State-Based Modelling in Hazard Identification *

Stephen A. McCoy†, Dingfeng Zhou and Paul W.H. Chung

Abstract

The signed directed graph (SDG) is the most commonly used type of model for automated hazard identification in chemical plants. Although SDG models are efficient in simulating the plant, they have some weaknesses, which are discussed here in relation to typical process industry examples. Ways to tackle these problems are suggested, and the view is taken that a state-based formalism is needed, to take account of the discrete components in the system, their connection together, and their behaviour over time. A strong representation for operations and actions is also needed, to make the models appropriate for modelling batch processes. A research prototype for HAZOP studies on batch plants (CHECKOP) is also presented, as an illustration of the suggested approach to modelling.

Keywords: Model-Based Reasoning, Qualitative Modelling, Simulation, Batch HAZOP.

1 Introduction

Chemical process plants are very complex systems, and computers have been used for many years now, to aid their detailed design and operation. Most applications have applied computer power to graphics or numerical calculations in this domain, such as mass and energy balances, computer-aided design, numerical simulation, etc.

*Published in Applied Intelligence, Vol 24, 2006, pp261-277

†Modelling and Reasoning Research Group, Department of Computer Science, Loughborough University, Loughborough, Leics., LE11 3TU, UK. e-mail: s.a.mccoy@lboro.ac.uk

In addition to these numerical applications, computers have also been applied to dynamic modelling of plant behaviour using qualitative approaches. Typical applications of techniques such as expert systems and qualitative reasoning have included:

- The analysis (and grouping) of alarms during plant operation, to diagnose causes of plant upsets and help the operators of the plant [1, 2].
- Evaluation of design plans, to find potential sources of operability problems or hazards in the plant when built [3, 4, 5, 6, 7].

The latter application has been worked on extensively by the authors and coworkers [8, 9], in work on developing a software tool for emulating HAZOP studies on chemical plant, and by other research groups in their own work [6, 10].

The approach used is to build models of equipment types (units), which are capable of predicting the dynamic behaviour of the equipment items, in normal operation and under deviations from normal plant operation. Such models have most often been based around a simulation of the important state variables in the equipment item, together with a range of information to capture the possible failures of the equipment, and the susceptibility of the equipment to deviations propagated from elsewhere. The equipment models are connected together to form plant models, within which the effects of deviations can be assessed by propagation of the disturbances modelled.

A very important decision in building such models is the knowledge representation to be used. If a highly complex representation is chosen, it may allow a wide range of phenomena to be modelled, but will incur penalties because of the extra work required to build models and the computational costs of driving simulations. If a simpler representation is used, the models will be easier to understand and to build, and will execute more efficiently, but may not be as expressive as the more complex representation. There is therefore a trade-off between the expressive power of the formalism chosen and the complexity of building models and driving simulations using it.

In much of the work done historically, a simple graphical representation, the signed directed graph (SDG), has been used. This has the advantages of being simple to understand and also very efficient in simulating the propagation of events in the plant – the SDG of the plant is just

searched for paths between remote locations to find all fault propagation scenarios predicting hazards. However, work with SDGs in modelling plants for hazard identification has revealed a number of disadvantages of choosing this formalism:

- The SDG only readily supports two deviations of a process variable ("more" and "less"), whereas there can often be a need for more values.
- The SDG doesn't include any information about the state of the equipment. Thus, it uses the same model of a healthy unit, even when the unit may have malfunctioned, meaning that its behaviour might be quite different from that of the healthy model.
- Sequences of events (and so-called "enabling" faults, which cause a condition in which the plant is susceptible to other failures) are not modelled well by the SDG, which can only handle single linear chains of events.
- Ambiguity – It can sometimes be impossible to tell which of a number of scenarios is actually possible and which impossible, using the SDG model, because the graph does not capture the constraints operating in the real world closely enough.
- Over-reporting hazards – Because the models do not include all that is known about a unit (the SDG will not support all this information), hazards are over-reported. Over-reporting is preferable to missing scenarios, but places a burden on the user of checking all the predictions to see which ones are realistic and which are not.

The weakness of the SDG representation is the cause of most of these problems – important features of the real process are missing from SDG models. The question then becomes: How can the process be better modelled?

In this paper, two cases are examined where the current system is clearly weak. By taking this approach of analysing the weaknesses, some indication of how to improve the knowledge representation can be given. After presenting the case studies, we look at possible ontologies for process systems and discuss current developments of the ideas presented, in an object-oriented approach to modelling batch process systems. The new modelling system is then illustrated by a simple research prototype for batch HAZOP emulation (CHECKOP).

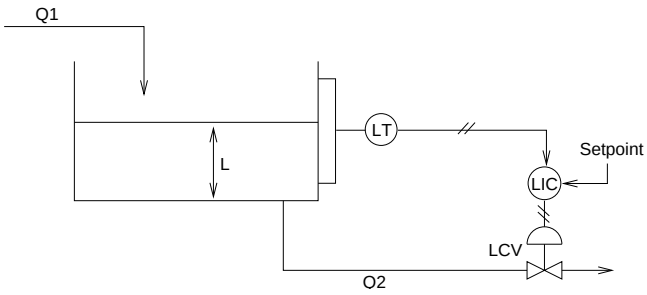


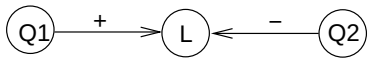
Figure 1: Level Control Example Plant

2 Level Control Example

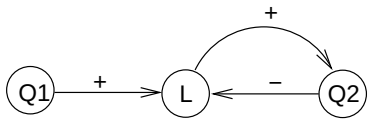
The scenario shown in Figure 1 is a very common one – the level of liquid in the tank is regulated by varying the output flow using feedback control.

Liquid flows into the tank at a certain flowrate Q1 and flows out of it at flowrate Q2. The level of liquid in the tank is L. This level is measured by a level transmitter instrument (LT), and transmitted to a "level indicating controller" (LIC), which compares the level to a setpoint level (and also provides an indication in the control room of what the level is). Depending on whether the level of liquid is higher or lower than the setpoint, the controller then instructs a level control valve (LCV) to open or close, thereby affecting the output flow, Q2. The LIC seeks to maintain the level in the tank at the specified setpoint value, despite changes in the input flow, Q1.

The simplest possible causal model of the tank ignores the presence of the control loop entirely (i.e. it is an "open loop" model). Here it is in SDG notation:



The two relevant hazards, from the point of view of the tank, are overflow and emptying. These are linked directly to the level node in the model, but could be caused by deviations in either flow, Q1 or Q2. Note that an additional arc could be added to this "open loop" plant-only model:



The $L \rightarrow Q2$ arc models the effect of changes in the level of liquid in the tank on output

flow induced by the "static head" pressure in the output pipe.[1] The validity of this link very much depends on the conditions in the outlet line from the tank – if there are control valves, pumps, etc., these may tend to override the "passive" behaviour of the tank + valve + liquid system. Therefore, the "static head" link will be ignored in this example.

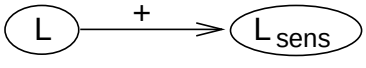
The behaviour of the control loop can be examined by breaking it down into its component parts, each of which corresponds to a common item of process equipment. By building models of each item and connecting them together, the whole plant behaviour can be predicted. The components of the control loop shown in Figure 1 are:

- Level Transmitter (LT)
- Signal Line from LT to LIC
- Level Controller (LIC)
- Signal Line from LIC to LCV
- Level Control Valve (LCV)

This breakdown doesn't show very much detail – other equipment items could have been included if a more refined model was needed, but the high level model serves to cover the overall structure and function of the loop.

2.1 Level Transmitter (LT)

At a very abstract level, the LT is a device which converts the value of a process variable (here, the level of the liquid in the tank) into a signal representing that value (electrically or pneumatically). Therefore, we could represent it as a single connection:



The validity of this link really depends on whether the transmitter is operational – the presence of malfunctions will affect the value transferred to L_{sens}.

[1] The static head is the pressure generated by the weight of a depth of liquid – the pressure increases for increased depth of liquid, which might have an effect on the flow out of the tank.

To identify more accurately what the failure modes could mean for this transmitter, we need to look at its components and consider *their* characteristic failure modes. For the example of a level transmitter, the equipment involved might include a float gauge with a transducer for converting the float position into an electrical signal. Possible failures here include the case where the float is "stuck", for some reason, meaning that variations in level are not measured and transmitted. Alternatively, the transmitter unit may be inoperative due to it being switched off intentionally, or due to power failure. All these would result in a loss of function for the LT unit as a whole.

So, the successful production of a signal by the LT depends on the state of that device being "on_line" (or some other label representing the normal operational state). In other states, a different L_{sens} is propagated to the signal line and thence to the LIC.

If we accept the need for modelling states, some of the relevant state values for the transmitter will include:

on_line Transmitter in normal operational state, receiving power and transmitting a value which accurately represents the level in the equipment item it is attached to.

power_failed Loss of electrical power (for an electrical transmitter), meaning that the output of the transmitter does not represent the level accurately.

air_failed Instrument air failure (for a pneumatic transmitter), having similar effects to power_failed.

off_line Transmitter turned off.

float_stuck The level sensing element (here a float inside the vessel) is stuck, meaning that the transmitter output doesn't accurately reflect changes in the level of liquid in the vessel.

2.2 Transmission Lines

The signal transmission lines from the LT to LIC, and from the LIC to the LCV, are also prone to various modes of failure. These depend on whether the signal (and transmission line) is electrical or pneumatic. Some of the states appropriate for electrical signal transmission lines include:

ok The signal line is in its normal operational state and capable of conveying electrical signals accurately.

power_failed The power supply needed to transmit the signal has failed.

severed The cable has been cut or severed at some point, meaning that no signal transmission is possible.

Some of the states appropriate for pneumatic signal lines include:

ok The line is operating normally, transmitting the required signal pressure accurately.

instrument_air_failed The air supply needed to transmit pneumatic signals has failed.

leaking The line is punctured or severed at some point.

overpressurised The line is being supplied with air at too high a pressure.

obstructed The line is blocked or partially blocked somewhere.

2.3 Level Controller (LIC)

The types of failure appropriate for the level indicating controller (LIC) will depend on the particular controller used in a given case. A number of the states to be expected for a controller device in general include:

on_line Automatically regulating output (i.e. the flow Q2 in the current example) based on controlling the measured variable (i.e. the level L) to a requested setpoint.

manual Controller output is a constant which may be set by the operator.

off_line The controller is switched off and is not providing an output signal.

2.4 Level Control Valve

The control valve is composed of an actuator assembly, attached to a valve body. The valve body has an input and output process connection, carrying the process fluid whose flow is to be regulated. The actuator has two inputs, one from the power supply (electrical power or pressurised air) and one delivering the signal from the controller to the valve (which again may be electrical or pneumatic).

The actuator usually has a default intended behaviour upon power failure, which could be to open the valve, close the valve or keep the valve steady at the last good value. Possible states for the control valve include:

on_line On-line and accepting commands from the signal input.

failed_open, failed_closed or failed_stuck Valve failure modes (each of which may be the default mode or not).

blocked Process side of the valve is obstructed, restricting or preventing the flow of fluid through the valve body.

valve_air_failed The air supply used to move the valve stem has failed (if pneumatically actuated).

valve_power_failed The power supply used to move the valve stem has failed (if electrically actuated).

3 Batch Reactor Example

This example deals with the situation when modelling batch plants, where the plant operation moves through a number of stages, rather than each equipment item remaining in a "steady-state" indefinitely, as is normal for continuously operating plants. The changes in equipment state over time mean that batch processes are more difficult to model accurately than continuous processes.

Figure 2 shows the reactor section of a plant in which product P is produced from two reactants A and B, using the simple chemical reaction $A + B \rightarrow P$. The reaction is exothermic

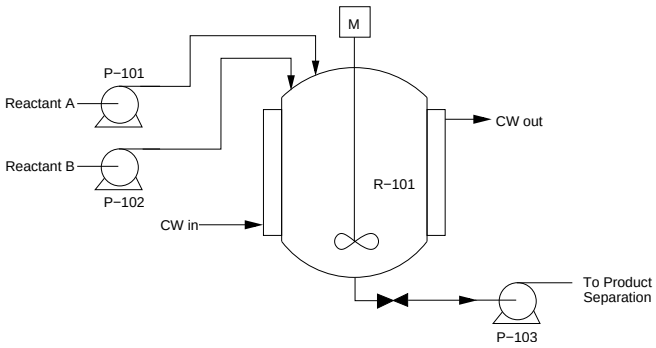


Figure 2: Batch Reactor Example Plant

(it produces heat) and is prone to thermal "runaway", which is one of the main hazards to be avoided in this plant. An excess of reactant B is used, so that reactant A is completely consumed in the reactor.

To safely produce the product, a sequence of operating instructions is followed by the human operators of the plant. A simplified example of such a sequence 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.[2]
4. Continue mixing reactor contents for a while, to allow reaction to complete.
5. Pump the product mixture away for separation.
6. When reactor is empty, switch off agitator, product transfer pump and cooling water.
7. Wash reactor internals thoroughly with water.

In order to reason about what could possibly go wrong with such a plant, it is essential that operating procedures such as this one can be represented. Not only is a knowledge representation needed, for the actions and quantities involved, but also some means of representing the (typically) implicit assumptions associated with each step.

[2]Note: Reactant B must be added at a controlled rate, because of the exothermic nature of the reaction between A and B.

For instance, the above procedure assumes that the reactor is clean and empty at the start of the batch. If, however, an operator had not adequately washed the reactor, or (worse still) had not pumped away the product mixture after the last batch had finished, adding reactant A to the reactor in such a state could have potentially hazardous consequences (such as a premature reaction with reactant B present from the last batch).

Predicting such consequences is possible if some way of representing the sequence of actions in the procedure is available, as well as a way of expressing the assumed state of the plant items before and after each of the steps. A combination of human error analysis and process modelling can be used to detect these hazardous situations and to suggest redesign of the plant to eliminate such dangers.

The points raised in this section point towards enhancing the models used to simulate plant behaviour by adding, not only equipment states, but also the notion of sequence and (by implication) time, into the system. These are typically not tackled in any SDG-based modelling system.

In addition to these, it is clearly important to model the actions of operators, as events which should take place in the operation of a batch plant. The various details of actions, such as quantity of material to transfer, time to wait, etc., have an important but lesser role in identifying hazards, than modelling the presence or absence of the actions in the required sequence.

4 A Domain Theory for Plant Systems

As a means of capturing knowledge representation issues, a domain theory (or "ontology") for chemical plant systems should distinguish between the different sorts of information to be modelled, and should allow an integration of these different sources. This section sets out some of the requirements of such a domain theory, while the following section looks at a research prototype system to implement some of the principles set out here.

4.1 Classification of Objects

The first thing to do is to draw up a taxonomy of the objects to be modelled, covering all the types of information needed to predict plant behaviour. We can start to draw some of the higher

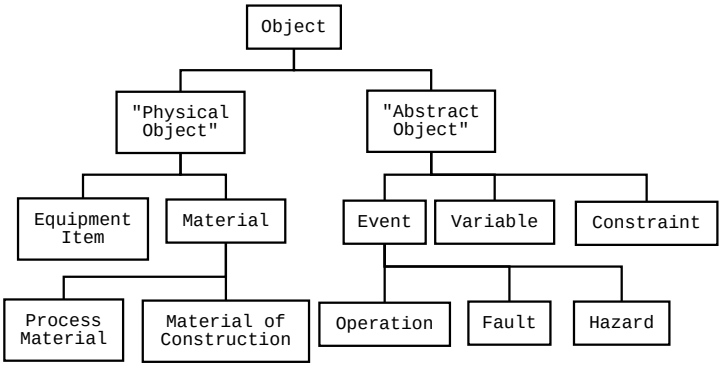


Figure 3: Top Levels of a Taxonomic Tree for Objects of Interest in Modelling Chemical Plants

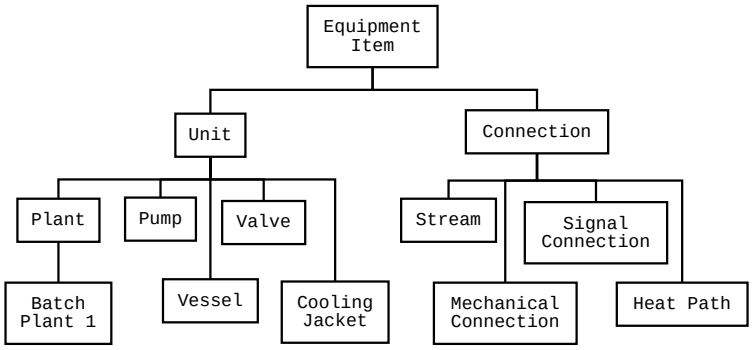


Figure 4: Top Levels of an Equipment Hierarchy

level types in the hierarchy as a tree (see Figure 3). The placement of objects within this tree is tentative at this stage and potentially open to modification; nevertheless, it is a good first framework for the relevant objects in this domain.

The first level distinguishes between “physical objects”, corresponding to tangible entities in the world, and “abstract objects”, which are the intangible elements which are essential to know how the plant is operated and how its behaviour is modelled in terms of variables whose values change over time.

An equipment item is a countable physical object which forms part of the plant. A partial hierarchy of equipment items, showing the distinction between units and connections between units, is shown as Figure 4. The thinking behind this is that the Units in a system can be connected to one another in different ways, which can be modelled as Connection objects of various types.

Materials are “liquid-like” in the knowledge representation sense of being mass-noun enti-

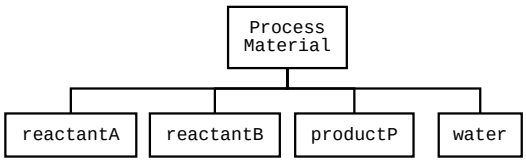


Figure 5: Process Materials Hierarchy for Batch Plant Example

ties with particular properties, but are not discretely countable. A distinction is made between the materials processed by the plant (solid, liquid and gaseous chemicals) and the materials from which the plant is constructed (steel, glass, plastic, etc.). Some possible process materials, for the earlier batch plant example, are shown in Figure 5.

The abstract part of the hierarchy in Figure 3 governs all the non-physical entities which determine the plant behaviour. Events are things which may cause or be associated with changes in the physical state of equipment or materials in the plant – three sub-types are given so far. Operations define the operating instructions of the plant, and can be highly structured (as will be seen below). Faults are spontaneous failures and Hazards are potentially dangerous events which occur due to Faults or other plant disturbances. A Variable is a plant item property such as a flow, temperature, pressure, etc. Variables will typically be modelled using a Quantity whose value type is qualitative, rather than numerical. The Constraint class allows a range of functional relationships to be modelled, so that causal (or non-causal) constraints between variables or events can be captured.

4.2 Structure in the Plant

The physical plant itself is modelled as an equipment item – albeit one with a complex structure and many components. Each component of the plant is an instance of an equipment model. A similar breakdown is possible at any level for any equipment model – a Unit or a Connection can be composed of a number of Units, connected together by Connections. Using a standard breakdown structure like this allows models to be built of equipment sub-components, such as agitators, motors, cooling jackets, etc. The sub-components can then be reused in many different models without change.

This compositional modelling approach can also be applied to other parts of the plant modelling problem – to model the interconnection of instrument systems (using signal connections)

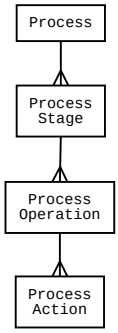


Figure 6: Process Breakdown according to the ISA/S88 Standard

or the relation between operating instructions in a plan (using precedence constraints).

4.3 Operating Instructions for Batch Plants

The same type of structure should be used for modelling the operating instructions of the plant, as for modelling the sequence of events in a scenario (hazardous or otherwise). What is needed in both cases is a way of organising the events that occur (or should occur, in the case of operating instructions) so that they can be arranged in either unordered groups or ordered sequences. The order of events may be governed by a set of precedence constraints in the plan, in the same way that STRIPS defines operators for planning. This would allow the more flexible discovery or variation of constraints than would the hard-coding of sequence or parallel into the plan.

In modelling the operating procedures of the batch plant, we follow a similar framework to that given in the ANSI/ISA S88 standard for batch process plant [11]. In the breakdown hierarchy for processes defined by S88 (see Figure 6), a Process is composed of a number of Process Stages, each of which is in turn composed of a number of Process Operations. The Process Operations are achieved by a sequence of Process Actions at the lowest level. Process actions are primitive actions that the operator may perform, such as “open valve”, “switch on pump”, etc.

Examples of process operations which may be present in a batch process include “charge the reactor R-101 with 5000 litres of solution A”, or “heat the reactor contents to 85 °C and maintain temperature for 3 hours”. The operations themselves are composed of a number of process actions, not necessarily in simple sequence. The process operations therefore correspond most closely to the level of operating instructions an operator will usually follow, whereas the process

actions correspond to the detailed activities that he/she will perform in order to achieve the goals of the stated operation.

The actions comprising a particular process operation (or the process operations comprising a particular process stage) can be arranged in a partial order using a Petri Net, or some other partial ordering scheme. It is then possible to reason about how different errors in operation (omitting actions/operations, doing them too late, too early, etc.) may affect the result of a given operating plan.

At least initially, process stages and higher levels of organisation are not so important to the representation of detail, and hence to analysing common errors, than the process operations and actions just mentioned. These higher levels will therefore be used simply to group operations together at this time.

4.4 Frames for Process Operations

The operations (whether specified at the "Process Operation" level or the "Process Action" level) can be defined in terms of a frame-based representation. Each type of operation is represented by a frame, with a number of slots. Each slot holds information giving more detail about the operation it belongs to. The set of slots relevant to each operation depends on what type that operation is. A number of action types may be defined, based on generalising the range of detailed actions seen in a real plant.

As an example, some of the slots which might be used for a "Transfer" operation ("Transfer 5000 litres of reactant B to reactor R-101") are:

- Preconditions – The conditions to be satisfied before the action starts.
- Postconditions – The conditions which are satisfied when the action finishes.
- Agent – The Operator, or Team of Operators who perform the action.
- Type – The base name of the action type (e.g. "transfer", or "charge").
- Object – The thing, or material, which the operation operates on. In this case, we have a quantity of process material to transfer:

  - Material – Reactant B.
  - Quantity – 5000 litres.

- Source – The starting place for the Object (storage tank?).
- Destination – The finishing place for the Object (reactor R-101).
- Plan = (Subaction*, Constraint*) – A number of subactions which achieve this action, together with any constraints between them, governing what order the subactions may be performed in.
- Time – The time interval/point over which the event takes place.

The preconditions and postconditions elements of the frame are important in establishing when an action or operation can be initiated and when it is over. These have impacts on whether the order of actions is fixed within an operation. Many of the other slots can be missing from the frame without the sense of the operation being lost.

4.5 Interface between Operations and the Plant Model

The model of the batch plant includes a number of equipment models, for each of the tangible equipment items on plant, and also a number of operating instructions, for performing specified Process Stages. When the operating instructions are executed, they cause changes to occur in the plant, which should be reflected in changes in the model states. We therefore need a formal way of mapping the actions performed (at the most basic level) to the effects they have on the plant equipment.

4.6 Petri Nets for representing Process Operation and State Transitions

Petri Nets (PNs) [12] are a mature technology for representing sequences of events in the context of simulating discrete event systems. It is worthwhile to consider whether Petri Nets are a suitable representation for the operating instructions in batch processes. In order to use them, we need to introduce a correspondence between the elements of a PN and the objects in the plant system:

Transitions correspond to actions which cause a change in the state of the plant.

Places correspond to states of the plant and its equipment.

Arcs connect places to transitions and vice-versa.

Tokens correspond to the associated state condition holding in the plant model.

In addition to the above elements (which are standard parts of the classical PN model), to represent operating procedures we must label one place in the net as the "start" (s) and one as the "finish" (f). This is necessary, to allow us to know when a given procedure has been completed. Given the batch plant example introduced earlier, we could represent its operation as the Petri Net shown in Figure 7, which is shown in the state where we are waiting for the reaction to complete. When the token (marked as an "x") reaches the place marked (f), the procedure is complete.

It is also natural to define some way of decomposing operations into smaller actions or steps so that commonly repeated operations can be modelled using a template or "model" for the operation. Therefore, using this Petri Net notation, a decomposition of high level actions into sub-actions is achieved by defining "sub-nets", each of which corresponds to a single action type and gives the detailed sequence of actions needed to complete that action.

As an example, consider a refinement of the "Charge B to excess" process operation from Figure 7, as shown in Figure 8. When the transition "Charge B to excess" is enabled in the first Petri Net, a sub-net is used, to determine the detailed steps required to achieve the stated operation. Using this technique, a Petri Net can be seen as a hierarchical structure, where some operations are achieved by sub-nets, which hide detail from the top level view of the PN transitions.

Petri Nets model the inherent sequence/parallelism in a fully formed plan very well. They are therefore quite an attractive model for visualising the operations being modelled. However, PNs do not allow a flexible enough representation of the "alternative plans" which arise from deviations from the intended operations of a plant. Thus, if we wish to consider what would happen if the order of two tasks were swapped, we find that it is very difficult to modify the PN to take account of this.

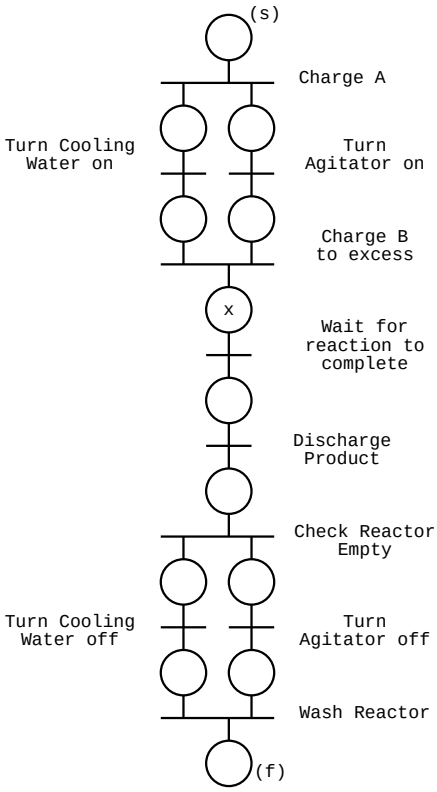


Figure 7: Example of a Batch Plant Petri Net

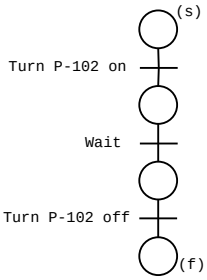


Figure 8: Detail of “Charge B to excess” Operation from Figure 7

Additionally, an experiment into the use of PNs to model a simple batch process (making a cup of tea) demonstrated that this type of representation is not best suited to presenting operations in a simple, easy to understand way. The PN constructed was complex and difficult to interpret, even for such a simple "plant", due to the inherent complexity of modelling the process at such a detailed level.

The weakness of Petri Nets for this application is that they integrate the places and transitions (i.e. the states of the plant and the actions that are performed on the plant) very closely. It is therefore very difficult to reason about the effect of varying elements from either domain:

- Consider what happens when some part of the plant is modified, or new equipment is added – the PN and the associated operating sequence are very likely to be wrong, and not in a simple way.
- If the order of operations in the procedure is changed or new ones are added (whether intentionally or through operator error), the effects on the PN are likely to be complex to predict, effectively meaning that a new PN needs to be constructed from scratch.

Given these observations, it is best to use a representation which de-couples the operations from the equipment in a plant, so that variations in either domain can be considered more cleanly.

Therefore, there are a number of problems with using PNs for this type of plant system, even without considering the issue of how to interface the PN representation to a potentially continuous plant model and its continuous feedback control loops. These are regrettable difficulties, as the visual appeal of the PN for simpler systems is undeniable.

For the purposes of our hazard identification system, we aim to model all the possible operating sequences of the plant by using local constraints between actions – without having to commit to a particular complete plan. This method can be used to determine if a given whole plan satisfies the constraints, or (ultimately) to generate an optimal plan sequence for the operation from a number of actions specified.

4.7 Other Qualitative Approaches

It is worth considering why the current work has not followed the methods established in the wider Qualitative Reasoning (QR) community, as typified by the "confluences" approach of De Kleer and Brown [13], the qualitative process theory of Forbus [14], or the qualitative simulation (QSIM) approach of Kuipers [15].

One of the major problems of QR has been the control of ambiguity in the predictions produced by its models. Many simple arithmetic operations such as addition are entirely ambiguous when transposed into the qualitative domain. This type of ambiguity results in a severely branching tree of predicted behaviours, and seriously limits the size of models whose behaviour can be simulated – and presented to a user in an intelligible way.

For this reason, we chose to develop a more strongly object-oriented, state-based, component-centred approach to system modelling, in which numerical quantities could be used as well as supporting qualitative reasoning in the shape of local constraints between objects considered to be physically connected. Ambiguity of behaviour will doubtless remain within this type of model, but we hope that it will be better controlled.

5 Batch Process Modelling

A number of the ideas outlined in previous sections have been implemented in a computer program ("CHECKOP") to tackle the issue of modelling batch plants and their operation for hazard identification. The simple batch reactor plant seen earlier has been extended for demonstration purposes, adding more detail to the design of both the plant equipment and the operating instructions.

In modelling the batch reactor we want to simulate the plant as it moves through a number of states during operation. It is important to capture the intended operation sequence of the plant and also to model how deviations from this can arise. This capability will allow us to perform automated batch HAZOP analyses and also consider human error in the operation of the plant.

Deviation	Causes	Consequences	Suggested Actions
More flow in feed to settling tank	LCV fails open or LCV bypass open in error	Settling tank overfills	Install high level alarm on LIC and check sizing of relief opposite liquid overfilling.
		Incomplete separation of water phase in tank, leading to problems on reaction section.	Extend J2 pump suction line to 12 inches above tank base.
More pressure	Isolation valve closed in error or LCV closes with J1 pump running	Transfer line subjected to full pump delivery or surge pressure	Install kickback on J1 pumps.

Figure 9: Example of Typical HAZOP Output (from [18])

5.1 Batch HAZOP

The commonly used HAZOP technique for hazard identification [16, 17] is a method study used by a small team of engineers, typically after the majority of detailed process engineering design has been decided for a new plant. The study considers all possible deviations of a plant from its intended operation, by using deviation guide words (More, Less, No, Other, etc.) applied to each of the important process variables in the plant in turn. 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 (a small example is given as Figure 9). For a continuous plant, the expected operation of the plant is a single steady state, and deviations from that are disturbances in the pressures, temperatures, flows, etc. in the plant. Operations such as maintenance, startup and shutdown are also usually considered as separate operating regimes.

Batch HAZOP [16, 19, 20] extends this method to consider deviations from the intended state of the batch plant during each of the stages of its operations. The technique also looks at variations in the operating instructions of the plant, to consider what would happen if operations were missed out, performed in the wrong order, for too long a time, etc. A set of extra deviation guidewords is used to relate to the operations part of the batch HAZOP: No action, Late/Early action, Other action, etc.

Both HAZOP and Batch HAZOP have been used for many years as conventional team-

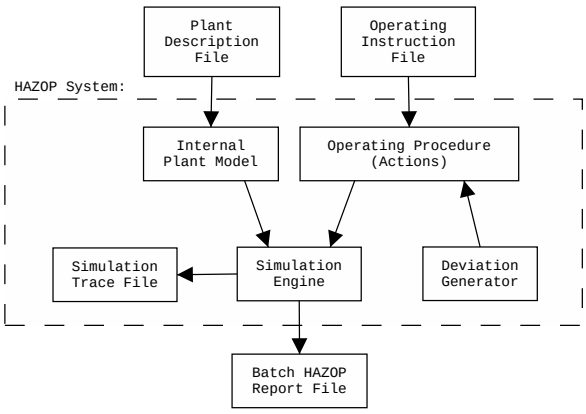


Figure 10: Outline CHECKOP Data Flow

based study techniques, performed by small groups of experienced engineers involved in the design of process plant. In view of the significant time spent in such studies, there is a strong incentive to automate them, or to facilitate their efficient execution.

The guide words used in conventional HAZOP and batch HAZOP meetings are treated most often as prompts to the group, to “spark off” thinking about deviations in the plant. In contrast, any computer system for emulating HAZOP by simulation of the process has to be precise in the meanings of the deviations it considers, so that these can be directly related to parameters in the plant model. For this and other reasons, the style of report produced by a HAZOP team secretary will usually be different to that produced by a HAZOP emulation program such as the HAZID software developed at Loughborough.

5.2 CHECKOP

The research prototype for batch process modelling and batch HAZOP is known as “CHECKOP”. As developed so far, CHECKOP reads a set of operating instructions from one file and a plant description from another file. It then uses the plant description to build an internal representation of the plant equipment (including the states of the equipment items). It uses the operating instructions file to construct a list of Action objects, each of which corresponds to a single operation, and a single line of the file. The data flow within CHECKOP is shown schematically in Figure 10.

The plant description file is written in the same format that has been used already in the

HAZID tool for automated HAZOP analysis, developed for analysis of continuous process plants [8, 9]. The plant description includes declarations of the equipment items in the plant, their types and the fluid flow connections between them. The difference between this plant description and the type given in HAZID for continuous plants, is that the batch plant is (by convention) given in its "idle" state, with all valves closed and pumps off-line – during operation, the states of these equipment items are changed, by the action of the plant operations.

When they are executed, the Actions in the operating procedure change the states of the Equipment objects in the plant. Warnings are issued if the required preconditions of Actions are not satisfied. Safety-related conditions of the plant (e.g. having the agitator still running when the reactor has been emptied) are also monitored, and warnings are issued where necessary.

The internal representation of operating instructions as a sequence of Actions allows alternative orderings of Actions to be considered by the program. This allows the procedure to be modified, so that the results on the plant model can be determined. An initial application of this is to consider the automatic generation of batch HAZOP guidewords for modelling possible abnormal scenarios.

Batch HAZOP emulation is achieved by driving the simulator engine using deviations of the given operating instructions, to examine the consequences of "No action", "Early/Late action" and "Early/Late termination" deviations. These are not a complete set of guidewords, but do cover a number of types of deviation, relating to sequence of action, which are most easily modelled by the current system. It is necessary to predict the sequence of plant states resulting from each deviated operating procedure, and to compare that to the states of the intended procedure. By making a note of the differences, the important effects of the deviation are captured for hazard reporting. Two outputs are generated by CHECKOP – the simulation trace gives a step by step description of the state of the plant as it is simulated, while the batch HAZOP report gives a tabular summary of the consequences found.

5.3 Application of CHECKOP

The batch reactor plant presented in Section 3 has been extended, by adding further detail to the equipment set and to the operating instructions given previously. The extended plant is shown schematically in Figure 11. The associated operating instructions file, for producing one batch

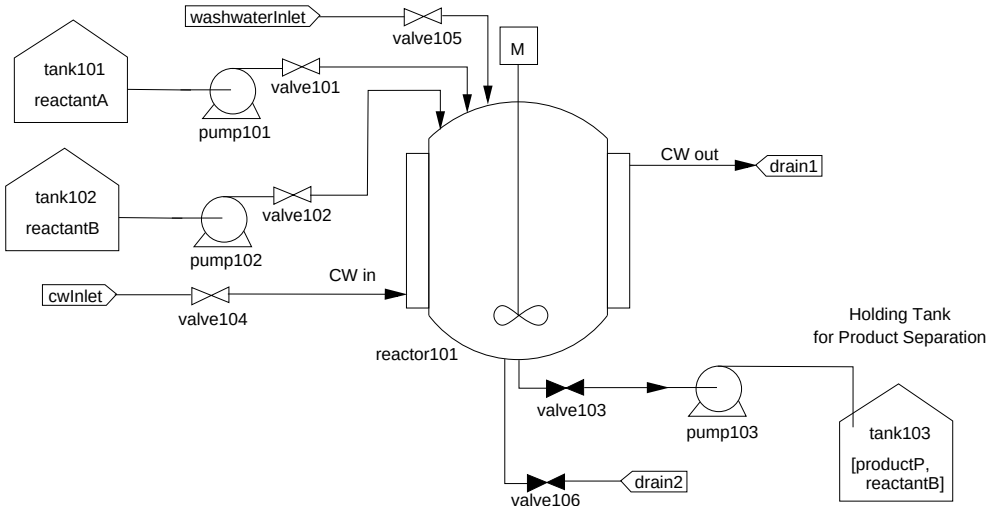


Figure 11: Extended Model of Batch Reactor Plant

of product P, is as shown in Figure 12. The plant description file is written in the format shown in Figure 13, which is similar to the format used in the HAZID tool for automated HAZOP analysis.

It should be noted that within the definition of the operating instructions, each of the operations mentioned in Section 3 is given, with associated primitive level actions. Each numbered line corresponds to one action in the procedure. Each line specifies the name of the object which is the main subject of the action first, followed by the name of the action and a list of arguments providing extra information.

Some of the arguments given may seem to be redundant, but they do play a part as a way of checking the plant state during operation against the stated intention of the action in the instructions. For example, in the action "reactor101 fill_from tank101 with reactantA until volume 30 percent", the "with reactantA" part may be considered as a check that tank101 *is* a source of reactantA – just in case the tank contents have been contaminated, or the action wrongly executed, for instance.

Note also that some of the actions shown in Figure 12 involve more operator involvement, and may therefore seem not to be "primitive", compared to others (e.g. compare "pump101 start" to "reactor101 wash with water"). This is inevitable, but does not prevent simulating the state of the plant during operation, so long as the preconditions and postconditions of the action are well-defined.

```
charge reactor101 with reactantA: {
   (1) valve101 open
   (2) pump101 start
   (3) reactor101 fill_from tank101 with reactantA until volume 30 percent
   (4) pump101 stop
   (5) valve101 close
}
start_up reactor101 cooling and agitator: {
   (6) agitator turn_on
   (7) valve104 open
   (8) jacket1 cool_content until temperature 25 degree
}
charge reactor101 from tank102 with excess of reactantB: {
   (9) valve102 open
   (10) pump102 start
   (11) reactor101 fill_from tank102 with reactantB until volume 60 percent
   (12) pump102 stop
   (13) valve102 close
}
wait until reaction complete: {
   (14) agitator stir_content of reactor101 until elapsed_time 20 minutes
}
discharge product mixture for separation: {
   (15) valve103 open
   (16) pump103 start
   (17) reactor101 empty_to tank103 with [reactantB, productP] until volume 0 percent
   (18) pump103 stop
   (19) valve103 close
}
shut_down reactor101 cooling and agitator: {
   (20) agitator turn_off
   (21) valve104 close
}
wash reactor101 thoroughly with water: {
   (22) valve106 open
   (23) valve105 open
   (24) reactor101 wash with water
   (25) valve105 close
   (26) valve106 close
}
```

Figure 12: Operating Procedure for Batch Plant

```
% Plant description for the extended model of the simple batch reactor plant.

instance(tank101 isa tank, [content info [reactantA], outports info [out is [pump101,in]]]).
instance(pump101 isa pump, [status is offline, outports info [out is [valve101,in]]]).
instance(valve101 isa valve, [status is closed, outports info [out is [reactor101, in2]]]).

instance(tank102 isa tank, [content info [reactantB], outports info [out is [pump102,in]]]).
instance(pump102 isa pump, [status is offline, outports info [out is [valve102,in]]]).
instance(valve102 isa valve, [status is closed, outports info [out is [reactor101, in3]]]).

instance(washWaterInlet isa inlet, [content info [water], outports info [out is [valve105,in]]]).
instance(valve105 isa valve, [status is closed, outports info [out is [reactor101,in1]]]).

instance(reactor101 isa stirred_tank_reactor, [
  outports info [out1 is [valve103,in], out2 is [valve106,in]],
  heatSink info [hout is [jacket101,hin]],
  reaction info [reaction_ab_p]
]).

instance(valve103 isa valve, [status is closed, outports info [out is [pump103,in]]]).
instance(pump103 isa pump, [status is offline, outports info [out is [tank103,in]]]).
instance(tank103 isa tank, [content info [reactantB,productP]]).

instance(valve106 isa valve, [status is closed, outports info [out is [drain2,in]]]).
instance(drain2 isa drain, []).

instance(cwInlet isa inlet, [content info [water], outports info [out is [valve104,in]]]).
instance(valve104 isa valve, [status is closed, outports info [out is [jacket101,in]]]).
instance(jacket101 isa cooling_jacket, [outports info [out is [drain1,in]]]).
instance(drain1 isa drain, []).

instance(reaction_ab_p, [thermo_type is exothermic,
  reactants info [[reactantA,1],[reactantB,1]],
  products info [[productP,1]]]).
```

Figure 13: Plant Description for Batch Plant

The plant description given in Figure 13 includes information about the flow connections and also about the thermal (heat flow) connections between equipment items in the plant. This allows CHECKOP to infer suitable qualitative models for fluid flow and heat transfer in the plant. It should also be noted that only very basic information about the reaction is adequate to give a model of what happens in the reactor when reactantA and reactantB mix.

A sample of the type of simulation trace output produced by CHECKOP, when it is used to analyse the plant for batch HAZOP purposes, is shown in Figure 14. A single guideword ("NO") is being considered in relation to a single operation in the operating instruction (in this case action (7), which is "valve104 open"); the state of the plant is simulated as the modified instructions are executed in sequence. It can be seen that CHECKOP detects one of the consequences of not opening the cooling water supply valve – that the reactor contents cannot be cooled in step (8). Further warnings are detected later in the procedure when the reactor overheats due to lack of cooling – these further warnings are not shown here due to lack of space.

The engine can be used to do step by step simulation of the plant state. It can also be driven by a batch HAZOP procedure, to examine all feasible deviations in the given operating instructions.

A short excerpt of the type of batch HAZOP output produced by CHECKOP is shown in Figure 15. It can be seen that the format shown here is similar to that produced in a traditional HAZOP study, except for the absence of a column for "Causes" (deviations as generated are deemed to be sufficient cause for the scenarios reported) and for "Suggested Actions" (CHECKOP does not attempt to advise on how to solve the problems it detects).

Note that the instructions in the operating procedure are referred to by line number in the Operation and Consequences columns. If no such number is given in the consequences column, then the operation referred to by that consequence is the deviated one. This is to reduce the scope for ambiguity in the report, and allows discovered consequences to be related to the point in the procedure when the problem arises. For Early/Late action, a number is given in parentheses after the entry in the Operation column – this gives the number of sequence steps early or late that the action takes place, for the deviation considered. For Early/Late termination deviations, the conditions specified in the "until" portion of the action are varied – details of

```
Considering guideword "NO" applied to single Action: (7) valve104 open
Processing Procedure...
(1) valve101 open ... OK.
valve101 state:
  aperture is: open; upstream is: [pump101,out]; downstream is: [reactor101,in2]
(2) pump101 start ... OK.
pump101 state:
  state is: on; upstream is: [tank101,out]; downstream is: [valve101,in]
(3) reactor101 fill_from tank101 with reactantA until volume 30 percent ... OK.
reactor101 state:
  level is: 30 percent volume; content is: reactantA; max. level is: 100; min. level is: 0;
  content temperature is: 20; upstream is: [valve101,out],[valve102,out],[valve105,out];
  downstream is: [valve103,in],[valve106,in]
tank101 state:
  level is: 40 percent volume; content is: reactantA; max. level is: 100; min. level is: 0;
  content temperature is: 20; upstream is: ; downstream is: [pump101,in]
(4) pump101 stop ... OK.
pump101 state:
  state is: off; upstream is: [tank101,out]; downstream is: [valve101,in];
(5) valve101 close ... OK.
valve101 state:
  aperture is: closed; upstream is: [pump101,out]; downstream is: [reactor101,in2]
(6) agitator turn_on ... OK.
agitator state:
  state is: on; duty is: [reactor101,liquid]
(7) NO (valve104 open) ... OK.
valve104 state:
  aperture is: closed; upstream is: [cwInlet,out]; downstream is: [jacket1,in]
(8) jacket1 cool_content until temperature 25 degree ...
*** Warning ***: no coolant flow in jacket1 - cannot cool process.
OK.
jacket1 state:
  state is: normal; content is: nil; duty is: [reactor101,liquid]
(9) valve102 open ... OK.
```

Figure 14: Sample Simulation Trace Output from CHECKOP

how much are given in brackets after the entry in the “Operation” column.

5.4 Comments on Batch Process Modelling Example

The operating procedure has been modelled as a sequence of actions, each of which has a number of effects on the plant’s state. Each operation/action also assumes some things about the prior state of the plant/equipment it relates to.

What happens if the assumptions are not true when the execution of the instructions is simulated? For example, the instruction “reactor101 fill_from tank101 with reactantA until volume 30 percent” assumes that reactant A is present in the tank. If this is not the case, then a message is printed, which is adequate to identify problems with the instructions as given in the input file. Having given a warning, the program then continues to predict the effects of executing the procedure as given. Two options are possible here:

- Firstly, the Action may not be allowed to complete and the procedure would block. This would happen if the inventory of A were exhausted and we were to insist that the “30 percent” condition were satisfied.
- Alternatively, the Action may be completed in its erroneous form, and the results simulated. For example, if tank101 instead contains reactantB, the program would simulate what happens if the reactor were filled to 30 percent with this material instead.

What to do here depends on the application domain of the simulation. For hazard identification, the policy which most accurately predicts real behaviour in a “credible worst case” scenario is what should be adopted. Without any heuristics to judge whether the “blocking conditions” are reliably enforced in the system, the best thing to do is to assume that the Action proceeds in error (i.e. the second option above). This is what is done for the batch HAZOP procedure in CHECKOP, as far as possible. It should be noted that this type of policy may lead to unwanted over-reporting of infeasible problems, but should not miss any problems which can be captured by the simulation.

Further work is required on the deviation generator part of CHECKOP, in order to allow it to generate the most credible deviations of a given operating instruction. “No action” is

Keyword	Operation	Consequences
No action	(1) valve101 open	(2) pump101 deadheading, equipment damage. (2) pump101 overheating. (3) reactor101 cannot be filled from tank101 because there is no flow path. (6) agitator running while vessel empty.
Late action	(1) valve101 open (+3)	as No action (1) valve101 open.
No action	(2) pump101 start	(3) reactor101 cannot be filled from tank101 because there is no driving force for flow.
Early action	(2) pump101 start (–1)	pump101 deadheading, equipment damage. (1) flow surge into reactor101 – possible overfilling or damage.
Late action	(2) pump101 start (+3)	(3) reactor101 cannot be filled from tank101 because there is no driving force for flow. pump101 deadheading, equipment damage.
No action	(3) reactor101 fill from tank101 with reactantA until volume 30 percent	(6) agitator running while vessel empty. (17) cannot empty [reactantB,productP] from reactor101 to tank103 because content of reactor is [reactantB].
Early action	(3) reactor101 fill from tank101 with reactantA until volume 30 percent (–2)	reactor101 cannot be filled from tank101 because there is no flow path.
Late action	(3) reactor101 fill from tank101 with reactantA until volume 30 percent (+3)	reactor101 cannot be filled from tank101 because there is no flow path.
Early termination	(3) reactor101 fill from tank101 with reactantA until volume 30 percent (to only 10 percent)	no consequence.
Late termination	(3) reactor101 fill from tank101 with reactantA until volume 30 percent (to 50 percent)	(17) cannot empty [reactantB,productP] from reactor101 to tank103 because content of reactor101 is [reactantA,productP]. (17) contamination in tank103. (17) exothermic reaction in tank103: reactantA + reactantB → productP.

Figure 15: Sample Batch HAZOP Output from CHECKOP

straightforward – there are only a limited number of ways of applying this guideword. "Early action" and "Late action" present more difficult decisions – how far is it permissible to move the action? For the illustration above, we have used a maximum of 3 sequence steps (early or late) in the procedure, but without some sensible limit on this and other deviation guidewords, exhaustive examination of all deviations could bury the user in unwanted detail.

A useful development of CHECKOP, which we can envisage for the future, is to combine it with the HAZID engine used for continuous process plant HAZOP emulation. Such a combination would be able to take advantage of the efficient simulation performance of HAZID, to examine batch process hazards and to check out startup and shutdown procedures of continuous plants.

5.5 Flow Modelling

CHECKOP solves one of the typical problems encountered in this and another related area – Operating Procedure Synthesis (OPS). The problem is that of how to determine the effect on flow of actions involving opening or closing a valve [21]. For example, if two valves are present in sequence in the same line, then opening one of them will not produce a flow through it if the other valve is closed. Similarly, if the two valves are in parallel sections, then closing one will not necessarily prevent fluid from flowing.

This flow modelling problem cannot be solved ahead of time and must be found during a run-time simulation of the system. This means that simple STRIPS-style operators (with associated lists of preconditions and effects) are inadequate for modelling the effects of actions in this domain, if the effects to be modelled include the facts of flow existing at different places in the plant.

In its use of state-based simulation and run-time search for flow path connections, CHECKOP uses the "action synergy" approach to flow modelling and generating the effects of valve operations, as also explored in the work of Soutter and others [21, 22]. In OPS systems, the action synergy approach is used to find safe sequences of valve operations to achieve planning goals, given that the operations will have overlapping and perhaps conflicting effects on the flows in the plant. In CHECKOP, the aim is to simulate accurately when flows are possible and when they are not possible.

6 Discussion

The example scenarios outlined in Sections 2 and 3 above have shown some problem areas for the SDG-based modelling approach. Such an analysis is useful in pointing the way to further improvements in the representation used for modelling the plant.

One thing that is clearly right about the SDG-based approach used to date is the decomposition of the plant into unit models corresponding to equipment items on the plant. This feature of the formalism must remain – structural decomposition in this way makes sense from the point of view of handling complexity and is also the way engineers tend to think about the plant and its design. Keeping this approach also means that the topology of the model will superficially resemble that of the plant itself, which may aid understanding.

However, as mentioned above, the decomposition process must be taken further, to allow components and sub-components to be defined within the equipment models. In previous work [8, 9], the unit models contained a large number of the "atomic" components in the system (SDG arcs), which added to their complexity. The task of modelling the unit behaviour can be simplified by defining components which are smaller than equipment units, but larger than the "basic unit" of the model system (whatever that might be in the chosen formalism). From the level control loop example, a level transmitter was composed of a float gauge component connected to a transducer component.

Most important, for the modelling of malfunctioning units, is the concept of defining the state of the units (not possible in the SDG approach). Equipment items most often behave according to the default "healthy" model, but have different behavioural modes, corresponding to "off-normal" states (e.g. if the unit is switched off, or not connected to needed information or power sources). State-based models can support reasoning about "common mode" failures, which arise due to power or other utility failures across the whole plant. Common mode failures are difficult to model using the SDG because of the inability to envisage the parallel development of scenarios in different parts of the plant at the same time.

For batch processes, the SDG is particularly poor in capturing the actions needed to operate the plant and the changes in state which occur during normal operation. To support this, the representation must include the idea of a sequence of operating instructions, which are operator

actions which are performed in order. Each action is associated with a set of assumptions about the state of the plant and an expectation of the results of the action. Incorporating these fields will allow the intended function of the plant to be compared with the simulation of what will actually happen in "normal operation". Furthermore, 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, to complement an analysis of the process-initiated ones.

Despite the points made above about state dependent models, the most usual mode of operation for units in the plant will correspond to the same type of model previously used when modelling using the SDG. Therefore, the models will behave in a very similar way, most of the time. As was seen in the level control example, the connectivity of the loop is only broken when one of the failure modes takes over in an equipment item.

These improvements are the basis of a representation system currently under development, and implemented in the "CHECKOP" research prototype, to allow more accurate process system modelling. We have presented the new system in the preceding section – it is based around a qualitative, object-oriented model of the process equipment and the instructions used to operate it. We have modelled the operating instructions of a number of plants, and one example is shown here, for a simple batch reactor. The example demonstrates how the state of the plant can be predicted through time, and how safety-related problems for each equipment item are monitored and warnings are given when needed. Operating problems, leading to the given instructions being impossible to complete, are treated in a similar way. Using this simulation engine, batch HAZOP can be demonstrated, by considering deviations from the intended sequence of operations, and their resulting hazards.

CHECKOP is significant in view of the fact that it incorporates a state-based model of the process equipment, a separate model of the operating procedure to be followed, and a simulation method used to predict the evolution of the system state through time. We believe this to be the first time such an integration has been achieved in the process safety domain. The current system is still limited, but we see a lot of promise in further developing the ideas in it – for batch HAZOP emulation and for many other application areas.

While the field of application is currently based around the identification of hazards in chemical plants, the techniques being developed can in time be used to tackle the more general prob-

lem of qualitative physical systems modelling.

7 Conclusions

This paper has discussed some of the knowledge representation issues of particular importance to the field of hazard identification in chemical plants. It has examined two typical examples, pointing out where the most commonly used qualitative models, based on the signed directed graph, have had difficulties. We have made some suggestions for how a range of these problems can be tackled, in connection with the on-going research programme at Loughborough University, to further develop representations for modelling and simulating complex systems such as process plants efficiently.

Some techniques for coping with the complexity of process systems have already been proven and will be extended further (e.g. the definition of generic models of equipment and their inter-connection). Modelling the state of a plant and its component units appears to be the most important open problem in this domain, and forms the focus of current work on a research prototype for emulating batch HAZOP studies for chemical plant.

Initial work on emulating batch HAZOP has shown the feasibility of simulating the effects of deviations in operating instructions, and reporting the resulting hazards in a HAZOP-style output table. Much remains to be done, on extending the types of deviation guidewords which can be considered, and on focussing on only the most important deviations to be modelled in batch HAZOP.

Some of the important messages of this work so far are:

- Concentration on Petri Nets technology for representing operating plans is inappropriate for this domain. Petri Nets are not sufficiently flexible to model the effect of operations on a state-based plant model where either the plant or the instructions are subject to variation.
- Modelling actions in the plant with simple STRIPS operators will not work, because the problem of determining if an action will succeed is a non-local search, initiated at run-time, wherever flow is concerned.
- The connectivity of a plant may be determined dynamically by run-time conditions during

operation of the plant. In domains such as the example of making a cup of tea, connections may be contingent on spatial relationships between equipment items (e.g. the kettle must be above the cup in order to pour water from the kettle to the cup).

These points further emphasise that the best way to get realistic qualitative simulations of process systems is to build state-dependent models of the systems and to evolve their behaviour step-by-step through operation. CHECKOP is the first step towards achieving this level of realism and accuracy in hazard identification.

8 Acknowledgements

The authors gratefully acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC grant GR/R37531/01), in funding the work of Dingfeng Zhou on batch hazard identification.

References

[1] F.P. Lees. Process computer alarm and disturbance analysis: Outline of methods for systematic synthesis of the fault propagation structure. *Computers and Chemical Engineering*, 8(2):91–103, 1984.

[2] V. Venkatasubramanian, R. Rengaswamy, and S.N. Kavuri. A review of process fault detection and diagnosis, Part II: Qualitative models and search strategies. *Computers and Chemical Engineering*, 27:313–326, 2003.

[3] A. Hunt, B.E. Kelly, J.S. Mullhi, F.P. Lees, and A.G. Rushton. The propagation of faults in process plants (Parts 6-10). *Reliability Engineering and System Safety*, 39:173–250, 1993.

[4] P. Heino, E. Kotikunnas, W.F. Shei, C.C. Shao, and C.H. Chen. Computer-aided HAZOP with knowledge-based identification of hazardous event chains. *Loss Prevention and Safety Promotion in the Process Industries*, 1:645–656, 1995.

[5] Y. Shimada, K. Suzuki, and H. Sayama. Computer-aided operability study. *Computers and Chemical Engineering*, 20(6):905–913, 1996.

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

[7] F.I. Khan and S.A. Abbassi. Towards automation of HAZOP with a new tool EXPERTOP. *Environmental Modelling and Software*, 15(1):67–77, 2000.

[8] S.A. McCoy, S.J. Wakeman, F.D. Larkin, M.L. Jefferson, P.W.H. Chung, A.G. Rushton, F.P. Lees, and P.M. Heino. HAZID, a computer aid for hazard identification. *Trans IChemE, Part B (Process Safety and Environmental Protection)*, 77:317–353, November 1999. (Papers 1 to 3 in a series of 5).

[9] S.A. McCoy, S.J. Wakeman, F.D. Larkin, P.W.H. Chung, A.G. Rushton, and F.P. Lees. HAZID, a computer aid for hazard identification. *Trans IChemE, Part B (Process Safety and Environmental Protection)*, 78:91–142, March 2000. (Papers 4 and 5 in a series of 5).

[10] R. Vaidhyanathan, V. Venkatasubramanian, and F.T. Dyke. HAZOPExpert: An expert system for automating HAZOP analysis. *Process Safety Progress*, 15(2):80–88, 1996.

[11] ANSI/ISA. *ANSI/ISA-S88.01-1995 Standard – Batch Control, Part 1: Models and Terminology*. 1995.

[12] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

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

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

[15] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–388, 1986.

[16] Chemical Industries Association Limited. *A Guide to Hazard and Operability Studies*. Chemical Industry Safety and Health Council of the Chemical Industries Association, 1977.

[17] R.E. Knowlton. *A Manual of Hazard and Operability Studies: The creative identification of deviations and disturbances*. Chemetics International Ltd. (Vancouver, B.C.), 1992.

[18] H.G. Lawley. Operability studies and hazard analysis. *Chemical Engineering Progress*, 70(4), 1974.

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

[20] T. Kletz. *HAZOP and HAZAN (4th edition)*. Institution of Chemical Engineers (Rugby), 1999. ISBN: 0 85295 421 2.

[21] R. Batres, J. Soutter, S.P. Asprey, and P.W.H. Chung. Operating procedure synthesis: Science or art? *The Knowledge Engineering Review*, 13(3):261–294, 2002.

[22] J. Soutter and P.W.H. Chung. Utilising hybrid problem solving to solve operating procedure synthesis problems. In *Proc. 1997 IChemE Jubilee Research Event*, 1997.