

A RULE-BASED SYSTEM FOR AUTOMATED BATCH HAZOP STUDIES

C. Palmer[1], P.W.H. Chung[1], and J. Madden[2]
[1]Department of Computer Science, Loughborough University, Leicestershire, LE11 3TU, UK
[2]Hazid Technologies Ltd, Beeston, Nottingham, NG9 2ND, UK

The hazard and operability study technique (HAZOP) is widely-used for identifying potential hazards and operability issues in process plants. To overcome the repetitive and time-consuming nature of the technique, automated hazard identification systems that emulate HAZOPs for continuous plants have been developed. This work considers batch processes, in which material undergoes processing in distinct stages within the plant equipment items according to a set of operating procedures, rather than each equipment item remaining in a "steady state", as is normal for continuously operating plants.

In batch plants deviations which can lead to hazards can arise both from deviations from operating procedures and process variable deviations. Therefore, the effect of operator actions needs to be considered.

CHECKOP is an automated batch HAZOP identification system being developed as a joint project between HAZID Technologies Ltd and Loughborough University. To produce a product in a batch process, a plant operator follows a series of operating instructions. For an operating procedure to be analysed by a computerised system, such as an automated HAZOP system, it must be formally represented. Deviations may be applied to the operating instructions to simulate batch HAZOP. CHECKOP's simulation engine applies the operating instructions to the plant configuration model, which describes the plant connectivity and state Each operating instruction acts to change the state of the plant. The rule-based system tests for potential hazards which result from the operating instructions and their effect upon the plant model.

This paper will focus upon describing the rule-based system. The rules may be categorised according to whether incompatible equipment state or incorrect operation is being investigated. Generic rules may be derived. Examples of incorrect plant operation, which cause the rules to be activated, will be shown. Future development of the system will be described.

INTRODUCTION

A widely used hazard identification technique within the process industry is HAZOP (hazard and operability study). A study considers all possible deviations of a plant from its intended operation, by using deviation guidewords (No, More of, Less of, Part of, Other) applied to each of the process variables (flow, pressure, temperature, etc.) in the plant in turn (Lawley, 1974). The HAZOP of batch processes requires extra time-related or order-related guidewords (Early, Late, Before, After, Quicker, Slower) which are utilised by introducing a particular error into an operating procedure (Bickerton, 2003; Mushtaq and Chung, 2000). The HAZOP technique seeks to identify the hazard or operating consequences

arising from the deviation. Postulating the interaction between deviation, consequence, hazard and performance demands the use of expert engineering knowledge.

To overcome the repetitive and time-consuming nature of the technique, automated hazard identification systems that emulate HAZOPs have been developed. A significant recent innovation in the automation of the continuous plant HAZOP procedure is the Hazid System from Hazid Technologies Ltd. Hazid combines a sophisticated knowledge base with a fault-propagation engine to apply deviations and propagate their effects throughout the complex plant networks.

Much of the research on automated HAZOP identification, based on signed-directed graphs, has concentrated on continuous plants (McCoy et al, 1999a and 1999b; Venkatasubramanian & Vaidhyanathan, 1994). However, very little work has been done in automated hazard identification of batch plants. The limited work done is based on the Petri-net representation (Kang et al, 2003; Srinivasan & Ventkatasubramanian, 1998a and 1998b). In batch processes the plant operation moves through a number of stages, rather than each equipment item remaining in a "steady state", as is normal for continuously operating plants. In batch plants deviations which lead to hazards can arise both from deviations from operating procedures and from process variable deviations. The effects of operator actions need to be considered.

The signed-directed graph approach used in continuous plant HAZOP emulators is found to be unable to capture the information needed to consider the deviations in operating instructions necessary for the HAZOP of batch plants. The main problem is that it does not keep state related information as the HAZOP analysis moves from one operating instruction to the next. A method of representing a sequence of actions (operating instructions) to achieve a state is needed.

This paper describes CHECKOP, a prototype automated batch HAZOP identification system. CHECKOP uses a state-based approach to HAZOP analysis. The paper commences by describing a simple batch plant case study. CHECKOP contains a rule-based system which identifies potential hazards or operability issues. This paper will focus upon discussing the rule-based system. Examples from the case study are used to illustrate the rules.

A SIMPLE EXAMPLE PLANT

In order to demonstrate how CHECKOP is used to aid batch HAZOP of a process, the simple batch processing plant illustrated in Figure 1 is considered. The plant consists of a reactor, two feed tanks and a product tank. The reactor shown produces a product *P* from two reactants *A* and *B*, using the simple chemical reaction $A + B \rightarrow P$. An excess of reactant *B* is used so that reactant *A* is completely consumed in the reactor.

OVERVIEW OF CHECKOP

CHECKOP is being developed as a joint project between Loughborough University and Hazid Technologies Ltd. McCoy et al (2006) describe an early version of the system.

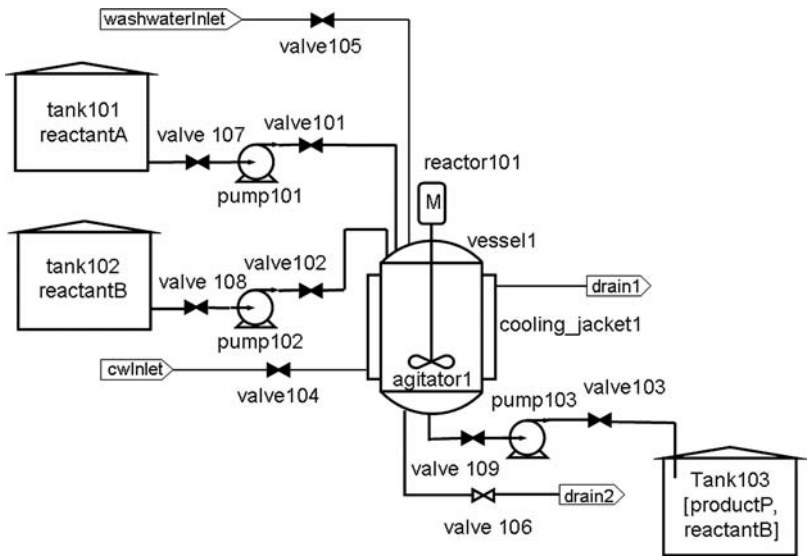

Figure 1. A simple batch processing plant example

To simulate a batch HAZOP CHECKOP systematically applies the HAZOP deviation guidewords to the operating procedure. CHECKOP infers the consequences if a certain instruction in the procedure is not executed, or if the instruction is carried out too early or too late, etc. A report is produced providing warnings against any undesirable situations that may result from the deviations.

CHECKOP's system model consists of four sections:

- an object-oriented plant configuration model
- the operating procedures
- a state-based simulation engine
- a rule-based system

The plant configuration model describes the plant connectivity and state. The topographical relationships of the plant model are derived from the plant piping and instrumentation diagram (P & ID). Initially, the plant is specified to be in its "idle" state with all valves closed and pumps stopped. Operating instructions act to update the states of the equipment items.

To produce a product in a batch process, a plant operator follows a series of operating instructions. For an operating procedure to be analysed by a computerised system, such as an automated HAZOP system, it must be formally represented. Deviations may be applied to the operating instructions to simulate batch HAZOP. The simulation engine

applies the operating instructions to the plant configuration model. Each operating instruction acts to change the state of the plant. The rule-based system tests for potential hazards or operability issues which result from the operating instructions and their effect upon the plant model.

Brief details on the first three sections of CHECKOP's system model are given below. The main focus of this paper, the rule-based system, is described in the next section.

THE PLANT CONFIGURATION MODEL

CHECKOP employs a unit-based object-oriented approach to model plant items and their connectivities, temperatures and pressures. This approach is capable of predicting the dynamic behaviour of the equipment items, in normal operation and under deviation from normal plant operation.

Process plants are built by connecting together smaller sets of units to carry out the required functions. The behaviour of each of these types of units can be modelled generically so that it will apply to any plant in which the unit is used. Each item of equipment in a plant is modelled as an instance of an equipment model, taken from a library of process unit models, which forms a knowledge-base.

The plant is modelled as a unit model which is itself composed of unit models. These unit models are the equipment instances. Instances are assigned a unique identifier. The units can be composed of sub-units. The generic models of the model library may also contain instances. This enables information re-use within the model library. For example, the model library may contain a reactor model, composed of instances of a vessel, an agitator and a cooling jacket. The instances are identified, using object-oriented notation, as: reactor.vessel1, reactor.agitator1 and reactor.coolingjacket1 (see Figure 2).

The unit models contain connections, attributes and actions. Connections enable the ports of equipment instances to be linked together to form a plant model. Each plant equipment instance has a state determined by the current value of its attributes. Actions alter the state of equipment instances by changing attribute values. The connection information and some attributes (e.g. pressures and temperatures) for the plant model are derived from the plant P & ID. Further attribute information and the actions permitted for each equipment item are contained by the generic models of the model library.

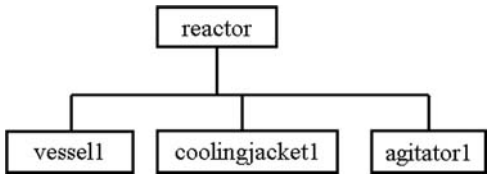

Figure 2. An example unit model

THE OPERATING PROCEDURES

To safely produce a product in a batch process, a plant operator follows a sequence of operating instructions. Each operating instruction directs one or more actions to change the state of the plant. As a plant operation moves through a number of stages, the states of the plant equipment instances are updated. Implicit in the operating instructions is the existence of a complete plant model representation.

An operating procedure is a sequence of operating instructions. For example, to produce productP the simple batch plant would require the following instruction sequence:

```
1) charge reactor101 with reactantA from tank101 until reac-
   tor101.liquid_amount = 30 %vol/vol,
2) operate reactor101.agitator1,
3) cool reactor101.cooling_jacket1 until reactor101.cooling_
   jacket1.liquid_temperature < 25 Celsius
4) charge reactor101 with reactantB from tank102 until reac-
   tor101.liquid_amount = 60 %vol/vol,
5) discharge reactor101 with reactantB, productP to tank103,
6) stop reactor101.agitator1,
7) shut_down reactor101.cooling_jacket1,
```

Each instruction has an implicit set of assumptions about the state of the plant. For example, the instruction "`Charge reactor101 with reactantA from tank101 until reactor101.liquid_amount = 30%vol/vol`" assumes that tank101 is a source of reactantA, is connected via a flow path to reactor101 and contains a sufficient quantity of reactantA to fill reactor101 to a level of 30%. The operating instructions are composed using a set formal template representation. For details of the formal template representation to describe operating procedures see Palmer et al (2006).

An instruction may be an action primitive or be composed of more instructions or action primitives. The following action primitives exist: open, close, operate, run, stop, check and wait. An example of a more complex instruction is "`charge reactor101 with water from washwaterinlet until reactor101.liquid_amount = 65 %vol/vol`" which is composed of the following action primitives:

```
open valve105,
wait until reactor101.liquid_amount = 65 %vol/vol,
close valve105
```

An action primitive operates on an equipment instance. For example, the action primitive "`open valve105`" acts on the equipment instance "valve105". This action primitive creates a flow path between the washwaterinlet and reactor101.

An instruction or action primitive may optionally contain a condition, such as "`until reactor101.level_amount = 30 %vol/vol`", which indicates when completion occurs. Identifiers (e.g. reactor101) link the operating instructions and their constituent action primitives to the instances within the plant configuration model.

THE STATE-BASED SIMULATION ENGINE

Relating the operating instructions to the plant configuration model allows the state of the plant configuration model to be changed at each stage of operation. An action primitive denotes which of the actions of the equipment instance will be executed by the simulation engine. For example, "`open valve105`" indicates that the action "open" contained by the equipment instance "valve105" will be executed.

An action updates the plant model state by changing the attribute values of equipment instances. The new attribute values may be contained by the action model or the condition of the action primitive.

The new plant state resulting from an action may necessitate further changes to the plant model. For example, applying the action "`open valve105`" to the simple batch plant will create a flow path between the washwaterinlet and reactor101 (see Figure 1). This flow path will allow water to transfer from the washwaterinlet to reactor101. After each action is executed the simulation engine tests for the presence of flow paths and updates the attributes describing the contents of connected equipment instances.

THE CHECKOP RULE-BASED SYSTEM

This section describes the CHECKOP rule-based system and explains describes how the rules are structured and categorised. Examples of the rules are given, listed within their categories. Instructions which relate to the simple batch plant are given to demonstrate how the rules are utilised. Examples of incorrect plant operation, which cause the rules to be activated, will be shown.

Given an operating procedure, which may or may not be complete or correct, the rule-based system verifies if it achieves its desired results and does not also lead to any additional, unexpected effects. Any potential problem identified will be reported. Formalising the operating procedure allows alternative orderings of the operating instructions to be considered. This allows the procedure to be modified. Simulation demonstrates the effect of the modified procedure on the plant model. Operating procedure deviations may be generated by automatically applying the HAZOP guidewords to the operating instructions. The system rules capture the important effects of the deviation for hazard reporting.

A rule-based system is well suited to represent the complex, unstructured knowledge required to test for potential hazards or operability issues which result from the operating instructions. The rule-based system is simple to understand and flexible. Existing rules may be changed or new rules may be easily added. This allows the system to be updated if new information becomes available as a plant design develops or to be adapted to specific plant configurations.

A rule consists of the structure "If …Then". For example,

> If the instruction is to charge a reactor and the reactor does not contain space,
> Then indicate a hazard or an operation problem.

If the stipulations of a rule's "If" section are met, the rule is said to be "activated" or "fire".

Three types of rule exist within the CHECKOP rule-base:

1. Generic
2. Specific
3. Independent

A rule may be classified as "generic" if it relates to more than one action primitive or instruction. For example, a generic rule could relate to the action primitives "open" "close" and "operate". Another generic rule could relate to both a "charge" and a "discharge" instruction. Generic rules are not employed to test the operating procedure but form patterns from which rules specific to a certain instruction may be derived. Independent rules do not depend on the structure of another rule. Independent rules relate to the equipment instances referred to by an operating instruction. Specific and independent rules are used to test the operating procedure. Currently the rule-base contains approximately thirty rules.

The rules may be categorised according to whether they investigate:

- incorrect operation
- incompatible equipment state.

These categories may be further divided depending upon whether a rule relates to an action primitive or to a more complex instruction which contain further instructions or action primitives. Rules which test for incorrect operation are applied before the operating procedures are simulated as plant state information is not required. Rules which investigate incompatible equipment state examine the set of assumptions each instruction contains about the state of the plant. As the simulation engine updates the state of the plant model for each instruction encountered, rules which investigate incompatible equipment state must be applied with each instruction.

If one of CHECKOP's rules is activated then a low level warning or a more important error message is issued to indicate a hazard or operation problem depending on how serious its potential effects are. The warning or error message identifies the nature of the hazard or operation problem and whereabouts in the operating procedure the problem occurs. For example,

```
Operation number 2: Charge cannot proceed as reactor101 is
  full
```

The rules described in the following sub-sections issue error messages unless stated otherwise.

RULES INVESTIGATING INCORRECT OPERATION

Incorrect operation occurs when the sequence of operating instructions is performed in the wrong order, an operating instruction is omitted or an extra instruction is executed.

Incorrect operation results from oversights in the operating procedure or operator error. In addition to causing operability issues, incorrect operation may also lead to hazardous consequences.

Rules applicable to action primitives
Rule type 1:
If an action primitive is to be performed and an action primitive which reverses its state does not exist within the operating procedure Then indicate a hazard or an operation problem.
Example rules of this type:

1. If open an equipment instance and a later instruction to close it does not exist within the operating procedure Then indicate a hazard. For example, if the instruction "`open valve101`" is found within the operating procedure, a later instruction "`close valve101`" must also occur within the procedure otherwise this rule will fire. A low level warning is generated if the action primitive to close the equipment instance is not found within the same instruction as the one which contains the action primitive to open the equipment instance but is found later within operating procedure, as this instruction sequence may allow an unexpected or extraneous flow to occur.
2. If operate an equipment instance and a later instruction to stop it does not occur within the operating procedure Then indicate an operation problem. For example if the instruction "`operate pump101.pump_drive`" is found within the operating procedure, a later instruction "`stop pump101.pump_drive`" must also occur within the procedure or this rule will be activated.

Rule type 2:
If an action primitive is to be performed and it reverses the state of the action primitive which occurs immediately prior to it Then indicate an operation problem. Rules of this type issue low level warnings if activated as a problem may not be conclusively indicated.
Example rules of this type:

1. If close an equipment instance and the immediate previous instruction is to open the equipment instance Then indicate an operation problem. For example the instruction couplet,

```
open valve101,
close valve101,
```

would cause this rule to fire.
2. If stop an equipment instance and the immediate previous instruction is to operate the equipment instance Then indicate an operation problem. For example, this rule would be activated by:

```
operate reactor101.agitator1,
stop reactor101.agitator1,
```

Rule applicable to complex instructions

Rule type 3:

If an instruction X occurs within the operating procedure and a previous instruction Y does not occur Then indicate a hazard or an operation problem. For example, if an instruction "`discharge reactor101`" is present in the operating procedure and a previous instruction "`charge reactor101`" does not occur within the procedure. If Y does occur but an instruction occurs in the operating procedure between Y and X which negates the effect of Y Then indicate hazard or an operation problem. For example the following instruction sequence would cause a specific expression of this rule to fire,

```
charge reactor101 from tank101, (instruction Y)
discharge reactor101 to tank103,
shut_down reactor101.cooling_jacket1,
discharge reactor101 to drain2, (instruction X)
```

Example rules of this type:

1. If discharge an equipment instance and a previous instruction to charge the equipment instance does not occur Then indicate an operation problem. Implicit in the "discharge" instruction is the assumption that the equipment instance is discharged until it is empty.
2. If wash an equipment instance (X) and a previous instruction to discharge the equipment instance (Y) does not occur Then indicate a hazard. If a previous instruction to discharge the equipment instance does occur but an instruction to charge the equipment instance occurs in the operating procedure between Y and X Then indicate a hazard. The following sequence of instructions would cause this rule to fire,

```
discharge reactor101 to tank103, (instruction Y)
shut_down reactor101.cooling_jacket1,
charge reactor101 from tank101,
wash reactor101, (instruction X)
```

RULES INVESTIGATING INCOMPATIBLE EQUIPMENT STATE

An incompatible equipment state may occur when an equipment state does not match that given in the operating instruction or conflicts with plant safety. Rules investigating incompatible equipment state compare the state of the plant model with the state of the plant as implied or defined by an instruction. For example the instruction, "`Charge reactor101 with reactantA from tank101 until reactor101.liquid_amount = 30%vol/vol`" implies that tank101 must contain enough reactantA to fill reactor101 to a volume of 30%. This instruction defines that upon completion reactor101 should contain a volume of 30% reactantA. Rules investigating incompatible equipment state fall into two groups: those which are applied before an instruction is simulated; and those applied after its simulation.

RULES APPLIED BEFORE AN INSTRUCTION IS SIMULATED

Rules applicable to action primitives

Rule type 4:

If action primitive brings about an existing equipment state Then indicate an operation problem. This rule indicates an instruction included in the procedure is deemed unnecessary.

Example rules of this type:

1. If operate an equipment instance which is already operating Then indicate an operation problem. For example this rule will fire if the instruction "`operate pump101.pump_drive`" is applied to a pump_drive instance, pump101.pump_drive, which is already of state "operating".
2. If stop an equipment instance which is already stopped Then indicate an operation problem.
3. If open an equipment instance which is already open Then indicate an operation problem.
4. If close an equipment instance which is already closed Then indicate an operation problem.

Rule type 5:

If an action primitive occurs in the operating procedure where the plant is not in an appropriate state for the action to take place then indicate a hazard.

Example rules of this type:

1. If the instruction is to operate an instance of a pump drive, the adjoining suction valve instance is not of a state "open", or the adjoining discharge valve instance is not closed or the following instruction does not open the discharge valve Then indicate a hazard. For example, considering pump101 in the simple batch plant, if an action primitive "`operate pump101.pump_drive`" is found within the operating procedure, the suction valve, valve107, must be open, the discharge valve, valve101, must be closed and the next instruction in the procedure must be "`open valve101`", otherwise this rule will fire.
2. If the instruction is to stop an instance of a pump drive, the adjoining suction valve instance is not of a state "open", or the adjoining discharge valve instance is not closed or the following instruction does not close the suction valve Then indicate a hazard. For example this rule will be activated if an action primitive "`stop pump101.pump_drive`" is to be performed, unless the suction valve, valve107, is open, the discharge valve, valve101, is closed and the next instruction in the procedure is "`close valve101`".

Rules applicable to complex instructions

Rule type 6:

If the material defined in the instruction is not the same as that contained in the source equipment instance Then indicate an operation problem. For example, if the

instruction "`Charge reactor101 with reactantA from tank101`" is to be performed and tank101 does not contain reactant A then a specific version of this rule will be activated. This rule applies to instructions which effect the transfer of material.

Example rules of this type:

1. If the equipment instance supplying the material for a charge instruction does not contain the same material as defined in the instruction Then indicate an operation problem. This rule needs to check the contents of the second equipment item described in the instruction. E.g. for "`charge reactor101 with reactantA from tank101`", checks that tank101 contains reactantA.
2. If the equipment instance supplying the material for a discharge instruction does not contain the same material as defined in the instruction Then indicate an operation problem. This rule verifies the contents of the first equipment instance described in the instruction. E.g. if the instruction occurs "`discharge reactor101 with productP to tank103`" in the operating procedure, checks that reactor101 contains productP.

Rule type 7:

If the source equipment instance does not contain a sufficient quantity of material to fulfil a condition defined in the instruction Then indicate an operation problem. E.g. for the instruction "`charge reactor101 with reactantA from tank101 until reactor101.liquid_amount = 30 %vol/vol`" a specific expression of this rule will test that tank101 contains enough material to fill reactor101 to a volume of 30%. This rule also applies to instructions which effect the transfer of material. Like the previous generic rule described above, two specific expressions of this rule exist: one which tests a charge instruction and one for a discharge instruction.

Rule type 8:

If the capacity of the sink equipment instance is less than the value of a condition defined in the instruction Then indicate a hazard. For the example, in the instruction "`Charge reactor101 with reactantA from tank101 until reactor101.liquid_amount = 15 tonnes`" the volume of reactor101 should be greater than or equal to 15 tonnes. Again, this rule applies to instructions which transfer material between two equipment items and two specific versions to the rule exist: one for a charge instruction and one for a discharge instruction.

Rule type 9:

If the condition of the instruction is not consistent with that of the equipment instance referred to by the condition Then indicate an operation problem. For example, a specific expression of this rule will fire if the instruction "`charge reactor101.vessel1 from tank101 until reactor101.liquid_amount = 30 %vol/vol`" occurs within the operating procedures and the volume of reactor101.liquid_amount is greater than 30%. This rule indicates that an instruction is unnecessary or that it is occurring out of sequence.

Example rules of this type:

1. If the instruction is "charge" and the value of the condition of the instruction is greater than that of the equipment instance referred to by the condition Then indicate an operation problem.
2. If the instruction is "cool" and the value of the condition of the instruction is less than that of the equipment instance referred to by the condition Then indicate an operation problem. For example, this rule will activate if the instruction "`cool reactor101.cooling_jacket1 until reactor101.cooling_jacket1.liquid_temperature < 25 C`" is to be applied to the batch plant and the value of reactor101.cooling_jacket1.liquid_temperature is already less than 25 C.

RULE APPLICABLE AFTER AN INSTRUCTION IS SIMULATED

Independent Rule Applicable to a Complex Instruction:

If a condition defined in an instruction is not achieved Then indicate an operation problem. For example when the instruction "`Charge reactor101 with reactantA from tank101 until reactor101.liquid_amount = 30 %vol/vol`" is complete, if reactor101 does not contain 30% by volume of reactantA this rule will be activated.

CONCLUSIONS AND FUTURE WORK

CHECKOP's plant model captures the state of the plant's equipment items. By simulating the effect of each operating instruction on the plant model the resulting plant state can be demonstrated and compared to the intended state of the procedure. Formalising the operating procedure allows alternative orderings of the operating instructions to be examined. This enables the procedure to be modified for batch HAZOP analysis.

The CHECKOP rule-base verifies that an operating procedure does not lead to unexpected consequences. Hazards or operability issues are detected which arise if the operating procedure is deviated from. Currently most of the rules detect hazards relating to flow. Further rules need to be added to detect hazards relating to pressure and temperature. The rule-base can be extended to identify hazards for specific plant set-ups.

The CHECKOP modelling system needs to be enhanced to capture chemical information in order that the plant model can be updated when a reaction occurs. This will allow the consequences to be detected of incompatible substances meeting or unforeseen chemical reaction occurring. More case studies are required to assess the capabilities of the rule-base, including more complex plants which utilise sequential or parallel processing.

REFERENCES

Bickerton, J., 2003, HAZOP applied to batch and semi-batch reactors, *Loss Prevention Bulletin*, 173: [1] 10–12.

Hazid. Available at <http://www.hazid.com> [Accessed 06/08/07].

Kang, B., Dongil S. and Yoon, E.N., 2003, Automation of the safety analysis of batch processes based on the multi-modeling approach, *Control Engineering Practice*, 11: 871–880.

Lawley, H.G., 1974, Operability Studies and Hazard Analysis, *Chemical Engineering Progress*, 70: [4] 45–56.

McCoy, S.A., Zhou, D. and Chung, P.W.H., 2006, State-based modelling in hazard identification, *Applied Intelligence*, 24: 263–279.

McCoy, S.A., Wakeman, S.J., Larkin, F.D., Jefferson, M., Chung, P.W., Rushton, A.G., Lees, F.P. and Heino, P.M., 1999a, HAZID, A Computer Aid for Hazard Identification 1. The STOPHAZ Package and the HAZID Code: An Overview, the Issues and the Structure, *Transactions of the Institution of Chemical Engineers*, 77: [B] 317–327.

McCoy, S.A., Wakeman, S.J., Larkin, F.D., Chung, P.W., Rushton, A.G. and Lees, F.P., 1999b, HAZID, A Computer Aid for Hazard Identification 2. Unit Model System, *Transactions of the Institution of Chemical Engineers*, 77: [B] 328–334.

Mushtaq, F. and Chung, P.W.H., 2000, A Systematic HAZOP procedure for batch processes, and its application to pipeless plants, *Journal of Loss Prevention in the Process Industries*, 13: 41–48.

Palmer, C., Chung, P.W.H., McCoy, S.A. and Madden, J., A Formal Method of Communicating Operating Procedures, 2006, *Hazards XIX*, IChemE Symposium Series 151: 448–457.

Srinivasan, R. and Ventkatasubramanian, V., 1998a, Automating HAZOP analysis of batch chemical plants: Part I The Knowledge representation framework, *Computers and Chemical Engineering*, 22: [9] 1345–1355.

Srinivasan, R. and Ventkatasubramanian, V., 1998b, Automating HAZOP analysis of batch chemical plants: Part II Algorithms and application, *Computers and Chemical Engineering*, 22: [9] 1357–1370.

Venkatasubramanian, V., & Vaidhyanathan, R., 1994, A knowledge based framework for automating HAZOP analysis, *AIChE Journal*, 40: [3] 496–505.