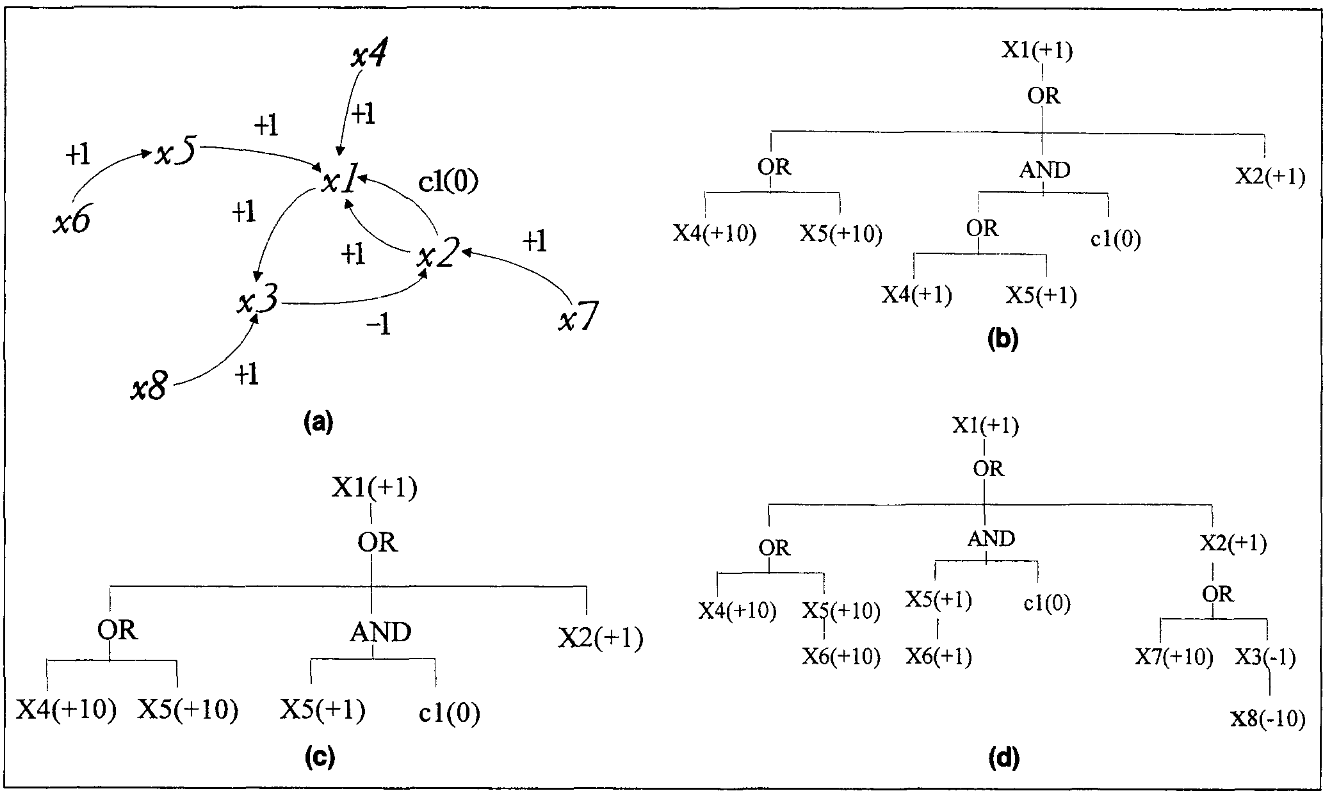

Figure 8. Example 4: (a) fictitious sytem; (b), (c), (d) development process of fault tree (i), (ii), (iii), respectively.

1. *Check to see if the current event is basic.* Initially, the current event is the top event. In later steps, the current event is an undeveloped event assigned automatically by the system. The present task is done simply by checking whether the corresponding node is primal. If it is primal, then the following four steps should be skipped and the next undeveloped event should be treated as the current event. In the present case, the primal nodes are $X4$, $X6$, $X7$, and $X8$. Consequently, the current event $X1(+1)$ is nonbasic and the fault tree can be developed further.

2. *Connect the current event with a proper tree structure.* Since $X1$ is located on a control FBLP, a standard structure with default deviation values can be used to expand the fault tree. The result is presented in Figure 8b.

3. *Remove unallowed input events.* In expanding the fault tree, it is necessary to make sure that all input events are allowed. Specifically, the computed input values should appear in the gain list. Otherwise, it should be removed from the fault tree. If, for example, the allowed deviation for $X4$ is $+10$ only, then the fault tree in Figure 8b should be changed to the one in Figure 8c.

4. *Delete repeated or inconsistent input events.* In the process of developing a fault tree along a path on FFLP or FBLP, it is possible to reach a node a second time. In that case, the corresponding input event is either repeated—its deviation value is the same as the previous one—or inconsistent—a different deviation value is obtained. In both cases, the input event should be removed from the tree. This step is not applicable to Figure 8c, but will become clear in later steps..

5. *Identify and record the undeveloped events.* The undeveloped events are recorded in a list. All inputs of the newly connected fault tree structure should be included in this list except failure conditions of types C or D. Initially, the list contains only the top event $X1(+1)$. After the fault tree in Figure 8c is obtained, all its inputs—$X4(+10)$, $X5(+10)$, $X5(+1)$, and $X2(+1)$—should be written in the next row of the list.

6. *Check if the construction process can be terminated and assign the current event.* The construction process should be terminated when there are no undeveloped events remaining in the list. Otherwise, the undeveloped events in the next available row will be treated in turn as the current event and processed according to steps 1–5.

The completed fault tree of this example is presented in Figure 8d. Notice that, under the event $X3(-1)$, it appears that the event $X1(-1)$ should also be included as an input according to the system digraph. However, this event is inconsistent and thus must be removed (see step 4).

7. *Write the fault tree in standard output format.* For the case of computer manipulation, the fault tree is stored in a data file with a standard format. For example, the fault tree in Figure 8d can be expressed with the data file presented below:

$X1(+1)$	OR	**1	**2	$X2(+1)$
**1	OR	$X4(+10)$	$X5(+10)$	
**2	AND	$X5(+1)$	$c1(0)$	
$X2(+1)$	OR	$X7(+10)$	X3(-1)	
$X5(+10)$	OR	$X6(+10)$		
$X5(+1)$	OR	$X6(+1)$		
$X3(-1)$	OR	$X8(-10)$		

Event-tree Analysis

For event-tree analysis, there are also two critical tasks that can be performed automatically: event-tree synthesis and accident sequence enumeration. Algorithms have already been developed for these purposes in previous studies (Hwang, 1993; Kuo, 1995). Following are some practical issues concerning their implementation.

Representation and ratification of the events in event trees

The convention used for representing events in event trees is the same as that adopted in fault trees. Thus, the corresponding ratification procedure is unchanged.

Identification of terminal nodes

An event tree is completed as long as all its outputs are "consequences," that is, events associated with terminal nodes in the system digraph. These nodes can be identified by comparing the list of all nodes in the system digraph with a list of input nodes. The latter contains all elements in the node list except those in the first column. The list of terminal nodes can then be established by eliminating the nodes in the latter list from the former.

Calculation of output deviation

In developing an event tree, the outputs to an intermediate event can be determined by identifying the corresponding output nodes in the digraph. The deviation values of the output can be computed on the basis of the following equation:

$$\text{Output deviation} = \text{input deviation} \times \text{gain}.$$

Compilation of event-tree structures

A set of standard structures has also been proposed to develop event trees under different conditions (Hwang, 1993; Kuo, 1995). Since these results were published in less accessible formats, they are presented in Figures 9a, 9b, and 9c for the sake of completeness. These different event-tree configurations—structures A, B, and C—are applicable when the output nodes are located on a tree, a FFLP, and a FBLP, respectively. Again, in the case of protection and process loops, the users are required to supply process-specific information concerning the net effects of external disturbances.

Construction procedure

The construction process is illustrated with the following example.

Example 5. Let us consider the digraph presented in Figure 10a. The initial event chosen in this case is $X5(+1)$. Following are the steps needed to build an event tree.

1. Check to see if the current event is a consequence.

In the beginning stage, the current event is the initial event specified by the user. As the event tree becomes larger, the current event is an undeveloped event assigned by the system. The present step is accomplished simply by checking whether the corresponding node is a terminal. If it is, then the next available undeveloped event is assigned as the current event and step 1 is repeated. Since the terminal nodes in Figure 10a are $X4$, $X6$, and $X7$, the event $X5(+1)$ can be developed further.

2. Identify the allowed output nodes.

In the case of building a fault tree, the unallowed events are identified and removed after connecting the current event with a standard structure. However, this process should be

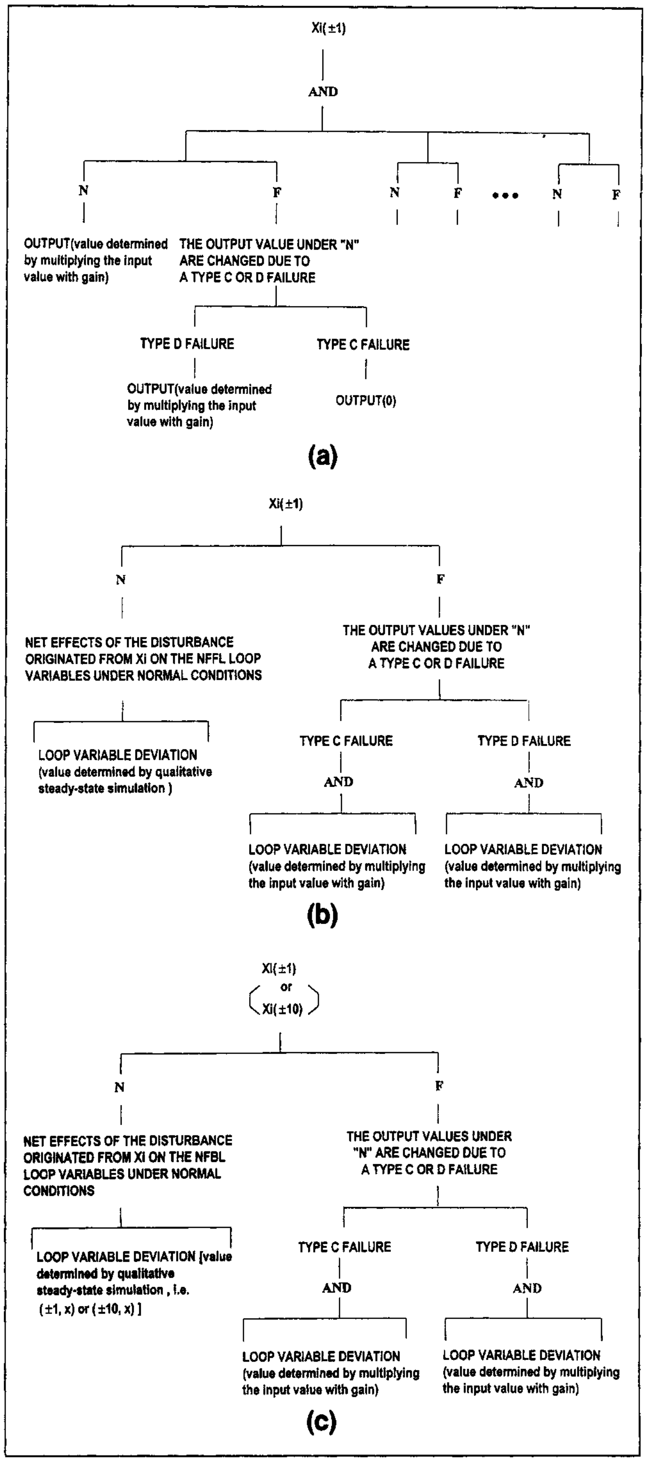
Xi(±1)
AND
N
F
N
F
•••
N
F
OUTPUT(value determined by multiplying the input value with gain)
THE OUTPUT VALUE UNDER "N" ARE CHANGED DUE TO A TYPE C OR D FAILURE
TYPE D FAILURE
TYPE C FAILURE
OUTPUT(value determined by multiplying the input value with gain)
OUTPUT(0)
(a)
Xi(±1)
N
F
NET EFFECTS OF THE DISTURBANCE ORIGINATED FROM Xi ON THE NFFL LOOP VARIABLES UNDER NORMAL CONDITIONS
THE OUTPUT VALUES UNDER "N" ARE CHANGED DUE TO A TYPE C OR D FAILURE
LOOP VARIABLE DEVIATION (value determined by qualitative steady-state simulation)
TYPE C FAILURE
TYPE D FAILURE
AND
AND
LOOP VARIABLE DEVIATION (value determined by multiplying the input value with gain)
LOOP VARIABLE DEVIATION (value determined by multiplying the input value with gain)
(b)
Xi(±1) or Xi(±10)
N
F
NET EFFECTS OF THE DISTURBANCE ORIGINATED FROM Xi ON THE NFBL LOOP VARIABLES UNDER NORMAL CONDITIONS
THE OUTPUT VALUES UNDER "N" ARE CHANGED DUE TO A TYPE C OR D FAILURE
LOOP VARIABLE DEVIATION [value determined by qualitative steady-state simulation, i.e. (±1, x) or (±10, x)]
TYPE C FAILURE
TYPE D FAILURE
AND
AND
LOOP VARIABLE DEVIATION (value determined by multiplying the input value with gain)
LOOP VARIABLE DEVIATION (value determined by multiplying the input value with gain)
(c)

Figure 9. Generalized event-tree structure for: (a) tree-like digraphs—structure A; (b) FFLPs—Structure B; (c) FBLPs—Structure C.

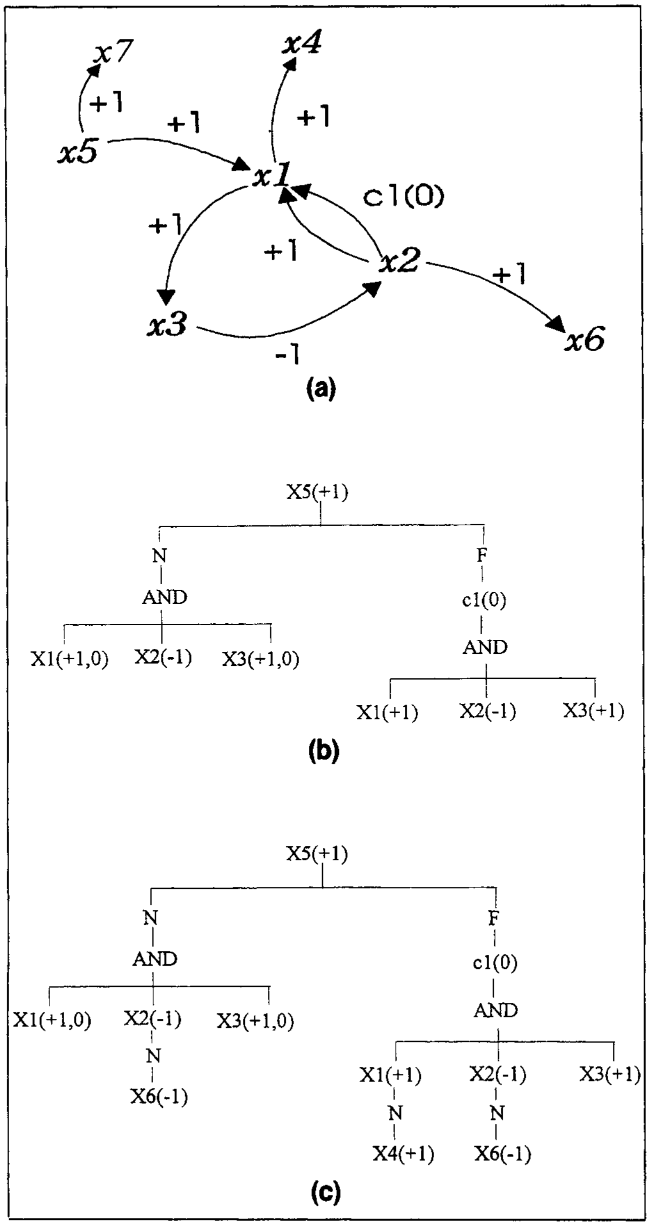

Figure 10. Example 5: (a) fictitious system in Example 5; (b), (c) development process of the event tree (i), (ii), respectively.

reversed in constructing the event trees. Specifically, an output node is selected if the deviation value associated with the current event is one of its allowable input values given in the corresponding row of the gain list. For example, if the gain list of the digraph model presented in Figure 10a is the one presented below, then $X7$ should *not* be an allowable output of $X5$.

$X1$	$X2$	$+1$	$+1$	$+10$	-1	-10	$c1+0$
$X1$	$X5$	$+1$	$+1$	$+10$	-1	-10	
$X2$	$X3$	-1	$+1$	$+10$	-1	-10	
$X3$	$X1$	$+1$	$+1$	$+10$	-1	-10	
$X4$	$X1$	$+1$	$+1$	$+10$	-1	-10	
$X6$	$X2$	$+1$	$+1$	$+10$	-1	-10	
$X7$	$X5$	$+1$	$+10$				

3. Connect the current event with a proper tree structure.

Considering only the allowable output nodes, a proper structure can be selected to expand the event tree. In the present case, a structure B can be connected to the current event $X5(+1)$. The result is presented in Figure 10b. It should be noted that, since the net effects of the current event on the FBLP $X1 \rightarrow X3 \rightarrow X2 \rightarrow X1$ are specified directly in the tree, there is no need to search for the inconsistent events.

4. Identify and record the undeveloped events.

In principle, all outputs of the newly expanded part of an event tree should be added in the list of undeveloped events. However, there are two exceptions. First, if the deviation associated with an output is zero, then the event should not be developed further. Second, since the loops are essentially treated as pseudonodes in developing event trees, the events corresponding to loop nodes without external outputs should not be considered as undeveloped events. For example, the undeveloped events in Figure 10b should be $X1(+1)$ and $X2(-1)$. In other words, the events $X1(+1, 0)$, $X3(+1, 0)$, and $X3(+1)$ should be considered as terminal events.

The undeveloped events are recorded in a list. Initially, the list contains only the initial event. Each time a new branch of the event tree is connected, a new row of undeveloped events may be added to the list.

5. Check if the construction process can be terminated and assign the current events.

The event tree is completed when there are no undeveloped events remaining in the list. Otherwise, the undeveloped events in the next available row will be treated one-at-a-time as the current event and processed according to steps 1 to 4. The completed event tree for the present example is presented in Figure 10c.

6. Write the event tree in standard output format.

For the ease of computer manipulation, the event trees are recorded in standard data files. For example, the event tree in Figure 10c can be represented with the data presented below:

$X5(+1)$	[N]	**1			
**1	[N]	$X1(+1, 0)$	$X2(-1, -1)$	$X3(+1, 0)$	
**1	$c1(0)$	[F]	X1(+1)	X2(-1)	X3(+1)
$X2(-1)$	[N]	$X6(-1)$			
$X1(+1)$	[N]	$X4(+1)$			

Accident-sequence enumeration procedure

After obtaining an event tree in standard format, the task of enumerating accident sequences is simply to concatenate and condense information stored in different rows of the data file. Note that, other than the initial event, the elements in the first column, that is, the input events, also appear as the outputs after [N] or [F] in other rows. To concatenate the

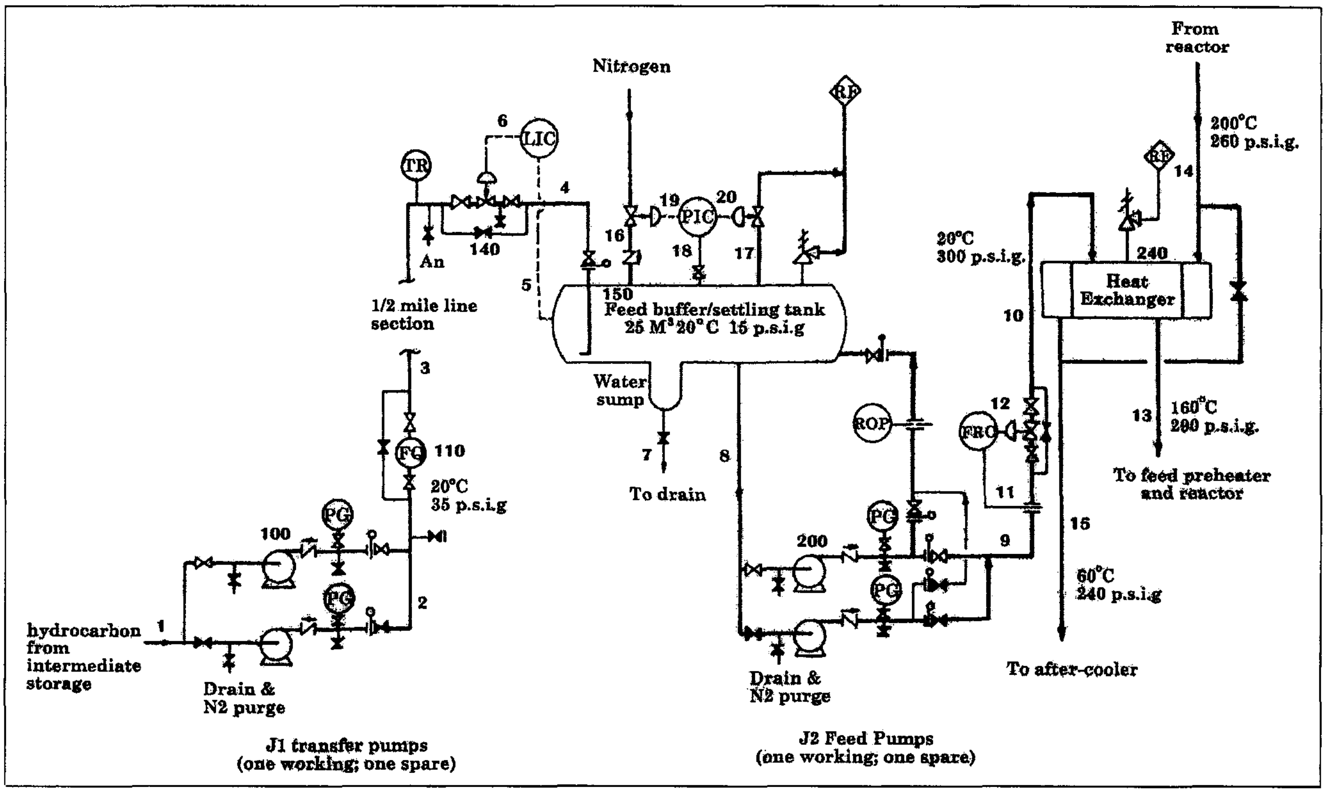

Figure 11. P&ID of a section of olefine dimerization process.

rows, the outputs in each row should be used to replace their inputs appearing in other rows (as an output) and the original row in which these outputs exist should be eliminated after substitution. If there are failure conditions specified before the symbol [F] in the original row, the condition should also be carried into the concatenated row. This substitution process is continued until only the initial event remains as the input event in all the concatenated rows. Although each row represents an accident sequence at this point, the results should be further condensed to a more concise form. Notice that the output events in each row may not be associated with terminal nodes. Since we are mainly interested in the detrimental outcomes of initial events, the intermediate events should be deleted from the accident sequences.

In our example, the accident sequences of the event tree shown at the end of the previous subsection were

$$\{X5(+1) \mid X6(-1)\}$$

and

$$\{X5(+1)c1(0) \mid X4(+1)X6(-1)\}.$$

Remedial-action Knowledge Base

Having obtained the causes—cut sets—and the corresponding consequences—accident sequences—of a deviation, the next task in HAZOP is to work out a proposal concerning the remedial actions required to lower the frequency of the causes or reduce the severity of the consequences. In the former case, the suggestions may be changes in design, operation, maintenance procedure, or even management policy of the plant. In the latter case, improvements in the protective system and/or the emergency response program are usually considered.

A simple rule-based expert system is used in IHAS for generating the list of remedial actions automatically. These conclusions are obtained *directly* from a set of premises: the causes, consequences, and safeguards. In other words, the procedure involves only one-step inference and the remedial-action knowledge base works simply like a look-up table. The rules are stored in the knowledge base in standard format. An example follows.

Example 6. Two cases are presented in this example. The first is concerned with the possible cause of a deviation "more flow," that is, LCV bypass open in error (*byfo*). The corresponding rule is written as

[IF]
byfo
[END]
[THEN]
Institute locking off procedure for bypass when not is use
[END]

Similarly, the same format can be used to express rules for reducing the severity of consequences. Let us consider the consequence of high-input flow rate to a buffer tank, that is, overfilling (*oln*). The rule can be represented as

[IF]
oln
[END]
[THEN]
Install high-level alarm
Check sizing of relief against overfilling
[END]

Note that, since both the causes and consequences are either events corresponding to the primal and terminal nodes or failures associated with the conditional edges, the premises of rules can always be expressed in terms of the symbols used in digraph model.

Generation of the HAZOP Report

From the previous discussion, it is clear that the tool programs developed in this study are capable of generating a large portion of the information needed in HAZOP analysis. The last function of IHAS is therefore to coordinate the execution sequence of these tools and tailor the results into the specific format of a HAZOP report. Let us use a section of the olefin dimerization process presented in Figure 11 as an example for illustrating the report-generation procedure. This well-known industrial process is the subject of many earlier HAZOP case studies (Lawley, 1974; Venkatasubramanian and Vaidhyanathan, 1994). Consequently, the report generated by IHAS can be compared with available results produced by a team of experts.

Note that the equipments and pipelines in Figure 11 have already been numbered. On the basis of these labels, the component models and their interconnections can be specified interactively using *process reader*. Next, the *digraph synthesizer* can be used to build the system digraph, and the *loop searcher* can then be adopted to identify and classify all the embedded FFLPs and FBLPs.

Having executed the *loop searcher*, the following steps are taken to generate the report.

1. *Select a deviation.* Basically, the deviations associated with every intermediate node, that is, a node that is neither primal nor terminal, should be considered. The allowable deviation values can be found in the gain list. For example, $p3$ (the pressure of line number 3) is an intermediate node, and its allowable deviations are $+1$, $+10$, -1, and -10. Thus the event $p3(+10)$ can be selected.

2. *Determine the guide word.* Unlike the manual HAZOP analysis, a guide word is determined *after* each deviation is selected in IHAS. This is done on the basis of a look-up table in which all proper deviations associated with every guide word are stored. For example, an increase in pressure—$p3(+10)$—is categorized as a possible deviation of the guide word MORE OF in IHAS.

3. *Convert the event symbol of deviation into the contents in the "deviation" column.* An algorithm has been developed to translate the event symbol into colloquial descriptions. The symbol $p3(+10)$ is interpreted as "[line no. 3]: pressure is too high."

4. *Construct a fault tree using the selected deviation as the top event.* The tool program *fault tree builder* can be used for such a task. The fault tree corresponding to $p3(+10)$ follows:

$p3(+10)$	OR	$p2(+10)$	$bfm115(+1)$	$btn115(+1)$
$p2(+10)$	OR	$p1(+10)$	$bvfcdn100(+1)$	$rpm100(+10)$

5. *Determine the minimum cut-sets of this fault tree.* The tool program *cause finder* can be used. The minimum cut-sets of the fault tree just given are $\{p1(+10)\}$, $\{bvfcdn100(+1)\}$, $\{rpm100(+10)\}$, $\{bfm115(+1)\}$, and $\{btn115(+1)\}$.

6. *Convert the event symbols in cut-sets into the contents in the "causes" column.* The same algorithm used in step 3 can be used to translate the event symbols into colloquial descriptions. For example, the event $p1(+10)$ can be interpreted as "[line no. 1]: pressure is too high."

7. *Construct the event tree corresponding to a minimum cut-set.* There is only one basic event in the minimum cut-set that is represented by a primal node in the digraph. This event is used as the initial event for building an event tree. *Event-tree builder* is the tool for this purpose. For example, the event tree corresponding to the initial event $p1(+10)$ follows:

```
p1(+10 [N] p2(+10)
p2(+10) [N] p3(+10)
p3(+10) [N] p4(+10)
p4(+10) [N] **1
**1 [N] p150(+10, +1) ss18(+10, +1) s20(+10) m17(+10) rvp210(0)
**1 cts160 [F] p150(+10, +1) ss18(0) s20(0) m17(0) rvp210(+10)
**1 ccs180 [F] p150(+10, +1) ss18(+10, +1) s20(0) m17(0) rvp210(+10)
**1 crp210 [F] p150(+10, +1) ss18(+10, +1) s20(+10) m17(+10) rvp210(0)
**1 crs210 [F] p150(+10, +1) ss18(+10, +1) s20(+10)m17(+10) rvp210(0)
**1 cts160 crp210 [F] p150(+10) ss18(0) s20(0) m17(0) rvp210(0)
**1 cts160 crs210 [F] p150(+10) ss18(0) s20(0) m17(0) rvp210(+10)
**1 ccs180 crp210 [F] p150(+10) ss18(+10) s20(0) m17(0) rvp210(0)
**1 ccs180 crs210 [F] p150(+10) ss18(+10) s20(0) m17(0) rvp210(+10)
p150(+10) [N] rpbn150(+10) p8(+10)
p150(+10, +1) [N] p8(+1)
p8(+10) [N] p9(+10)
p8(+1) [N] p9(+1)
p9(+10) [N] p10(+10)
p9(+1) [N] p10(+1)
```

8. *Enumerate the accident sequences associated with this event tree.* The *scencario enumerator* described previously can be used to find all possible accident sequences after the initial event occurs. The accident sequences associated with $p1(+10)$ are presented below:

$p1(+10)$	$\mid$	$p10(+1)$			
$p1(+10)$	$cts160$	$\mid$	$p10(+1)$		
$p1(+10)$	$ccs180$	$\mid$	$p10(+1)$		
$p1(+10)$	$crp210$	$\mid$	$p10(+1)$		
$p1(+10)$	$crs210$	$\mid$	$p10(+1)$		
$p1(+10)$	$cts160$	$crp210$	$\mid$	$rpbn150(+10)$	$p10(+10)$
$p1(+10)$	$cts160$	$crs210$	$\mid$	$rpbn150(+10)$	$p10(+10)$
$p1(+10)$	$ccs180$	$crp210$	$\mid$	$rpbn150(+10)$	$p10(+10)$
$p1(+10)$	$ccs180$	$crs210$	$\mid$	$rpbn150(+10)$	$p10(+10)$

9. *Convert the accident sequences into the contents in the "safeguards" and "consequences" columns.* The same algorithm used in step 3 can be used to translate the accident sequences into colloquial descriptions. For example, the sequence

$$\{p1(+10)\ cts160\ crp210 \mid rpbn150(+10)\ p10(+10)\}$$

can be interpreted as

"[line no. 1]: pressure is too high AND [pressure sensor no. 160]: stuck AND [relief valve no. 210]: stuck RESULT IN [settling tank no. 150]: ruptures AND [line no. 10]: pressure is too high."

The "consequences" columns in the HAZOP report should be filled with contents after RESULT IN in this translation. The contents in "safeguards" column, however, are related to only a subset of the events stated before RESULT IN. Specifically, the basic events in the cut-set, that is, those described in the CAUSES column, must be excluded. Thus, "[pressure sensor no. 160]" and "[relief valve no. 210]" should be two possible safeguards preventing the initial event $p1(+10)$ from developing into the consequence $rpbn150(+10)$.

It should also be noted that there are two different ways of recording HAZOP results—cause-by-cause and deviation-by-deviation (CCPS, 1992). Both of them are acceptable in industrial practice. In the former case, the consequences are recorded separately for different causes. But no such distinctions are made in the HAZOP report if the latter method is adopted. Since some of the consequences associated with different causes of a deviation may be the same, reports generated with the latter approach are in general more concise and therefore easier for the readers to comprehend. However, since different causes may also create different effects, the reports recorded in the deviation-by-deviation format may not be logically sound. In this study, the former, more rigorous approach is adopted to produce a preliminary report first, and then condensed to the latter, more readable form.

10. *Produce a list of remedial actions against the events in accident sequences.* This work is done with the simple rule-based expert system in IHAS. Each event in the accident sequence is treated as the possible premiss of a rule. An action or actions can be found if a rule in the knowledge base is applicable. For example, the remedial measures may be:

- $p1(+10)$
  [line 1]: install a pressure indicator and alarm system.
- $cts160$
  [pressure sensor no. 160]: improve the testing, calibration, and maintenace program.
- $crp210$
  [relieve valve no. 210]: improve the testing, calibration, and maintenance program.
- $rpbn150(+10)$
  [buffer tank no. 150]: check the sizing of relief valve.
  [buffer tank no. 150]: install a pressure indicator and alarm system.
- $p10(+10)$
  [line no. 10]: install a pressure indicator and alarm system.

11. *Repeat steps 7 to 10 for all the minimum cut-sets determined in step 6.*

12. *Repeat steps 1 to 10 for other credible deviations.*

Applications

The information needed to run IHAS is:

- The P&IDs of the system under study;
- The cause-and-effect relations among process variables and failures for the equipment that are not included in the component database;
- The net effects of external disturbances on protection and process loops, if these loops exist in the system digraph.

As long as these process-specific data are available, the computer program can be executed within minutes. IHAS has been successfully applied to a number of industrial processes. Due to space limitations, a detailed account of these applications cannot be provided here. Instead, an example has been presented in the SUPPLEMENTARY MATERIAL to illustrate the operating procedures of IHAS for performing the FTA, ETA, and HAZOP analyses.

To assess the practical value of IHAS, the computer-generated results must be compared with those produced manually by experts. As mentioned before, the olefin dimerization plant in Figure 11 is the subject of many HAZOP studies, and thus a realistic version of HAZOP report is available in the literature. Consequently, this report has been used as a reference to evaluate the performance of IHAS in this study. Samples taken from both the reference and computer-generated reports are summarized below. First, a comparison of the HAZOP reports generated by experts and by IHAS concerning the event "no flow in line 4":

(i) Causes of the Deviation "No Flow" in Line 4.

- HAZOP meetings:
  —No hydrocarbon available at intermediate storage.
  —Centrifugal pump no. 100 fails.

—Line blockage, isolation valve closed in error, or LCV fails shut.
—Line fracture.

- IHAS:
  —Centrifugal pump No. 100:
  * centrifugal pump fails.
  * upstream storage tank empty.
  * suction line blocked or isolation valve closed in error.
  * discharge line blocked or isolation valve closed in error.

  —Transfer line No. 115:
  * fracture.

  —Level sensor No. 120:
  * sensor fails high.
  * a large positive drift in zero.

  —Level controller No. 130 (reverse acting):
  * set point change.
  * instrument air pressure is too low AND level sensor No. 120 stuck.

  —A/O control valve No. 140:
  * valve fails shut or isolation valve closed in error.

  —Centrifugal pump No. 200:
  * centrifugal pump fails.
  * suction line blocked or isolation valve closed in error.
  * discharge line blocked or isolation valve closed in error.

  —Level sensor No. 230:
  * sensor fails high.
  * a large positive drift in zero.

  —Level controller, No. 240 (reverse acting):
  * set point change.
  * instrument air pressure is too low AND level sensor No. 230 stuck.

  —A/O control valve No. 250:
  * valve fails shut or isolation valve closed in error.

(ii) Consequences of the Deviation "No Flow" in Line 4.

- HAZOP meetings:
  —Loss feed to reaction section and reduced output.
  —Polymer formed in heat exchanger under no flow conditions.
  —Centrifugal pump No. 100 overheats.
  —Hydrocarbon discharged into area adjacent to public highway.
- IHAS:
  —Centrifugal pump No. 100:
  * centrifugal pump overheats.

  —Transfer line No. 115:
  * process material discharged into operation area.

  —Buffer tank No. 150:
  * tank pressure is too high.

  —Shell-and-tube heat exchanger No. 260:
  * polymer formed in the heat exchanger.

  —Line 13:
  * flow rate is too low.

(iii) Remedial Actions of the Deviation "No Flow" in Line 4.

- HAZOP meetings:
  —Ensure good communications with intermediate storage operator.
  —Install low-level alarm on buffer tank No. 150.
  —Install kickback on centrifugal pump No. 100.
  —Check design of centrifugal pump No. 100 strainers.
  —Institute regular patrolling and inspection of transfer line.
- IHAS:
  —Centrifugal pump No. 100:
  * strengthen the current pump maintenance program.
  * institute a regular testing schedule for spare pumps.
  * ensure good communications with upstream storage operator.
  * install low-level alarm on upstream storage tank.
  * install kickback.
  * check design of pump strainers.
  * provide clear written instructions about when and how to open/close the isolation valves in operator manual.

  —Transfer line No. 115:
  * institute regular patrolling and inspection of transfer line.

  —Level sensor No. 120:
  * institute a regular calibration schedule.
  * strengthen the current maintenance program of level sensors.

  —Level controller (reverse acting) No. 130:
  * strengthen the current maintenance program of level controllers.

  —A/O control valve No. 140:
  * strengthen the current maintenance program of control valves.
  * provide clear written instructions about when and how to open/close the isolation valves in operator manual.

  —Buffer tank (with drain valve) No. 150:
  * check sizing of pressure relief devices and install new ones if necessary.
  * install independent pressure indicator and alarm system.
  * install low-level alarm on buffer tank.
  * install high-level alarm on buffer tank.

  —Pressure sensor No. 160:
  * institute a regular calibration schedule.
  * strengthen the current maintenance program of pressure sensors.

  —Pressure controller No. 180:
  * strengthen the current maintenance program of pressure controllers.

  —Centrifugal pump No. 220:
  * check design specifications of pump suction line.
  * provide the correct pump operation procedure to avoid cavitation
  * strengthen the current pump maintenance program.
  * institute a regular testing schedule for spare pumps.

  —Flow sensor No. 230:
  * institute a regular calibration schedule.
  * strengthen the current maintenance program of flow sensors.

—Flow controller (reverse acting) No. 240:
 * strengthen the current maintenance program of flow controllers.
 * check design of pump strainers.
 * provide clear written instructions about when and how to open/close the isolation valves in operator manual.

—A/O control valve No. 250:
 * strengthen the current maintenance program of control valves.
 * provide clear written instructions about when and how to open/close the isolation valves in operator manual.

Again, following is a comparison of the HAZOP reports generated by experts and by IHAS concerning the event "high pressure in line 4":

(i) Causes of the Deviation "More Pressure" in Line 4.

- HAZOP Meetings:
  —Isolation valve closed in error or LCV closes, with pump No. 100 running.
  —Thermal expansion in an isolated valved section due to fire or strong sunlight.
- IHAS:
  —Line No. 1:
   * pressure is too high.
  —Centrifugal pump No. 100:
   * pump rotation speed is too high.
   * discharge line blocked or isolation valve closed in error.
  —Transfer line No. 115:
   * thermal expansion due to fire or strong sunlight.
   * upstream temperature is too high.

(ii) Consequences of the Deviation "More Pressure" in Line 4.

- HAZOP meetings:
  —Transfer line subjected to full pump delivery or surge pressure.
  —Line fracture or flange leak.
- IHAS
  —Line No. 13:
   * undesirable reactions occur due to high water concentration.
  —Centrifugal pump No. 100:
   * pump overheats.
  —Transfer line No. 115:
   * transfer line subjected to full pump delivery or surge pressure.
   * line fracture or flange leak.
  —Buffer tank (with drain valve) No. 150:
   * liquid overflows (Safeguards: level sensor No. 120, level controller No. 130, A/O control valve No. 140).
   * tank ruptures (Safeguards: level sensor No. 120, level controller No. 130, A/O control valve No. 140, pressure sensor No. 160, A/O control valve No. 170, pressure controller No. 180, relief valve No. 210).

(iii) Remedial Actions of the Deviation "More Pressure" in Line 4.

- HAZOP meetings:
  —Install kickback on centrifugal pump No. 100.
  —Check line, FG, and flange ratings.
  —Reduce stroking speed of LCV if necessary.
  —Install a pressure gauge upstream of LCV:
  —Install an independent pressure gauge on buffer tank.
  —Install thermal expansion relief on valved section.
- IHAS:
  —Line No. 1:
   * Install pressure indicator and alarm system.
  —Line No. 13:
   * install water analyzer and high concentration alarm.
  —Centrifugal pump No. 100:
   * install kickback.
   * provide clear written instructions about when and how to open/close the isolation valves in operator manual.
   * check and ensure the stability of power supply.
  —Transfer line No. 115:
   * install thermal expansion relief.
   * install gas detectors and alarms.
  —Level sensor No. 120:
   * institute a regular calibration schedule.
   * strengthen the current maintenance program of level sensors.
  —Level controllers (reverse acting) No. 130:
   * strengthen the current maintenance program of level controllers.
  —A/O control valve No. 140:
   * strengthen the current maintenance program of control valves.
  —Buffer tank (with drain valve) No. 150:
   * check sizing of pressure relief devices and install new ones if necessary.
   * install independent pressure indicator and alarm system.
   * install high-level alarm on buffer tank.
   * extend the discharge line above tank base.
  —Pressure sensor No. 160:
   * institute a regular calibration schedule.
   * strengthen the current maintenance program of level sensors.
  —A/O control valve No. 170:
   * strengthen the current maintenance program of control valves.
  —Pressure controller (reverse acting) No. 180:
   * strengthen the current maintenance program of level controllers.
  —Pressure relief valve No. 210:
   * Institute a regular testing schedule.

And the final comparison of the HAZOP reports generated by human experts and by IHAS concerning the event "low temperature in line 4" is presented in the sequel:

(i) Causes of the Deviation "Less Temperature" in Line 4.

- HAZOP Meetings:
  — Winter conditions.
- IHAS:
  —Centrifugal pump No. 100:
   * upstream temperature is too low.
  —Transfer line No. 115:
   * winter conditions.

(ii) Consequences of the Deviation "Less Temperature" in Line 4.

- HAZOP Meetings:
  - —Water sump and drain line freeze up.
- IHAS:
  - —Line No. 7:
    - ∗ drain valve freezes up.
  - —Line No. 13:
    - ∗ temperature is too low.
  - —Line No. 15:
    - ∗ temperature is too low.
  - —Buffer tank No. 150:
    - ∗ water sump freezes up.

(iii) Remedial Actions of the Deviation "Less Temperature" in Line 4.

- HAZOP Meetings:
  - —Lag water sump down to drain valve.
  - —Steam trace drain valve and drain line downstream.
- IHAS:
  - —Line No. 7:
    - ∗ steam trace pipeline.
    - ∗ install lagging around pipeline.
    - ∗ install temperature indicator and alarm system.
  - —Line No. 13:
    - ∗ install temperature indicator and alarm system.
  - —Line No. 15:
    - ∗ install temperature indicator and alarm system.
  - —Centrifugal pump No. 100:
    - ∗ ensure good communications with upstream storage operator.
    - ∗ install temperature indicator and alarm system on upstream storage tank.
  - —Buffer tank No. 150 (with drain valve):
    - ∗ install lagging around sump.

From these results, one can see that the computer-generated report is not only correct but also more comprehensive in the sense that many of the possible accidents not included in the reference report can be identified.

On the other hand, it is also apparent that some of the hazards and/or operability problems that can be identified in HAZOP meetings will always be neglected by IHAS. First of all, not all guide words (e.g., OTHER THAN) and thus not all possible deviations can be easily incorporated in the computer program. Second, since the analyses performed in IHAS are essentially P&ID-based, operational and/or management issues that cannot be modeled as failure modes of the equipment are inevitably excluded from consideration. Finally, the potential of practical applications is limited by the inherent nature of digraph models. In particular, it is not convenient to use them for describing dynamic systems in sequential operations; see, for example, Shaeiwitz et al. (1977), and thus the current version of IHAS is incapable of carrying out the *procedure* HAZOP analysis.

From the preceding discussion, one can see that the most appropriate way of using IHAS is to treat it as a design aid. It can be run either before or after the HAZOP meetings. In the former case, since the time spent on identifying "routine" accident scenarios is saved, the experts can concentrate on the rare events that may have serious implications. In the latter situation, IHAS can be utilized as a check against the human-generated reports. Finally, it should be noted that, as a result of building such a system for a particular plant, the process-specific operation experiences and knowledge are transformed into standard forms and become easily accessible. Consequently, IHAS can also be used as an effective tool for training new engineers.

Conclusions

The prototype of an integrated hazard-analysis system IHAS has been developed in this study. Three widely accepted hazard-assessment techniques—FTA, ETA, and HAZOP—can be performed automatically with this software. From the results we obtained in practical applications, one can conclude that the quality of hazard analysis can be improved significantly if IHAS is used as an *aid* to the experts.

Acknowledgment

This work was supported by the National Science Council of the ROC government under Grant NSC83-0421-P006-005Z.

Notation

aia	= sudden change in the instrument air pressure
alk	= control valve leaks
atd	= sensor failure of type A (drift in the zero)
bcvfc	= control valve fails to close (type B failure)
bfrn	= thermal expansion in transfer line due to a local fire
bsfc	= switch failure of type B
btd	= type B sensor failure
btn	= temperature of upstream surge tank is too high
bvfcup	= inlet isolation valve of a pump closed in error
bvfcdn	= outlet isolation valve of a pump closed in error
byfo	= control valve bypass opened in error
ccs	= controller is stuck (type C failure)
crp	= relief valve fails (type C failure)
crs	= relief valve is stuck due to line blockage (type C failure)
css	= sensor is stuck (type C failure)
csvs	= solenoid valve is stuck (type C failure)
cts	= tranducer, i.e., sensor/transmitter, is stuck (type C failure)
cvs	= control valve is stuck (type C failure)
L	= liquid level
$m1$, $m2$, . . .	= mass flow rates in pipelines 1, 2, . . .
$p1$, $p2$, . . .	= pressure in pipelines 1, 2, . . .
rpbn	= settling tank ruptures
rpm	= rotation speed of a pump
rvp	= relief valve position
$s1$, $s2$, . . .	= electric or pneumatic signals in lines 1, 2, . . .
$ss1$, $ss2$, . . .	= sensor signals in lines 1, 2, . . .
$t1$, $t2$, . . .	= temperature in pipelines 1, 2, . . .

Literature Cited

Allen, D. J., "Digraphs and Fault Tree," *Ind. Eng. Chem. Fundam.*, **23**, 175 (1984).

Allen, D. J., and M. S. M. Rao, "New Algorithms for the Synthesis and Analysis of Fault Trees," *Ind. Eng. Chem. Fundam.*, **19**, 79 (1980).

Andow, P. K., "Difficulties in Fault-Tree Synthesis for Process Plant," *IEEE Trans. Rel.*, **R-29**, 2 (1980).

Andow, P. K., "Fault Tree and Failure Analyses: Discrete State Representation Problem," *Trans. Ind. Chem. Eng.*, **59**, 125 (1981).

Andrews, J. D., and J. M. Morgan, "Application of Digraph Method of Fault Tree Construction to Process Plant," *Rel. Eng.*, **14**, 85 (1986).

Andrews, J. D., and G. Brennan, "Application of Digraph Method of

Fault Tree Construction to a Complex Control Configuration," *Rel. Eng. Sys. Saf.*, **28**, 375 (1990).
Camarda, P., F. Corsi, and A. Trentadue, "An Efficient Simple Algorithm For Fault Tree Automatic Synthesis from Reliability Graph," *IEEE Trans. Reliab.*, **R-27**, 215 (1978).
CCPS, *Guidelines for Hazard Evaluation Procedures*, 2nd ed., American Institute of Chemical Engineers, New York (1992).
Chamow, M. F., "Direct Graph Techniques for the Analysis of Fault Trees," *IEEE Trans. Rel.*, **R-27**, 7 (1978).
Chang, C. T., and H. C. Hwang, "New Development of the Digraph-based Techniques for Fault-Tree Synthesis," *Ind. Eng. Chem. Res.*, **31**, 1490 (1992).
Chang, C. T., H. C. Hwang, and D. M. Hwang, "Fault-Tree Synthesis Techniques for Process Systems with Coupled Feedward and Feedback Loops," Int. Conf. Safety, Health, and Loss Prevention in Oil, Chemical, and Process Industries, Singapore (1993).
Chang, C. T., H. C. Hwang, K. S. Hwang, and D. S. Hsu, "The Loop Identification and Classification Algorithms for Digraph-Based Analysis," *Comput. Chem. Eng.*, **21**, 233 (1997).
Chang, C. T., and K. S. Hwang, "Studies on the Digraph-based Approach for Fault-Tree Synthesis 1. The Ratio-Control Systems," *Ind. Eng. Chem. Res.*, **33**, 1520 (1994).
Chang, C. T., D. S. Hsu, and D. M. Hwang, "Studies on the Digraph-based Approach for Fault-Tree Synthesis: 2. The Trip Systems," *Ind. Eng. Chem. Res.*, **33**, 1700 (1994).
Cummings, D. L., S. A. Lapp, and G. J. Powers, "Fault Trees Synthesis from a Direct Graph Modes for a Power Distribution Network," *IEEE Trans. Rel.*, **R-32**, 140 (1983).
Fussell, J. B., "Synthetic Fault Tree Model—A Formal Methodology for Fault Tree Construction," Rep. ANCR 1098, Aerojet Nuclear Company, Nat. Reactor Testing Station, ID (1973).
Galluzzo, M., and P. K. Andow, "Failure in Control System," *Rel. Eng.*, **7**, 193 (1984).
Himmelblau, D. M., *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*, Elsevier, New York (1978).
Hwang, H. S. *Automation Studies of Fault Tree Analysis and Event Tree Analysis*, MS Thesis, National Cheng Kung Univ., Tainan, Taiwan, ROC (1993).
Kelly, B. E., and F. P. Lees, "The Propagation of Faults in Process Plants: 1. Modelling of Fault Propagation," *Rel. Eng.*, **16**, 3 (1986a).
Kelly, B. E., and F. P. Lees, "The Propagation of Faults in Process Plants: 2. Fault Tree Synthesis," *Rel. Eng.*, **16**, 39 (1986b).
Kelly, B. E., and F. P. Lees, "The Propagation of Faults in Process Plants: 3. An Interactive, Computer-based Facility," *Rel. Eng.*, **16**, 63 (1986c).
Kelly, B. E., and F. P. Lees, "The Propagation of Faults in Process Plants: 4. Fault Tree Synthesis of a Pump Changeover Sequence," *Rel. Eng.*, **16**, 87 (1986d).
Khan, A. R., and A. Hunt, "The Propagation of Faults in Process Plants: Integration of Fault Propagation Technology into Computer Aided Design," *Ind. Chem. Eng. Symp. Ser.*, **114**, 35 (1989).
Kumamoto, H., and E. J. Henley, "Safety and Reliability Synthesis of Systems with Control Loops," *AIChE J.*, **20**, 376 (1979).
Kuo, D. H., *A Digraph-based System for Automatic HAZOP Assessment*, MS Thesis, National Cheng Kung Univ., Tainan, Taiwan, ROC (1995).
Lambert, H. E., "Comments on the Lapp-Powers 'Computer-Aided Synthesis of Fault Trees,'" *IEEE Trans. Rel.*, **R-28**, 6 (1979).
Lapp, S. A., and G. J. Powers, "Computer-aided Synthesis of Fault-Trees," *IEEE Trans. Rel.*, **R-26**, 2 (1977).
Lapp, S. A., and G. J. Powers, "Update of Lapp-Powers Fault Trees," *IEEE Trans. Rel.*, **R-28**, 12 (1979).
Lawley, H. G., "Operability Studies and Hazard Analysis," *Chem. Eng. Prog.*, **70**, 105 (Apr. 1974).
Leone, H., "A Knowledged-based System for HAZOP Studies—The Knowledge Representation Structure," *Comput. Chem. Eng.*, **20**(Suppl. A), S369 (1996).
Salem, S. L., G. E. Apostolakis, and D. Okrent, "A New Methodology for the Computer-aided Construction of Fault Trees," *Ann. Nucl. Energy*, **4**, 417 (1977).
Shaeiwitz, J. A., S. A. Lapp, and G. J. Powers, "Fault Tree Analysis of Sequential Systems," *Ind. Eng. Chem. Process Des. Dev.*, **16**, 529 (1977).
Srinivasan, R., and V. Venkatasubramanian, "Petri Net-Digraph Models for Automating HAZOP Analysis of Batch Process Plants," *Comput. Chem. Eng.*, **20**(Suppl. A), S719 (1996).
Venkatasubramanian, V., and R. Vaidhyanathan, "A Knowledge-based Framework for Automating HAZOP Analysis," *AIChE J.*, **40**, 496 (1994).
Weatherill, T., and I. T. Cameron, "A Prototype Expert System for Hazard and Operability Study," *Comput. Chem. Eng.*, **13**, 1229 (1989).

Appendix: Classification of Faults and Failures

The definitions of faults and failures suggested by Himmelblau (1978) are followed in this work. The word *fault* is used to designate the departure from an acceptable range of a measurable process variable or calculated parameter associated with a piece of equipment. *Failure*, on the other hand, is taken to mean complete inoperability of a piece of equipment for its intended purpose. Further, they are classified into four types based on their digraph representations and the patterns of their propagation in the system as follows.

Type A

For faults such as disturbances in the process variables or partial component failures (i.e., degradation in the equipment's performance) such as a small leak or a partial plug in a control valve, the corresponding digraph representation should be a node without inputs. The outward edges of such nodes are directed to process variables. A typical digraph model can be found in Figure A1, where x_1 and x_2 are process variables and f is the fault or failure of Type A. The effects of this type of fault/failure can be determined by assigning a nonzero value (± 1 or ± 10) to f, and the values of the other variables in the digraph can then be evaluated accordingly. Notice that, in analyzing these effects for the purpose of classification, the implied assumption is that no other failures exist simultaneously. Further, it should also be noted that, if *both* x_1 and x_2 are on the same FBL, the value of x_2 can be affected not only by f but also by x_1.

Type B

The digraph configuration of component failures such as sensor failing high or control valve failing to close is actually the same as that of Type A. However, their effects should be analyzed differently. If a failure of Type B (f) occurs and both x_1 and x_2 are variables on the same NFBL, then x_2 is always affected by f alone and should be independent of the input x_1.

Type C

Component failures such as sensor stuck or control valve

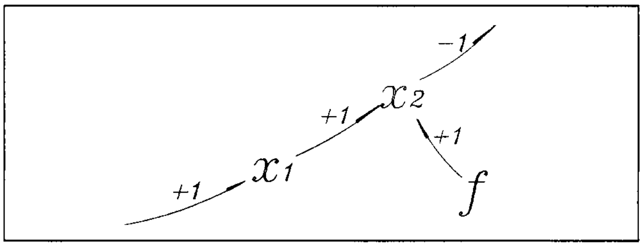

Figure A1. Component failures: types A and B.

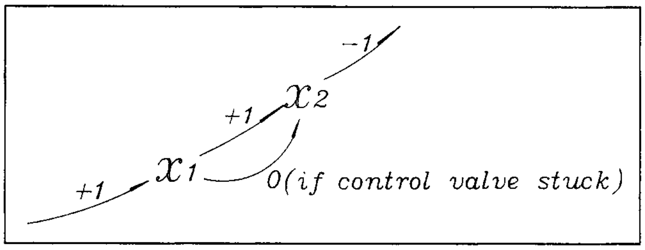

Figure A2. Component failures: type C.

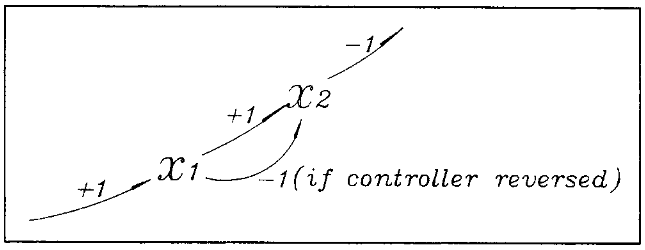

Figure A3. Component failures: type D.

tem digraph, that is, the edge between x_1 and x_2 can be considered as nonexistent. The state variables of the system remain at the normal levels without additional disturbances.

Type D

Component failures such as controller reversed (from direct action to reverse action, or vice versa) or control valve reversed (from air-to-open to air-to-close, or vice versa) can also be represented by conditional edges. An example of such failures is presented in Figure A3, which is also represented by a change in the configuration only. Obviously, the occurrence of a failure of Type D changes the direction of the effects of an additional fault (if it occurs) propagating from x_1 to x_2.

Manuscript received Sept. 9, 1996, and revision received Jan. 13, 1997.

See NAPS document no. 05397 for 25 pages of supplementary material. Order from NAPS c/o Microfiche Publications, P.O. Box 3513, Grand Central Station, New York, NY 10163. Remit in advance in U.S. funds only $7.75 for photocopies or $4.00 for microfiche. Outside the U.S. and Canada, add postage of $4.50 for the first 20 pages and $1.00 for each of 10 pages of material thereafter, $1.50 for microfiche postage.