 © 2006 IChemE

FORMAL METHOD OF COMMUNICATING OPERATING PROCEDURES

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

HAZOP is a widely-used technique for identifying potential hazards in process plants. However, the application of the technique is very repetitive and time consuming. One way to overcome this bottleneck is to develop automated hazard identification systems that emulate the HAZOP technique. 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. This work concentrates on batch processes where a batch plant moves through a number of different stages during operation.

To safely produce a product in a batch process, a plant operator follows a sequence of operating instructions. In order for an operating procedure to be analysed by a computerized system it must be formally represented. Instructions written in natural language require complex analysis algorithms and their meaning may also be ambiguous. However, machine language is incomprehensible to humans. Therefore, there is a need to develop operating instruction formats that are both intuitive for humans and unambiguous for the computer. In this paper formal operating procedure templates are described. These templates can be used for writing a wide range of operating instructions. Examples of the templates in use will be given. On-going research problems are discussed.

INTRODUCTION

Hazard identification is a critical task that needs to be carried out for safe process design and operation. HAZOP studies are widely used for identifying hazard and operability problems. However, HAZOP studies are time consuming, labour intensive and expensive. Automated HAZOP identification systems that emulate the HAZOP technique have been developed to overcome this bottleneck. Much of the research on automated HAZOP identification, based on signed-directed graphs, has concentrated on continuous plants (Larkin, Rushton, Chung, Lees, McCoy & Wakeman, 1997; Venkatasubramanian & Vaidhyanathan, 1994; Wakeman, Chung, Rushton, Lees, Larkin & McCoy, 1997). While a lot of progress has been made in this area (McCoy et al., 1999a and 1999b; Chung et al., 2003), very little work has been done in automated hazard identification of batch plants.

The HAZOP of continuous plants focuses on deviation from steady state operations. This work concentrates on batch processes where the plant moves through a number of

stages during operation, rather than each equipment item remaining in a "steady state", as is normal for continuously operating plants.

To safely produce a product in a batch process, a plant operator follows a sequence 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. 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. By formalising the actions that occur in the batch plant, likely human failure modes which could lead to hazards in the plant can be considered.

In this paper formal operating procedure templates are described. These templates can be used for writing a wide range of operating instructions. Examples of the templates in use will be given. Models supporting the templates are described. An example operating procedure is provided. The automated batch HAZOP application is briefly described. On-going research problems are discussed.

THE TEMPLATE STRUCTURE

Each operating procedure describes an action to perform upon the plant. In order to avoid the difficulties and ambiguities caused by natural language, the operating procedures are composed using a formal template representation. The structure of a template is:

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

An example operating instruction using the template structure is:

Charge reactor1 with caustic from tank2 until reactor1.liquid_amount = 50% vol

where "charge" is the action, "reactor1" is Item1, "caustic" is the material, "from" is the Filler-word, "tank2" is item2 and "liquid_amount = 50% vol" is the condition.

More generally, the "Action" is the operating procedure activity, e.g. "open", "cool", "charge", etc. "Item1" and "Item2" are the equipment instances undergoing the Action, e.g. reactor101, valve103, etc. The Action originates with "Item1" and may act to affect the properties of a second equipment instance, "Item2". In the example above, charging a reactor will affect the contents level of the tank supplying the reactor.

"Material" is a process material, i.e. the contents of an equipment item.

The "Condition" is an ordered triplet consisting of: variable1, logical operator, variable2. Variable1 is the name of the attribute that needs to be monitored. The logical operator comprises of one of the following: "$<$", "$>$", "$\leq$", "$\geq$", "$=$", "$\approx$". Variable2 is the value which terminates the Action.

Table 1. Example instructions in the template format

Action	Item1	*with*	Material	*Filler-word*	Item2	*Until*	Condition
open	valve1						
check	valve2						state = closed
cool	jacket1			content		until	temperature <25°C
wait						until	tank1.level_reading = 0% vol
wash	reactor1	with	water	from	inlet1		
connect	cylinder3			to	line5		
charge	reactor1	with	caustic	from	tank6		
discharge	reactor1			to	tank2	until	liquid_amount = 0% vol

The words "*with*" and "*until*" enable an instruction written using the template structure to be readily understood by a human. "*Filler-word*" is a variable which can take any word as its value and is often a preposition, inserted to make the instruction intuitive to a human.

The "Material" component of the template provides a check on the plant state during operation against the stated intention of the instruction. For example in the instruction "charge reactor1 with caustic from tank2" the Material "caustic" indicates that tank2 is expected to be a source of caustic and should be checked.

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 1. A table format is used to demonstrate the formal template structure. Actual operating procedures use a text format.

OPERATING PROCEDURE STRUCTURE

S88 is a batch control standard (Instruments Society of America, 1995). The S88 standards recommend the information about a batch process is represented hierarchically. The standards describe a model for procedural control. Procedural control directs an ordered sequence of equipment-oriented actions to perform a process-oriented task. The model consists of four levels:

1. The phase is the lowest level element.
2. An operation groups a set of phases.
3. An ordered set of operations comprises a unit procedure. An operation is carried to completion in a single unit.
4. At the highest level is a procedure which consists of an ordered set of unit procedures.

This paper defines the following terms to describe operating procedure structure:

- An action primitive is a basic action which acts directly upon an equipment item. This is equivalent to the S88 phase.
- An operation provides a structure to group together action primitives. This is equivalent to an S88 unit procedure.
- An operating instruction consists of an action primitive not grouped within an operation or an operation.
- An operating procedure consists of a set of operating instructions combined to achieve an overall objective, such as moving material from two different sources into a single tank and then mixing them up.

The following action primitives exist: open, close, operate, run, stop, check and wait. The actions "open", "close", "operate", "run", "stop" and "check" act directly upon an equipment item to change equipment attribute values which model equipment state. They may also act indirectly to change further attribute values. For example, opening a drain valve on a tank may have the indirect effect of causing the attribute tank level to decrease. "Wait" consists of a pause until a condition becomes true and may be more properly viewed as an inaction, e.g. wait until temperature = 25 C.

An operation is a major process activity. An operation details the process activity and provides a structure to group action primitives together. For example, an operation which connects a flow between two equipment instances details the equipment it connects and groups the action primitives which bring about the flow connection. All operations may be broken down into the basic set of action primitives shown above. For example, an operation to charge a reactor will consist of actions to open the valves in the filling lines and operate the charging pumps.

It can be seen that table 1 contains a mixture of primitive actions and high level operations. "Cool", "wash", "connect", "charge" and "discharge" are examples of operations.

Action primitives are grouped together within an operation as shown below. Each line represents one action. Actions are separated by a comma and a carriage return. Brackets ("[","]") indicate the beginning and end of the operation.

```
operation
[
action primitive 1,
...
action primitive n
]
```

The operation connecting a flow between two equipment instances described earlier may be expressed formally as:

```
connect_flow reactor101 to tank101
        [
        open valve101,
        operate pump101
        ]
```

An operating procedure is a sequence of operating instructions. An operating procedure may consist of a mixture of action primitives and operations.

MODELING PLANT STATE

Batch processes require the state of the plant to be expressed. The plant state is changed by the operating procedures. An object-oriented approach is used to describe the plant. Each item in the plant is declared in a plant description file. For each item the following information is given:

- The type of unit it belongs to;
- Which other plant items it is connected to.

For example,

```
instance(tank101 isa tank,
        [
         content info [reactantA],
         outports info [out is [pump101,in]]
        ]).
```

By convention the batch plant is given in its "idle" state, with all valves closed and all pumps off-line. During operation the states of these equipment items are changed by the action of plant operations.

Unit models of equipment item types are defined within a model library as frames. A plant description file contains a number of declarations of "instances" of equipment model frames, which represent the equipment used in a particular plant. Both instances and frames are stored as frame objects. Frame models are stored in the equipment model library. Instances are created when a plant description is read into the automated batch HAZOP system.

Each frame contains a name, a parent name and list of slots. Frames are arranged in an inheritance hierarchy. The parent name is the name of the frame from which a frame is derived, e.g.

```
frame('centrifugal pump' isa 'rotary pump',
```

An example for an instance would be:

```
instance('J1' isa 'centrifugal pump',
```

Inheritance operates from parent frame to child frame within the hierarchy and from a frame to instances of the frame in a plant description. Each frame contains a list of slots which contain equipment state information and may be updated during a batch simulation.

Each slot has a defined type. The type describes allowed values. In addition, the type may also describe possible value units. The allowed values may be discrete, e.g. [discrete, [operating, stopped]] etc., or be chosen from a continuous numerical range, e.g. [cont, [0, ∞]], 'tonnes'.

A slot may take a single item as its value, e.g. mode_of_operation is 'running'. A slot may also take multiple items as its value, e.g. operating_temp info [25, 'C']. In general, the keyword "is" is used to indicate a single item need to be specified. The keyword "info" is used to specify multiple items for a slot. Special slots, "contents slots", group together those slots containing information on equipment item contents, e.g.

```
liquid_Content info
[
liquid_temperature info [25, 'C'],
liquid_amount info [50, '%vol'],(percent of reactor volume)
]
```

The equipment slots defined will vary depending on the individual equipment item class. For example, a valve will have an attribute "state", which may possess the value "open" or "closed". A reactor will not possess this attribute, but will have other attributes.

AN EXAMPLE PROCEDURE

To demonstrate the formal operating procedure template in use, the simple batch processing plant shown in figure 1 is considered. A sequence of operating instructions is given to produce 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. This procedure consists of a mixture of operations and primitives. Equipment models of the items the operating instructions refer to are found within a model library.

To safely produce the product the sequence of operating instructions might be:

1. Charge the reactor with reactant A.
2. Turn on the agitator and the cooling water flow through the cooling jacket.
3. Gradually add a sufficient excess of reactant B to the vessel.
4. Continue mixing reactor contents for a while, to allow reaction to complete.

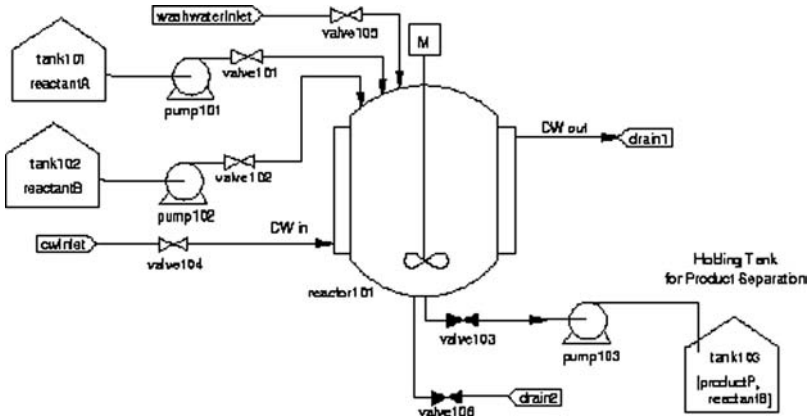

Figure 1. A simple batch processing plant example

Using the formal operating procedure template these instructions are written as:

```
Charge reactor101 with reactantA from tank101 until
liquid_Content.liquid_amount=30 %vol
[
    check valve101 state=closed,
    check valve102 state=closed,
    check valve105 state=closed,
    check valve103 state=closed,
    check valve106 state=closed,
    connect_flow reactor101 to tank101
        [
        open valve101,
        operate pump101
        ],
    wait until reactor101.level_reading=30 %vol,
    disconnect_flow reactor101 from tank101
        [
        stop pump101,
        close valve101
        ]
], (end of charge with reactantA operation)

operate agitator101A,

Cool coolingjacket101A content until
```

```
liquid_Content.liquid_temperature < 25 C
[
    open valve_104, (open inlets of the sub-unit
    coolingjacket101A)
    wait until coolingjacket101A.temp_reading < 25 C,
], (end of cool operation)

Charge reactor101 with reactantB from tank102 until
liquid_Content.liquid_amount = 60 %vol
[
    check valve101 state = closed,
    check valve102 state = closed,
    check valve105 state = closed,
    check valve103 state = closed,
    check valve106 state = closed,
    connect_flow reactor101 to tank102
        [
        open valve102,
        operate pump102
        ],
    wait until reactor101.level_reading = 60 %vol,
    disconnect_flow reactor101 from tank102
        [
        stop pump102,
        close valve102
        ]
], (end of charge with reactantB operation)

wait until agitator101A.time_reading ≈ 20 minutes
```

Attributes are described using an object-oriented notation. For example, the attribute "time_reading" of agitator101A is described as "agitator101A.time_reading". The attribute "liquid_amount" of reactor101 is described as "liquid_Content.liquid_amount" as it is contained within a "liquid_Content" slot. To avoid duplication if the equipment instance is the same as Item1 of the formal template representation the equipment instance name is not shown. The object-oriented notation allows the operating procedures to readily update the plant model state during batch simulation.

It can be seen that operations may be nested. The operation "charge" shown above contains the nested operations "connect_flow" and "disconnect_flow".

THE AUTOMATED BATCH HAZOP APPLICATION

The automated batch HAZOP application takes a plant description and an operating procedure as input and produces a HAZOP report automatically. The heart of the system is the

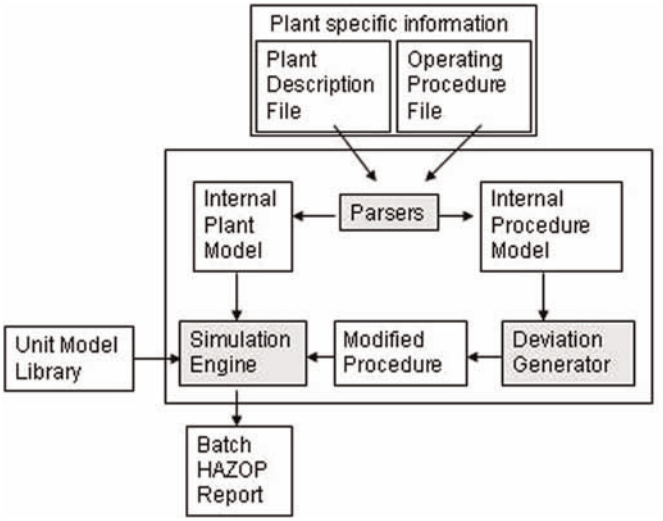

Figure 2. An overview of the automated batch HAZOP application

simulation engine. Given an operating procedure, it applies each of the instructions one at a time and simulates its effect by changing the state of the plant. Therefore, the plant moves from one state to another until all the instructions are completed.

A HAZOP study considers all possible deviations of a plant from its intended operation, by using deviation guidewords (None, More of, Less of, Part of, Other). The causes and consequences of each of these deviations are then examined, and a report of all the important hazards and operability problems is prepared. In the automated batch HAZOP system the deviation guidewords are systematically applied to the operating procedure. The simulation engine 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. Having gone through all the deviations, the simulation engine produces a report file providing warnings against any undesirable situations that may result from the deviations. An overview of the automated batch HAZOP application is given in figure 2.

CONCLUSIONS AND FUTURE WORK

This paper has described a formal template representation to model operating procedures. By formalising the actions that take place in the batch plant, we can analyse the most likely human failure modes which lead to hazards in the plant. The operating instructions have been modelled. A system for modelling plant state has been introduced. This modelling system allows knowledge to be captured and consistently applied.

By combining an operating procedure and a plant description with a simulation engine batch HAZOP can be demonstrated. Deviations from the intended sequence of operations are considered and their resulting hazards documented.

It can be seen in that the example operating procedure contains action duplication, e.g. the valve101 is checked for "state = closed" twice. Some procedure optimisation tests are needed to discover if some of the duplication may be removed.

To assess whether the formal template representation is sufficient a range of operating procedures needs to be analysed. Case studies are necessary to consider whether the automated batch HAZOP tool under development is capable of achieving its purpose. Relevant case studies must be selected. HAZOP results generated by the automated Batch HAZOP tool will be compared with those of traditional manual HAZOP studies.

REFERENCES

Chung, P.W.H., McCoy, S.A. and Zhou, D.F. (2003) Computer-aided Hazard Identification, *Proceedings of Mary Kay O'Connor Process Safety Center Annual Symposium*, October 2003, Mannan, M.S. (ed.), Texas A&M University, Omnipress, pp 409–417.

Instruments Society of America (1995) SA-S88.01.1995 Batch Control Part 1: Models and Terminology. Standards, The International Society for Measurement and Control.

Larkin, F.D., Rushton, A.R., Chung, P.W.H., Lees, F.P., McCoy, S.A., & Wakeman, S.J. (1997) Computer-aided hazard identification: methodology and system architecture, Series No.141, *Hazards XIII Process Safety—The Future*, pp 37–348.

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, Part B, 77, pp 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, Part B, 77, pp 328–334.

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

Wakeman, S.J., Chung, P.W.H., Rushton, A.R., Lees, F.P., Larkin, F.D., & McCoy, S.A. (1997) Computer-aided hazard identification: fault propagation and fault–consequence scenario filtering, IChemE Symposium Series No.141, *Hazards XIII Process Safety—The Future*, pp 305–316.