An automated system for batch hazard and operability studies

An Automated System for Batch Hazard and Operability Studies

C. Palmer[a] and P.W.H. Chung[a]

[a]Department of Computer Science, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK

Tel.: +44-1509-222543; fax: +44-1509-635722

email: p.w.h.chung@lboro.ac.uk, c.palmer3@lboro.ac.uk

corresponding author: P.W.H. Chung

Abstract

A widely used hazard identification technique within the process industry is HAZOP (hazard and operability study). To overcome the repetitive and time-consuming nature of the technique automated systems are being 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 continuous 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. CHECKOP uses a state-based approach to HAZOP analysis. CHECKOP takes a plant description and a set of operating instructions as input and produces a HAZOP report automatically. The overall system architecture and the details of the major components of the systems will be described. Examples of incorrect plant operation along with the resulting output generated by CHECKOP will be shown. The advantages and limitations of CHECKOP will be discussed.

KEYWORDS: batch HAZOP, batch process, operating procedures, operating instructions, qualitative simulation

1 Introduction

To overcome the repetitive and time-consuming nature of the HAZOPs technique automated systems are being developed. Much of the research on automated hazard identification so far has concentrated on continuous plants by considering the causes and consequences of deviations from steady state (McCoy et al [1, 2]; Venkatasubramanian & Vaidhyanathan [3]; Zhang et al [4]). HAZID Technologies Ltd successfully market the tool "HAZID", a commercial automated hazard identification system for continuous plants.

The signed-directed graph approach used in automated identification systems for continuous plants is found to be unable to capture the information needed to consider the deviations in operating instructions necessary for the HAZOP of batch plants (Srinivasan and Ventkatasubramanian [5]). In batch processes the plant operation moves through a number of stages, rather than each equipment item remaining in a "steady state" indefinitely, as is normal for continuous plants. In a batch process a fixed quantity of material (a batch) is produced by subjecting quantities of input materials to a sequence of processing actions using one or more pieces of equipment. The changes in equipment state over time means that batch processes require the state of the plant to be expressed. A method of representing a sequence of actions (operating instructions) to achieve a state is needed. The role of human factors in operability analyses needs to be addressed (Kariuki & Löwe [6]; Colombo & Demichela [7]).

To produce a product in a batch process, a plant operator follows a series of operating instructions. In order for an operating procedure to be analysed by a computerized system, such as an automated HAZOP identification system, it must be formally represented. A knowledge representation is needed for the actions and quantities involved. The representation must be able to capture the idea of a sequence of operating instructions – the operator actions which should be performed in order. An ontology is needed to categorise plant components, operations, and equipment attributes. Instructions written in natural language require complex analysis algorithms and their meaning may also be ambiguous. Therefore, there is a need to develop operating instruction formats that are both intuitive for humans and unambiguous for the computer. Deviations may be applied to the operating instructions to simulate batch HAZOP.

Each operating instruction acts 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. A rule-based system is used to test for potential hazards which result from the operating instructions and their effect upon the plant model.

This paper commences by presenting an overview of the HAZOP technique. The batch HAZOP technique is explained. A brief review of the current methods used to automate batch HAZOP is given, showing that only limited research is done on automated batch HAZOP and more research is needed. The most widely used method is not flexible enough to allow modifications to the plant or to the operating procedure to be easily depicted. The CHECKOP system is described. CHECKOP presents a new, state-based approach to HAZOP analysis. The tool's system architecture is outlined and each section of the architecture is discussed. Some examples of the results generated by CHECKOP are given. Limitations of the work are considered and possibilities for future work discussed.

2 HAZOP Studies

This section describes the HAZOP technique. A brief review of the current methods used to automate batch HAZOP is given. Some conclusions are advanced on the tools surveyed.

HAZOP studies are widely used for identifying hazard and operability problems. The HAZOP technique applies guide words (which relate to process deviations) to plant diagrams on a line by line basis to identify causes and consequences of the process deviations. The list of guide words includes No, More of, Less of, Part of, Other, etc. (Lawley [8]). Deviations are derived from a combination of guidewords and process variables, e.g. if the guide word is "More" and the variable is "level" then the combination of the two gives "More level". A line consists of a set of equipment items which perform a processing function, e.g. heating section, reaction section, etc. Consequences of particular interest are potential releases of hazardous process materials from equipment and/or failure of an equipment item to perform its processing function.

All HAZOP studies are traditionally carried out manually by a 5-8 man team of experienced engineers. The engineers base the study on the detailed plant diagrams (P & ID) which provide a complete

definition of the plant and control configuration. The team examines every plant item and considers in turn all potential process deviations from normal operation and the plant fault which could give rise to the deviation is recorded. They then propagate the effects of the deviation through the plant by tracing through the diagrams until some item is reached where experience indicates a consequence could ensue and the consequence is recorded. The risk posed by the consequence is evaluated and if it is judged to be unacceptable, remedial action is recommended. All the faults, consequences, risks and actions are recorded in a formal HAZOP report.

HAZOP of batch plant is difficult because the operating state of the plant undergoes changes at each stage of the batch process according to the operating procedures. In addition to the process variables covered in a standard HAZOP study batch processes require the extra dimensions of time and sequence control to be considered. Possible cross contamination between products and the potential for charging the wrong material also need to be considered (Bickerton [9])

As well as applying guidewords to the process variables as for a continuous process, HAZOP analysis of a batch process must also take into account the "deviations" from operating instructions. Besides the basic set of deviation guidewords used in continuous processes the HAZOP of batch processes requires extra time-related guidewords. Bickerton [9] , Mushtaq & Chung [10] and Trucco & Leva [11] define several extra guidewords shown in table 1.

(table 1 around here)

The guidewords are utilised by introducing a particular error into an operating procedure. An example instruction is given below to charge a reactor in a simple batch plant. The plant contains a reactor, reactor101, which is filled from a feed tank, tank101. A pump assembly consisting of a suction valve, valve107, a pump, pump101, and a discharge valve, valve101, is situated between reactor101 and tank101. In the example the guide words *no* and *before* are applied systematically to generate different versions of an operating procedure which deviate from the normal procedure.

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

```
Charge reactor101 with reactantA from tank101 until
reactor101.level_amount = 30%
```

```
(1) open valve107
(2) operate pump101
(3) open valve101
(4) wait until reactor101.level_reading = 30% vol/vol
(5) close valve101
(6) stop pump101
(7) close valve107
```

systematically removing one instruction at a time from the procedure will result in seven different operations.

For example, procedure with instruction 1 omitted:

```
(2) operate pump101
(3) open valve101
(4) wait until reactor101.level_reading = 30% vol/vol
(5) close valve101
(6) stop pump101
(7) close valve107
```

Procedure with instruction 2 omitted:

```
(1) open valve107
(3) open valve101
(4) wait until reactor101.level_reading = 30% vol/vol
(5) close valve101
(6) stop pump101
(7) close valve107
```

The modified operating procedures created by applying the deviation guideword may be incorrect and lead to mal-operation or dangerous situations. For example, the procedure with instruction 1 omitted will never reach completion (reactor101.level_amount = 30%) since there is no material flow into reactor101 as valve107 may be closed.

Now, considering the guide word *before*, when it is applied to the operation, actions are moved earlier in the procedure. For example, moving the instruction "`operate pump101`" one step forward will result in the procedure:

```
(2) operate pump101
(1) open valve107
(3) open valve101
(4) wait until reactor101.level_reading = 30% vol/vol
(5) close valve101
(6) stop pump101
(7) close valve107
```

Again, the deviation from the original operating instruction may lead to an undesirable or hazardous situation. Batch HAZOP identifies potential hazardous problems due to procedural error, i.e. when the operator fails to follow a set of instructions as intended.

Surveying current automated batch HAZOP tools, the following techniques were found to be in use:

1. Petri nets
2. Statecharts
3. Mini fault trees
4. Expert systems

A Petri net is a directed graph containing two types of nodes: places and transitions. Places and transitions are connected via arcs. A place forms the input place to a transition if it is connected to the transition by a directed arc. A place is the output place of a transition if there is a directed arc connecting the transition to the place. A place may hold any number of tokens. Tokens in input places enable a transition to fire. When the transition is fired the input place tokens are redistributed and the net achieves a new state. The features of Petri net modelling are described by Malhota [12] and Dutuit [13]. An overview of Petri nets and their applications in batch processes is given by Gu and Bahri [14].

Kang, Shin and Yoon [15] have developed a prototype tool utilizing the basic Petri net structure within a multiple modelling approach. Basnyat, Chozos and Palanque [16] build a system model with Petri

nets to test performance of task models. Wang, Wu and Chang [17] use a timed Petri net approach. Time constants are associated with transitions. The only commercial automated batch HAZOP tool, PHASuite (Zhao, Bhushan, and Ventkatasubramanian [18, 19]), uses the Petri net technique. A similar approach is employed in PHASuite to that of Kang, Shin and Yoon [15].

Statecharts are state-transition-systems. Music and Matko [20] describe the basic differences between statecharts and Petri nets. Asral and Suzuki [21] propose a model to be used in safety analysis of a batch plant. Plant behaviour is modelled using statecharts. Graf and Schmidt-Traub [22] enhance the basic statechart technique. The statecharts are modelled as a hierarchy.

Mini fault trees simulate the propagation of faults. In fault tree analysis an undesired event is taken as the root ("top event") of a tree of logic. Then, each event which could cause that event is added to the tree as a series of logic expressions. A mini fault tree applies to an equipment unit. Bartolozzi et al [23] apply this technique using two types of model: operation and equipment. Separate models are used to describe each phase an equipment unit passes through as it functions. The equipment qualitative models developed are similar to those proposed by Lees et al [24, 25].

An expert system consists of two principle components: a knowledge-base and an inference engine which is used to reach the solution to a problem. The expert system of Shimada et al [26] comprises a generic knowledge-base, a plant specific knowledge-base and an inference engine. Yet-Pole [27] has developed an automated HAZOP system for the semiconductor manufacturing industry.

In common with the applications surveyed CHECKOP employs a unit-based modelling approach. Petri nets are the most widely used technique. However, they do not allow a flexible enough representation of the alternative operating sequences which arise from deviations from intended operations, or where some part of the plant is modified. McCoy et al [28] have considered the use of Petri nets in hazard identification systems. They found that if the order of two operating instructions is exchanged, or if a new equipment item is added to the plant, it is very difficult to modify the Petri net to take account of this.

All of the tools surveyed are designs or prototypes and are unable to perform a fully automated batch HAZOP. Even the commercial tool, PHASuite [18, 19], requires further model development to be fully operational. The system of Bartolozzi et al [23] unable to record state-related information as the HAZOP analysis moves from one operating instruction to the next. Some of the tools are unable to simulate a complete batch HAZOP as deviations are not applied to the operating instructions (Zhao, Bhushan, and Ventkatasubramanian [18, 19]; Shimada et al [26]; Yet-Pole [27]). Instead Shimada et al [26] consider operating time as a process variable. Yet-Pole [27] relates operations to equipment.

Several of those tools surveyed which construct a plant model suffer from complexity. McCoy et al [28] and Dutuit et al [29] found that even for a simple plant the Petri net approach is complex and difficult to interpret. Applying deviations to operating instructions requires the plant model to be redrawn. Bartolozzi et al. [23] rebuild the tree model of the plant Zhao, Bhushan, and Ventkatasubramanian (18, 19) employ a complicated translation algorithm to transfer information between causal (signed directed graphs) and state (Petri nets) models.

Due to the time-consuming and labour-intensive nature of the technique automated HAZOP identification systems have been developed. Much of the research on automated HAZOP identification, based on signed-directed graphs, has concentrated on continuous plants (McCoy et al [1, 2]; Venkatasubramanian & Vaidhyanathan [3]; Zhang et al [4]. Relatively little work has been done in automated hazard identification of batch plants. Possible reasons for this emphasis are that:

- the automated HAZOP study technique was developed originally for continuous processes
- the inherent extra complexity due to the requirement of keeping state information after each operation in batch processes.

More research is required to achieve similar automation benefit for the HAZOP of batch processes as now exists for continuous.

3 CHECKOP System Architecture

CHECKOP aims to identify the result of deviations from operating procedures. These deviations may lead to mal-operation of the process plant or cause potentially hazardous situations. CHECKOP inputs consist of a plant description file of the plant under study derived from the plant piping and instrumentation diagram (P&ID), a library of generic plant models and the plant operating procedure.

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 comprises 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 description file. 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.

Initially, the plant is specified to be in its "idle" state with all valves closed and pumps stopped. The operating instructions act to update the states of the equipment items. 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 describing CHECKOP's plant configuration model are given below. The representation structure used to model the operating instructions; how the simulation engine enacts the effects of the operating instructions upon the plant model; and the structure of the CHECKOP rule-base, which tests for potential hazards resulting from the operating instructions, are explained in more detail. Some example results generated by CHECKOP are given.

4 CHECKOP Plant Configuration Model

CHECKOP employs a unit-based object-oriented approach to model plant items and their connectivities. CHECKOP's system design has been developed in UML. UML is a notation-based language developed for use in the design and development of object-oriented systems.

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 the library of generic plant 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 1).

(figure 1 around here)

Consider a plant which contains two reactor equipment items, reactor101 and reactor102. When the reactor model is instantiated within the plant as an equipment item, e.g. reactor101, the instances are copied into the plant item. The instance identifiers are updated. Reactor101 will contain the instances reactor101.vessel1, reactor101.agitator1 and reactor101.coolingjacket1. Reactor102 will be composed of reactor102.vessel1, reactor102.agitator1 and reactor102.coolingjacket1.

The unit models contain connections, attributes and actions (see figure 2). 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. Attributes consist of a name, operator, value and an attribute type. The type describes allowed values. In addition, the type may also describe possible value units. The allowed values may be discrete, e.g. {open, closed}, {operating, stopped}, or be chosen from a continuous numerical range, e.g. 0 – 100 %, -273 – infinity Celsius. Legitimate operators are "<", ">", "=" and "approx". Operators can be combined in the normal way as

commonly found in formal languages, e.g. programming languages. The use of operators enables imprecise values to be modelled. Attributes with a discrete type are constrained to the operator "=". Examples of attributes are:

valve_state = closed (discrete attribute type)

temperature < 50 Celsius (continuous attribute type).

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 around here)

5 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 an action 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. Below is an example of an operating procedure for a simple batch plant. The plant consists of a water-cooled reactor, two feed tanks (tank101 and tank102) and a product tank (tank103). The reactor produces a product P from two reactants A and B, using the simple chemical reaction A + B → P. An excess of reactant B is used so that reactant A is completely consumed in the reactor. The assumption is made that cooling is required to aid the reaction. Reactant A is transferred to the reactor from tank101 through pump101. Reactant B is transferred to the reactor from tank102 through pump102. After the reaction product P is discharged from the reactor to tank103 via pump103 and the reactor is washed. Reactor101 may be flushed with nitrogen by opening valve111 and vented to atmosphere by opening valve112.

Reactor101 is composed of the subunit instances reactor101.vessel1, reactor101.agitator1 and reactor101.coolingjacket1. Pumps 101, 102 and 103 are centrifugal pumps. Each pump has connected

suction and discharge valves. For example, pump 101 is connected to suction valve, valve107, and discharge valve, valve101. Valve105 controls the flow of water between a washwaterinlet and reactor101.

(Figure 3 around here)

To produce productP the simple batch plant would require the following instruction sequence:

1) `flush reactor101 with nitrogen`
2) `vent reactor101`
3) `charge reactor101 with reactantA from tank101 until reactor101.liquid_amount = 30 %vol/vol,`
4) `operate reactor101.agitator1,`
5) `cool reactor101.cooling_jacket1 until reactor101.cooling_jacket1.liquid_temperature < 25 Celsius`
6) `charge reactor101 with reactantB from tank102 until reactor101.liquid_amount = 60 %vol/vol,`
7) `discharge reactor101 with reactantB, productP to tank103,`
8) `stop reactor101.agitator1,`
9) `shut_down reactor101.cooling_jacket1,`
10) `wash reactor101 with water from washwaterinlet`

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%. CHECKOP's rule-based system verifies these assumptions.

In order to avoid the difficulties and ambiguities caused by natural language, the operating procedures are composed using a formal template representation which forms the syntax for the operating procedures. The structure of a template is:

Action Item1 *with* Material *Filler-word* Item2 *until* Condition

For example, considering instruction 1 shown above, "charge" is the action, "reactor101" is Item1, "reactant1" is the material, "from" is the Filler-word, "tank101" is item2 and "liquid_amount = 30% vol" is the condition.

More generally, the "Action" is the operating procedure activity, e.g. "operate", "close", "check", etc. "Item" is the equipment instance undergoing the Action, e.g. pump101, valve103, etc. The "Condition" is an ordered triplet consisting of: variable1, operator, variable2. Variable1 is the name of the attribute that needs to be monitored. The operator comprises of one of the following: "<", ">", "≤", "≥", "=", "≈". The operator "=" signifies that the plant reading, varying around a set point, is accepted at this value. Variable2 is the value which terminates the Action. Attributes are described using an object-oriented notation. For example, the attribute attribute "liquid_amount" of reactor101 is described as "reactor101.liquid_amount".

The template components are ordered. An instruction written using the template must contain an "action" component. All the other components are optional. Some examples of the template in use are shown in table 2. For more details of the formal template representation describing operating procedures see Palmer et al [30].

(table 2 around here)

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.

It can be seen that table 2 contains a mixture of complex instructions and action primitives. "Cool", "wash", "charge" and "discharge" are examples of more complex instructions.

The *Item1* and *Item2* template components are identifiers linking the operating instructions and their constituent action primitives to the plant configuration model. The identifier (e.g. reactor101) indicates which equipment instance the instruction or action primitive refers to.

6 The State-based Simulation Engine

A HAZOP study consists of a qualitative analysis of the effect of deviations to major process variables. HAZOP considers if these deviations result in potential hazards or operability problems. The HAZOP team must know the state of the plant (i.e. which equipment is operating and which is idle) and also must take into account process flows and conditions (e.g. temperature, pressure, etc.) A batch plant has several different states as the batch process is executed. The assumption is made that the plant is in a suitable state for the execution of each of the stages of the operating procedure.

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. Complete dynamic quantitative simulation of the process plant based on complex equations, as performed by commercial batch simulators, is not required as there is no necessity to calculate the plant heat and mass balances. CHECKOP simulates the effect of the operating procedure upon the plant model.

CHECKOP tests that the expected plant state as defined by the operating procedure is achieved at each procedural stage. The operating procedure provided is not necessarily correct. Simulation will confirm if the operating procedure achieves its desired results and does not also lead to any additional, unexpected effects.

To model a batch process three types of simulation are required:

1. reaction
2. mass transfer
3. heat transfer.

Reaction may be physical or chemical. Physical reaction consists of a phase change, e.g. from a solid to a liquid or a liquid to a gas. During a chemical reaction one or more of the chemical species present are changed into one or more new species.

Mass transfer sub-divides into the transfer of:

1. gas
2. liquid
3. vapour
4. two phases
5. powder
6. solids
7. paste

The simulation of gas transfer will consider the changes in pressure and gaseous material composition with the plant model. Simulating liquid transfer models changes in liquid material composition and quantity. The interaction of liquid transfer with gas transfer needs to be considered. A vapour consists of a liquid entrained within a gas. Modelling the transfer of a vapour must consider liquid condensation. Two phase transfer may be further categorised into the transfer of liquid/vapour and gas/ powder. An example of a liquid/vapour phase is the top extract from a distillation column. An example of a gas/powder phase is when pressurised air is forced into powder to effect transfer. A powder is composed of fine solid particles and has the property of flow. Batch process solids consist of discrete particles, which are larger than those of a powder, e.g. pellets. Solids are often transferred via a chute. A paste is a solid which becomes liquid under pressure.

Simulating heat transfer looks at changes in temperature within the plant model. Modelling the transfer of heat as a result of gas, liquid or vapour transfer may be required.

Currently CHECKOP's simulation engine is limited to modelling gas and liquid transfer. The simulation engine implements the action primitives and more complex instructions defined within the operating procedures. Each operating instruction acts to change the state of the plant model. The rest of this section describes how the information contained in the operating instructions is used to update plant model attributes. The simulation of action primitives and more complex instructions composed of action primitives is discussed. The plant states required to enable gas and liquid transfer are considered.

6.1 Simulating Action Primitives

An action primitive denotes which of the actions of an equipment instance will be executed. For example, "`open valve104`" indicates that the action "open" contained by the equipment instance "valve104" will be executed. The action updates the plant model state by changing an attribute value of the equipment instance. The new attribute value may be specified within the condition of the action primitive or contained by the action model as a default to be applied if no condition is defined by the action primitive. For example, the action primitive "`check valve106 valve_state = closed`" contains a condition defining that the valve_state attribute should have the value "closed". The action primitive "`open valve104`" does not contain a condition so the default "`valve_state = open`" contained by the action model "open" is used to update equipment instance valve104.

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 described in section 5 creates a liquid flow path between the washwaterinlet and reactor101 (see figure 3). This flow path will allow water to transfer from the washwaterinlet to reactor101.

6.2 Simulating Complex Instructions

If an action primitive is a member of a more complex instruction the simulator tests if the new plant state brought about by the action primitive will allow the conditions defined by the grouping complex instruction to be achieved. If these conditions can be realized the simulator updates the attributes of the equipment instances described in the complex instruction. For example, when the instruction

"`charge reactor101 with water from washwaterinlet until reactor101.liquid_amount = 65 %vol/vol`" is simulated, if the constituent action primitives enable a liquid flow path between the washwaterinlet and the reactor the simulator will update the liquid_amount attribute of reactor101 to a value of 65 %vol/vol.

Currently the batch simulator models the effects of the following complex instructions: flush, vent, charge and discharge. Modelling "flush" and "vent" requires simulating gas transfer. Modelling "charge" and "discharge" requires simulating liquid transfer. The following sub-sections describe the plant conditions required for gas and liquid transfer and how the instructions entailing gas and liquid transfer are simulated.

In instructions involving mass transfer if the conditions defined by the instruction are to be realized a physical connection path must exist between the equipment items of the instruction in a given direction. The direction is implied by the instruction. For example in the instruction provided above, reactor101 must be connected downstream of the washwaterinlet.

The search algorithm utilised by CHECKOP for identifying a physical connection path consists of locating the next equipment instance linked in the direction indicated to Item1 of the instruction, then the next instance linked to this is found, etc. The search continues in this manner until a given equipment item is located. For descriptions of more detailed algorithms which allow paths to be identified in plants containing equipment items with multiple connections see Palmer [31] and Drath et al [32].

6.3 Simulating Gas Transfer Instructions

For gas transfer to occur a flow path must exist between a source of a gas and a gas container or outlet. For a flush operation to take place, e.g. "`flush reactor101 with nitrogen`", *Item1* of the instruction must be physically connected to a gas source and a flow of gas must exist between *Item1* and the gas source. For a vent operation to occur, e.g. "`vent reactor101`", *Item1* must be physically connected to a gas outlet and a flow of gas must exist between *Item1* and the gas outlet. For a flow of gas to exist between *Item1* and a gas source, *Item1* must be connected to a gas source and all valves located on the connection path must be open. For a flow of gas to exist between *Item1* and a gas

outlet, *Item1* must be connected to a gas outlet and all valves located on the connection path must be open. See section 5 for a description of the simple batch plant to which the instruction examples apply.

To test if the conditions of a gas transfer instruction can be achieved the batch simulator checks if a flow of gas exists. If a flow path does exist an attribute of *Item1* is updated. If the instruction contains a "wait" action primitive the new attribute value of *Item1* is set to that specified by the "wait" condition. For example, in the instruction given below the simulator will update the attribute "vapour_pressure" of reactor101 to 3 BarG.

```
flush reactor101 with nitrogen
     [
          open valve111
          wait until reactor101.vapour_pressure = 3 BarG
          close valve111
     ]
```

If the instruction does not contain a "wait" action primitive item the simulator updates *Item1* with the condition defined in the gas transfer operation or a default value if no condition is given.

If gas flow still exists after the plant state defined by the instruction is achieved, a further update to the plant model is required. For a flush operation the operator of the *Item1* attribute is set to greater than, e.g. reactor101.vapour_pressure > 3 BarG. For a vent operation the operator of the *Item1* attribute is set to less than, e.g. reactor101.vapour_pressure < 3 BarG.

6.4 Simulating Liquid Transfer Instructions

For a liquid transfer to happen a flow path must exist between a source of liquid and a sink. The source must have a replacement substance, i.e. be connected downstream of another source of mass. The sink must be connected to a mass displacement outlet. In a charge operation, e.g. "`charge reactor101 with reactantA from tank101 until reactor101.liquid_amount = 30 %vol/vol`" *Item2* of the instruction (e.g. tank101) forms the source and *Item1* (e.g. reactor101) the sink. In a discharge operation, e.g. "`discharge reactor101 with reactantB, productP to tank103`", *Item1* of the instruction (e.g. reactor101) forms the source and *Item2* (e.g.

tank103) the sink. See section 5 for a description of the plant to which these example instructions apply. The simulator tests if the source and sink as defined by an instruction are valid. In order for an equipment item to be a valid source it must contain a quantity of liquid. To be a valid sink an equipment item must have spare liquid capacity.

The batch simulator examines if the source has a replacement substance. The existence of a flow path between the source and a vent to atmosphere is tested for. The simulator also tests if the sink is connected to a displacement outlet, by considering if there is a flow path between the sink and a vent to atmosphere. A simplifying assumption is made that a vent to atmosphere will supply a replacement substance and a means of mass displacement. Details of how a gas flow path is detected are given in the previous section.

For instructions requiring liquid transfer to be realized a source of liquid must be physically connected to a sink and a flow of liquid must exist between the source and sink equipment items. For a liquid flow to exist between the source and the sink all valves located on the connection path must be open and all the pumps located on this path must be operating.

If the batch simulator discovers a liquid flow to exist the plant model is updated. If the instruction contains a "wait" action primitive the new plant attributes values are defined by or calculated from the value specified by the "wait" condition. If the instruction does not contain a "wait" action primitive item the simulator uses the condition defined in the liquid transfer operation to update the plant attribute values or a default value if no condition is given.

For example, in the instruction given below the simulator will use the value "300 litres" defined in the condition of the "wait" action primitive to update plant attribute values.

```
charge reactor101 with water from washwaterinlet
      [
      open valve105,
      wait until reactor101.liquid_amount = 300 litres,
      close valve105
```

]

For the instruction "`discharge reactor101 with reactantB, productP to tank103`" a default value of reactor101.liquid_amount = 0% vol/vol is employed. The assumption is made that, when no other information is available, the reactor is discharged until empty.

To simulate a liquid transfer operation, as well as liquid amount attributes, plant attributes describing material composition also require updating. The simulator updates the sink material composition with that of the source. For example, for the instruction "`charge reactor101 with reactantA from tank101`" the simulator adds reactantA to the material composition list of the sink. If the amount of liquid contained by the source is found to be zero the simulator removes the material composition information from the source.

If liquid flow still exists after the plant state defined by the instruction is achieved, a further update to the plant model is required. The operator of the attribute describing the amount of liquid contained by the sink is set to greater than. The operator of the attribute describing the amount of liquid contained by the source is set to less than.

7 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 shown in figure 3 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 an operating 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 "fired".

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 more complex 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 forty rules.

The rules may be broadly 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. At the present, these rules only consider equipment states resulting from changes in pressure and material flow due to the limitations of the simulation engine.

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:

```
ERROR: In operation number 3 Charge cannot proceed as reactor101 is
full
```

The rules described in the following sub-sections issue error messages unless stated otherwise. The rules form a knowledge-base. This knowledge-base has been verified by expert engineers.

7.1 Rule 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.

7.1.1 Rule 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`" is found within the operating procedure, a later instruction "`stop pump101`" must also occur within the procedure or this rule will be activated.

7.1.2 Rule Applicable to Complex Instructions

Rule type 2:

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 drain2,
```

```
shut_down reactor101.cooling_jacket1,
discharge reactor101 to tank103, (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)
```

7.2 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.

7.2.1 Rules Applied before an Instruction is Simulated

7.2.1.1 Rules Applicable to Action Primitives

Rule type 3:

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. These rules generate low level warnings as the main adverse effect is simply a waste of an operative's time.

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`" is applied to a pump instance, pump101, 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 4:

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`".

7.2.1.2 Rule Applicable to Complex Instructions

Rule type 5:

If the material defined in the instruction is not the same as that contained in the liquid 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`", a check is made 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 "`discharge reactor101 with reactantB,productP to tank103`" occurs within the operating procedure, a check is made that reactor101 contains reactantB `and` productP.

7.2.2 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.

8 CHECKOP Results

To demonstrate the capabilities of CHECKOP the operating procedure given in section 2 (HAZOP) is applied to the simple batch plant. To generate some example batch HAZOP results the three deviations mentioned in that section are introduced into the operating procedure. The results output are shown below.

To introduce the first deviation the guideword *no* is applied to the procedure by omitting the first instruction. CHECKOP generates the following output:

```
Advisory Warning: valve107 has the action "close" within operation
number 1 but is already "closed"
ERROR: In operation number 1 pump101 has the action "operate" but the
upstream valve "valve107" is closed
ERROR: In operation number 1 pump101 has the action "stop" but the
upstream valve "valve107" is closed
ERROR: Operation number 1 has not achieved
"reactor101.level_reading = 30 %vol/vol".
Plant state is "reactor101.level_reading = 0 %vol/vol".
```

The advisory warning results from a rule of type 3 firing (see section "Rules Investigating Incompatible Equipment State"). The action referred to (`close valve107`) brings about an existing state. The first error is produced by example rule 1 of type 4 firing. The plant is not in an appropriate state for the action "`operate pump101`" to occur. When operating pump101 the suction valve, valve107, must

be open. The assumption is made that the valve is closed as the plant is initially specified to be in its "idle" state with all valves closed and pumps stopped.

The second error is due to example rule 2 of type 4 firing. The plant is not in an appropriate state for the action "`stop pump101`" to take place. When stopping pump101 the adjoining suction valve must be open.

The final error generated for this deviation results from the independent rule applicable to a complex instruction firing (see section "Rule applicable after an Instruction is Simulated"). The condition of "`reactor101.level_reading = 30 %vol/vol`" has not been achieved since there was no material flow into reactor101 as valve107 was closed. A flow into reactor101 was not discovered during simulation, hence the contents of reactor101 were not updated within the plant model.

The second deviation also applies the guideword *no* to the procedure and is produced by omitting the second instruction. The second deviation generates:

```
Advisory Warning: pump101 has the action "stop" within operation
number 1 but is already "stopped"
ERROR: Operation number 1 has not achieved
"reactor101.level_reading = 30 %vol/vol".
Plant state is "reactor101.level_reading = 0 %vol/vol".
```

As in the first deviation the advisory warning results from a rule of type 3 firing (see section "Rules Investigating Incompatible Equipment State") but in this case the action referred to is "`stop pump101`". The assumption is made that when the plant is in its initial state its pumps are not operating. The error is created when the independent rule is triggered (see section Rule applicable after an Instruction is Simulated). The condition of "`reactor101.level_reading = 30 %vol/vol`" has not been achieved since there was no material flow into reactor101 as pump101 was not operating.

The third deviation applies the guideword *before* to the operating procedure by moving the second instruction (`operate pump101`) sequentially one step forward in the procedure. CHECKOP generates the following error:

```
ERROR: In operation number 1 pump101 has the action "operate" but the
upstream valve "valve107" is closed and the next instruction in the
procedure does not open the downstream valve "valve101"
```

This error is produced by example rule 1 of type 4 firing (see section "Rules Investigating Incompatible Equipment State"). The plant is not in an appropriate state for the action "`operate pump101`" to occur. When operating pump101 the suction valve, valve107, must be open and the following instruction must be to open the discharge valve, valve101.

9 System Assessment

In developing CHECKOP four main contributions have been made by this project:

1. Developing a modelling system able to represent the range of equipment items which comprise a process plant and capture the state of the equipment items.
2. Creating an action model representation that covers a wide range of operations and the ability for grouping sets of operations.
3. Creating a state-based simulation engine which simulates the effect of the operating procedure upon the plant model.
4. Building a rule-based system to tests for potential hazards or operability issues which result from the operating instructions and their effect upon the plant model.

More research is needed in the area of automated batch HAZOP identification. Ways in the method defined in this paper may be further developed are:

- Further use of guidewords
- Integrating with the commercial tool HAZID

- Enhancing the rule-base
- Further development of the state-based simulation engine
- Representing concurrent and parallel processing

9.1 Further Use of Guidewords

CHECKOP utilises the guidewords *no*, *before* and *after* (see section 2 for a description of batch HAZOP guidewords). Further research is required into the use of existing guidewords and how to apply more guidewords.

No is straightforward – there are only a limited number of ways of applying this guideword. *Before* and *after* present more difficult decisions – how far is it permissible to move the action? Without a sensible limit on this, exhaustive examination could bury the user in unwanted detail.

The next step is to consider the guidewords *early* and *late*, where the action is performed for a longer or shorter time than intended. This is particularly applicable for those actions which have an "until" clause (e.g. what happens if a reactor is only filled to 10% instead of 30%?). Variants of the *other* guide word should also be taken into account. This would include considering what happens if the action is performed on the wrong piece of equipment (e.g. what if the wrong pump is switched off?) or if the wrong material is used within a process. Care needs to be taken to ensure that the most sensible and likely deviations are created for a given guide word, as the potential number of variations for even a single action step is huge.

9.2 Integration with HAZID

Future research will consider integrating CHECKOP with HAZID Technologies' system for the automated HAZOP of continuous plants. CHECKOP considers the role of operating procedures within the batch HAZOP process. CHECKOP does not take into account equipment failure and the process variable deviations derived from this. In batch plants deviations which can lead to hazards can arise both from deviations from operating procedures and process variable deviations. To undertake a complete batch HAZOP study CHECKOP needs to be integrated with HAZID. Integrating CHECKOP with HAZID will enable the evaluation of hazards at any given stage in the batch process. This

integration will allow semi-continuous plants and continuous plants during the start-up and shut-down phases to be analysed.

9.3 Enhancing the Rule-base

Most of the rules detect hazards relating to flow. Further rules need to be added to detect hazards relating to pressure, temperature and reaction when the simulator is enhanced. The rule-base can be extended to identify hazards for specific plant set-ups.

9.4 Further Development of the State-based Simulation Engine

Currently CHECKOP's simulation engine is limited to modelling gas and liquid transfer. These algorithms may be enhanced to consider a pressure differential as a condition of the flow path. A negative pressure differential will cause the direction of flow to reverse. The present liquid transfer algorithm tests that pumps located on a physical connection path between equipment items are operating. However, pressure differences between the equipment items may also arise due to differences in height or applied vacuums. The liquid transfer algorithm requires amending to take account of these types of pressure differences. The simulation of the other types of mass transfer, heat transfer and reaction needs to be considered (see section 7). Simulating reaction may require more complex modelling and simulation of fluid compositions.

9.5 Representing Concurrent and Parallel Processing

Concurrent processing occurs when multiple versions of the same operation takes place at the same time in different equipment items. For example, if a plant contained two reactors both reactors could be charged at the same time.

Parallel processing occurs when two instructions take place within the same time period. For example, in the operating procedure given in section 5 the instruction "`cool reactor101.cooling_jacket1 until reactor101.cooling_jacket1.liquid_temperature < 25 Celsius`" occurs at the same time as "`charge reactor101.vessel1 with reactantB from tank102 until reactor101.liquid_amount = 60 %vol/vol`". The "`cool`" operation does not complete until the operation "`shut_down reactor101.cooling_jacket1`" is achieved. The

operation "`shut_down reactor101.cooling_jacket1`" forms the second half of the the "`cool`" operation.

CHECKOP generates superfluous warnings when instructions occur in parallel. At the present CHECKOP models the operating instructions sequentially and is unable to represent the concept of more than one instruction occurring within the same time period. It may be necessary to consider that operating instructions have state. For example, an operating instruction could be "active" or "complete".

10 Conclusions

A need for further research in the area of automated batch HAZOP has been identified. An automated batch HAZOP identification system, CHECKOP, has been described. CHECKOP's plant model describes connectivity and state of the plant being analysed. 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 permits alternative orderings of the operating instructions to be examined, allowing the procedure to be modified for batch HAZOP analysis.

CHECKOP will enable:

- Rigorous verification of operating procedures
- Identification of mal-operation problems
- A HAZOP to identify process hazards at any stage (normal or mal-operated) of operation.

Future research will consider integrating CHECKOP with HAZID Technologies' system for the automated HAZOP of continuous plants, enabling the evaluation of hazards at any given stage in the batch process. This integration will allow semi-continuous plants and continuous plants during the start-up and shut-down phases to be analysed. Most of the rules detect hazards relating to flow.

Further rules need to be added to detect hazards relating to pressure, temperature and reaction when the simulator is enhanced. Currently CHECKOP's simulation engine is limited to modelling gas and liquid transfer. These algorithms may be enhanced to consider a pressure differential as a condition of the flow path.

Acknowledgements

This work is supported by the Engineering and Physics Research Council, U.K. and by Hazid Technologies LTD.

References

[1] McCoy, S, Wakeman, SJ, Larkin, FD, Jefferson, M, Chung, PWH, Rushton, AG, Lees, FP and Heino, PM. 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, 1999; 77 (Part B), p. 317-327.

[2] McCoy, SA, Wakeman, S, Larkin, FD, Chung, PWH, Rushton, AG and Lees, FP. HAZID, A Computer Aid for Hazard Identification 2. Unit Model System, Transactions of the Institution of Chemical Engineers, 1999; 77 (Part B), p. 328-334.

[3] Venkatasubramanian, V and Vaidhyanathan, R. A knowledge based framework for automating HAZOP analysis, AIChE Journal, 1994; 40 (3), p. 496-505.

[4] Zhang, Z, Wu, C, Zhang, B, Xia, T and Li, A. SDG multiple fault diagnosis by real-time inverse inference, Reliability Engineering & System Safety, 2005; 87(2), p. 173-189.

[5] Srinivasan, R and Ventkatasubramanian, V. Automating HAZOP analysis of batch chemical plants: Part I The Knowledge representation framework, Computers and Chemical Engineering, 1999; 22(9), p. 1345-1355.

[6] Kariuki, SG, and Löwe, K. Integrating human factors into process hazard analysis, Reliability Engineering & System Safety, 2007; 92(12), p. 1764-1773.

[7] Colombo, S and Demichela, M. The systematic integration of human factors into safety analyses: An integrated engineering approach, Reliability Engineering & System Safety, 2008; 93(12), p. 1911-1921.

[8] Lawley, HG, Operability Studies and Hazard Analysis, Chemical Engineering Progress, 1974; 70 (4), p, 45-56.

[9] Bickerton, J. HAZOP applied to batch and semi-batch reactors, Loss Prevention Bulletin, 2003; 173, (1), p. 10-12.

[10] Mushtaq, F and Chung, PWH. A Systematic HAZOP procedure for batch processes, and its application to pipeless plants, Journal of Loss Prevention in the Process Industries, 2000; 13, p. 41-48.

[11] Trucco, P and Leva, MC. A probabilistic cognitive simulator for HRA studies (PROCOS), Reliability Engineering and System Safety, 2007; 92, p. 1117-1130.

[12] Malhotra, M and Trivedi, KS. Dependability modeling using Petri nets, IEEE Transactions on Reliability, 1995; 44 (3) p. 428-440

[13] Dutuit, Y, Châtelet, E, Signoret, JP and Thomas, P. Dependability modelling and evaluation by using stochastic Petri nets: application to two test cases, Reliability Engineering and System Safety, 1997; 55 (2) p. 117–124.

[14] Gu, T and Bahri, PA. A survey of Petri net applications in batch processes, Computers in Industry, 2002; l47, p. 99-111.

[15] Kang, B, Dongil S and Yoon, EN. Automation of the safety analysis of batch processes based on the multi-modeling approach, Control Engineering Practice, 2003; 11, p. 871-880.

[16] Basnyat, S, Chozos, N and Palanque, P. Multidisciplinary perspective on accident investigation, Reliability Engineering & System Safety, 2006: 91(12), p. 1502-1520

[17] Wang, Y, Wu, J and Chang, C. Automatic hazard analysis of batch operations with Petri nets, Reliability Engineering & System Safety, 2002; 76 (1), p. 91-104.

[18] Zhao, J, Bhushan, M and Ventkatasubramanian, V. PHASuite: An Automated HAZOP Analysis Tool For Chemical Processes Part I: Knowledge Engineering Framework, Process Safety and Environmental Protection, 2005a; 83 (B6), p. 509-532, IChemE.

[19] Zhao, J. Bhushan, M and Ventkatasubramanian, V. PHASuite: An Automated HAZOP Analysis Tool For Chemical Processes Part II: Implementation and Case Study, Process Safety and Environmental Protection, 2005b; 83 (B6), p. 533-548, IChemE.

[20] Music, G. and Matko, D. Petri Net Based Control of a Modular Production System. In: Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE '99. Bled, Slovenia, July 12-16. 1999; 3, p. 1383 -1388.

[21] Asral, DR and Suzuki, K. A Concept of Modeling PVC Batch Plant in Object Oriented Approach for Safety Analysis. In: Developments in Applied Artificial Intelligence: 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2003. Loughborough, UK, 2003; June 23-26, 2718, p. 271 – 276.

[22] Graf, H and Schmit-Traub, H. An integrated approach to early process hazard identification of continuous and batch plants with statechart modelling and simulation, Computers and Chemical Engineering, 2001; 25, p. 61 -72.

[23] Bartolozzi, V, Castiglione, L, Picciotto, A and Galluzzo, M. Qualitative models of equipment units and their use in automatic HAZOP analysis. Reliability Engineering and System Safety, 2000; 70, p. 49-57.

[24] Kelly, BE and Lees, FP. The propagation of faults in process plants: 1. modelling of fault propagation, Reliability Engineering, 1986: 16 p. 3-38.

[25] Hunt, A, Kelly, BE, Mullhi, JS, Lees FP and Rushton, AG. The propagation of faults in process plants: 6. Overview of, and modelling for, fault tree synthesis, Reliability Engineering & System Safety, 1992; 39 (2) p. 173-194.

[26] Shimada,Y, Yang, Z, Song, J, Suzuki, K and Sayama, H. Computer-aided Operability Study for Batch Plants. Loss Prevention and Safety Promotion in the Process Industry, 1995; 2, p. 587-598, Mewis, J.J., Pasman, H.J. and de Rademaeker, E.E. (eds), Elsevier Science.

[27] Yet-Pole, I. Development and applications of CASEHAT - A multipurpose computer aided hazard analysis automation system used in semiconductor manufacturing industry. Journal of Loss Prevention in the Process Industries, 2003; 16, p. 271-279.

[28] McCoy, SA, Zhou, D and Chung, PWH. State-based modelling in hazard identification. Applied Intelligence, 2006; 24, p. 263- 279.

[29] Dutuit, Y, Innal, F, Rauzy A and Signoret J-P. Probabilistic assessments in relationship with safety integrity levels by using Fault Trees. Reliability Engineering & System Safety, 2008; 93(12), p. 1867-1876.

[30] Palmer, C, Chung, PWH, McCoy, SA and Madden, J. A Formal Method of Communicating Operating Procedures. In: Hazards XIX, IChemE Symposium Series 2006; 151, p. 448-457.

[31] Palmer, C. Creating Signed Directed Graphs for Process Plants, Ph.D. Thesis, Loughborough University, 1999

[32] Drath, R, Fay A and Schmidberger T. Computer-aided design and implementation of interlock control code. In: IEEE International Symposium on Computer-Aided Control Systems Design, 2006; 2653-2658.

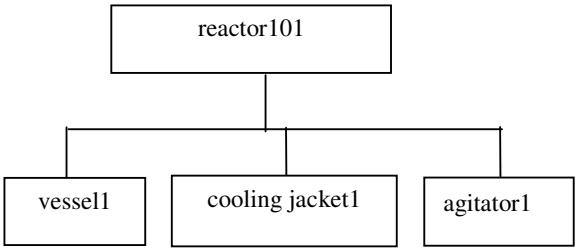

Figure 1. An Example Plant Instance

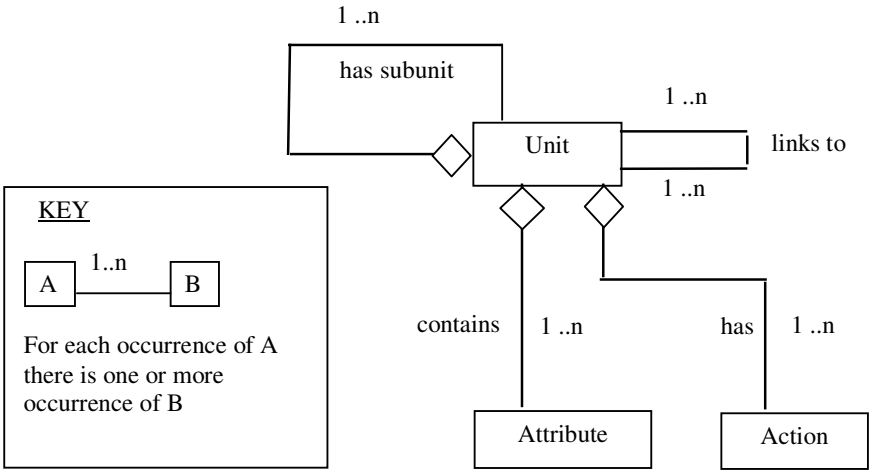

Figure 2. Overview of CHECKOP's Plant Configuration Model

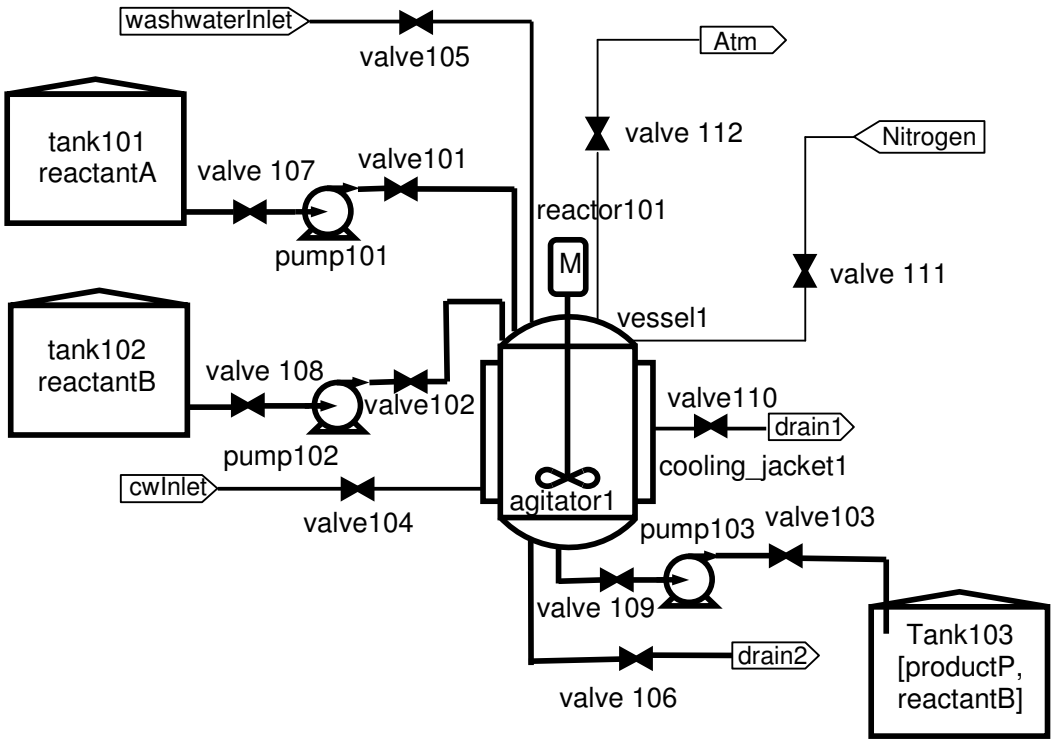

Figure 3. A Simple Batch Processing Plant Example

Guideword	Interpretation
Early	Something happens earlier, relative to the clock, than the design intent.
Late	Something happens later, relative to the clock, than the design intent.
Before	Something happens earlier, in terms of sequence, than the design intent.
After	Something happens later, in terms of sequence, than the design intent.
Quicker	Something happens more quickly than design intent.
Slower	Something happens more slowly than design intent.
Repeated	Something is repeated a second time.
Opposite	Something is performed the opposite way around.

Table 1. Time-related Guidewords for Batch HAZOP

Action	Item1	*with*	Material	*Filler-word*	Item2	*until*	Condition
open	valve 101						
check	valve 102						state = closed
cool	reactor101.jacket1			content		until	reactor101.jacket1.temp < 25 C
wait						until	tank101.level_reading = 0 % vol
wash	reactor101	with	water	from	Wash water inlet		
charge	reactor101	with	reactantB	from	tank 102		
discharge	reactor101			to	tank 102	until	reactor101.liquid_amount = 0 % vol

Table 2 Example instructions in the template format