Extending The Application Of Formal Methods To Analyse Human Error And System Failure During Accident Investigations

C.W. Johnson and A.J. Telford,
Glasgow Accident Analysis Group,
Department of Computing Science,
University of Glasgow, Glasgow, G12 8QQ.
Tel: +44 1413306053
Fax: +44 1413304913
Email: {johnson, alasg}@dcs.gla.ac.uk
http://www.dcs.gla.ac.uk/johnson, alasgtair

Abstract

Recent disasters at Bhopal, Chernobyl, Habsheim and Kegworth illustrate the point that software is rarely the sole cause behind major accidents. Operator intervention, hardware faults, even the weather conditions and malicious acts all combine to create the conditions for failure. In the aftermath of these accidents, it seems difficult for software engineers, systems developers, forensic scientists and interface designers to predict all of the ways in which systems can fail. It is, therefore, important that we learn as much as possible from those failures that do occur. Unfortunately, it is often difficult to gain a coherent overview from the mass of detail that is typically contained in many accident reports. This makes it difficult for readers to identify the 'catastrophic' events that produced the necessary conditions for disaster. This paper argues that formal specification techniques can be used to resolve these problems. In particular, an import 'Temporal Logic of Actions' is used to build a unified account of the human error and system failures that contributed to the Three Mile Island accident. This notation provides high-level abstractions that can be used to strip away the mass of irrelevant details that often obscure important events during disasters. Formal proof techniques can then be applied to the model as a means of identifying the causal relationships that must be broken in order to prevent future failures.

Keywords: Accident analysis, formal methods, human factors, Temporal Logic of Actions.

1 Introduction

Accident analysis is a multi-disciplinary activity [17, 25]. Forensic scientists, metallurgists, meterologists as well as software engineers and human factors experts all contribute to accident investigations [1]. The reports that are produced by these groups are, typically, divided into a number of chapters. Each of these sections reflects the perspective of a 'parent' discipline. This makes it difficult for readers to obtain a clear and coherent overview of the events leading to major accidents. Critical failures in one chapter are often missing in other sections. The importance associated with a particular event also varies within many accident reports. This can prevent readers from gaining an accurate impression of the primary and secondary causes leading to human and system failure [9, 10]. Unless regulatory authorities and companies are

convinced about the accuracy of accident reports then it will be difficult for them to accept the validity of their recommendations.

1.1 The Role Of Formality In Accident Analysis

This paper argues that the application of mathematical specification techniques can be extended from the fields of software and hardware engineering to describe the sequences of events that lead to major accidents. We intend to use formal representations to identify safety requirements after major accidents. This is markedly different from conventional approaches in which fault-trees and other formal notations are used to analyse failures before a system is built. A central objective in this work is to develop a common framework that can be used to analyse both system and human failures. This approach is appropriate because a number of commercial and regulatory organisations have mandated the use of formal notations in the engineering of safety-critical systems [3].

Human error has played an important role in the course of many major accidents. It is, therefore important that accident investigation techniques provide means of representing and reasoning about operator intervention. Unfortunately, mathematical specifications have not previously been used to support such analysis. They have, however, been used to support the design of interactive systems. For instance, Harrison and Thimbleby recruit algebraic notations to specify high level requirements for interactive systems [8]. Dix has extended this work to investigate timing problems in distributed, multi-user systems [5]. Sufrin and He have exploited the Z schema calculus to represent interaction with a text editor [21]. The following pages build on this work to show how formal specification and proof techniques can be extended from the field of software engineering and interface design to support the analysis of human factors and system failures.

Figure 1 presents an overview of the approach that is advocated in this paper. Traditional accident analysis techniques are employed by domain experts, human factors specialists, forensic scientists, software engineers and so on. The aim of this analysis is to identify the causal events that lead to failure. Their findings are, typically, expressed in natural language and they form

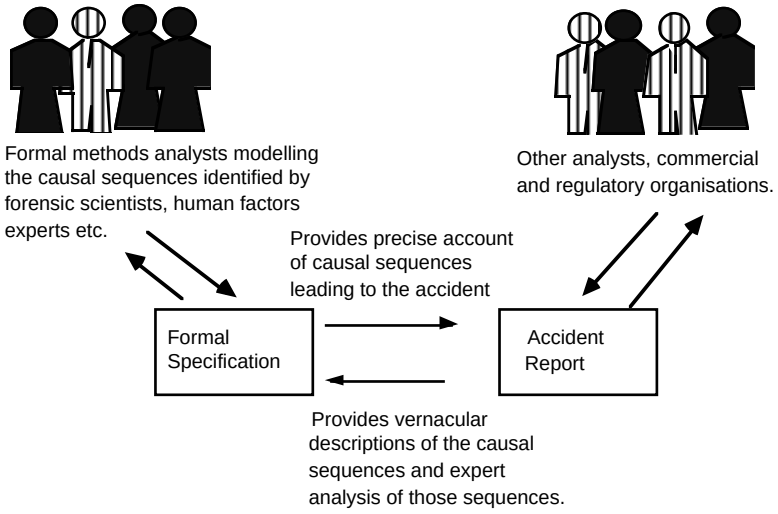

Figure 1: The Role of Formalism in Accident Analysis.

the basis of an initial accident report [13]. This paper argues that mathematical notations can

then be used to model the causal sequences that are identified in these reports. The formalisation process helps to extract these critical sequences from the mass of background detail that may be presented during an initial investigation. Essential information must be represented in the model, contextual details are omitted. A further benefit is that the resulting models are amenable to formal proof techniques. For instance, analysts can establish whether or not the causal sequences in a model actually do lead to the catastrophic events that occur during an accident. Finally, the formal model can be translated back into the natural language documents that communicate recommendations to commercial and regulatory organisations.

1.2 Graphical or Textual Notations?

The overview shown in Figure 1 raises a number of practical problems. For instance, there are no well-developed procedures for deriving formal models from the informal accounts in accident reports [9, 24, 23]. Similarly, there has been little work into the utility of formal notations for accident analysis. This is a significant omission because the choice of notation profoundly affects a non-formalist's ability to understand formal analyses [12]. For example, Figure 2 shows how the graphical Petri Net notation can be used to model the events leading to the Kegworth accident. This diagram was produced as a result of an earlier collaboration with a multi-disciplinary team of psychologists, linguists and systems engineers. The use of a graphical notation helped the team to discuss the causal sequences that were represented in the model [13]. It also helped to focus our analysis upon particular interpretations of the events leading to failure. For instance, the development of the model forced the team to address the debate about what the First Officer observed on the Airborne Vibration Monitor.

Unfortunately, Figure 2 also illustrates some of the problems that arise with a graphical notation. They quickly become intractable as more and more events are represented. The full diagram for the Kegworth accident takes many pages, even though the model is at a relatively high level of abstraction. Such problems can be reduced by structuring mechanisms. For instance, hierarchical Petri nets enable analysts to construct sub-networks, sub-sub-networks and so on. The problem with this is that important information may be hidden within the various layers of the model. A second issue is that the intuitive appearance of graphical notations must often be balanced against correspondingly more complex proof techniques. For example, the vector analysis that is used to establish liveness and safety in the sub-networks of a Petri net cannot easily be explained in terms of intuitive graphical models. This raises a third problem. Analysts may believe that they understand the consequences of introducing a new element into a graphical notation without fully comprehending the additional proof obligations that are created by their actions. For example, the introduction of an inhibitor arc into a Petri Net can invalidate all previous liveness results. Such problems justify the approach described in Figure 1. The 'specification' of the accident is not directly accessed by non-formalists. In contrast, the model is built and interpreted by trained analysts in much the same way that forensic scientists build and interpret continuous models of physical processes, such as the Health and Safety Executive's combustion simulators [11]. The U.K. Engineering and Physical Sciences Research Council has recently funded a three year project that is addressing these issues in more detail (GR/K55040). In anticipation of the results of this work, the remainder of this paper applies a textual notation to analyse the events leading to the Three Mile Island accident.

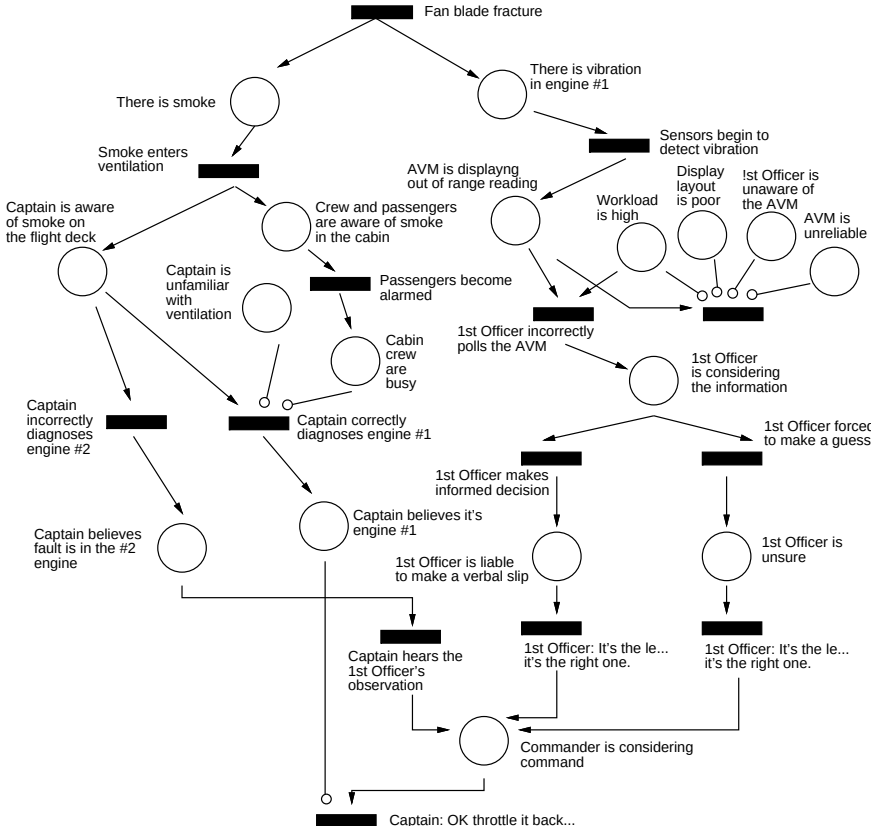

Figure 2: The Events Leading To The Kegworth Disaster.

1.3 Temporal Logic of Actions

The following pages exploit Lamport's Temporal Logic of Actions (TLA) [16]. This textual notation offers a number of benefits for accident analysis. It provides well developed proof techniques. It also provides means of representing the temporal properties that frequently have a profound impact upon the course of an accident. The underlying model in TLA is formed from an infinite series of states, known as behaviours. For example, $\langle\langle s_0, s_1, \ldots \rangle\rangle$ denotes a behaviour whose first state is s_0, its second state is s_1 and so on. From this we derive the definition for $\Box$ (read as 'always') given any formula in the logic F:

$$\langle\langle s_0, s_1, \ldots \rangle\rangle \models \Box F \stackrel{\Delta}{=} \forall n \in Nat : \langle\langle s_n, s_{n+1}, s_{n+2}, \ldots \rangle\rangle \models F \quad (1)$$

This operator can be used to represent invariant conditions that do not change during the course of an accident. The $\Diamond$ (read as 'eventually') operator can be defined as:

$$\Diamond F \stackrel{\Delta}{=} \neg \Box \neg F \quad (2)$$

This operator can be used to describe properties that occur at an indeterminate point during interaction. This is a significant advantage because, given limited sensing technology and partial sources of evidence, it is frequently impossible to develop a precise timeline for all of the events during an accident. The important point here is that analysts can use TLA operators, such as $\Box$, to be precise when they have the necessary evidence. Non-deterministic operators, such

as $\exists$, can be used when there is limited information about the temporal ordering of particular failures during major accidents.

The $\Box$ and $\exists$ operators are common features of many different temporal logics [18]. Lamport's Temporal Logic of Actions is distinguished from these notations in that elementary temporal formulas can be used to represent actions. An action A, is a relation between states. A boolean value can be obtained from applying the action to a pair of states, s, t. In other words, the action connects the old state, s, to the new state, t, by updating elements of that state. Lamport [16] provide additional information on the theoretical foundations of the notation. In contrast, the remainder of this paper focuses upon the application of TLA to support accident analysis.

1.4 The Three Mile Island Accident: Overview

The Three Mile Island Accident provides an appropriate example for this paper because it illustrates the pathological mix of system failures and operator errors that pose the greatest challenges for software engineers and human factors analysts. Figure 3 presents a schematic overview of the main components in this accident. The problems in this reactor started when the main feedwater system malfunctioned. This was the result of a demineraliser failure. The auxiliary feedwater supply was supposed to have started automatically. It did not because several manual valves in the auxiliary system had been inadvertently left closed after a test of the system. Without the feedwater, the steam generators dried out. This deprived the reactor of its main heat sink and so there was a rise in temperature and pressure inside the primary coolant circuit. The rise in pressure automatically shut-down the reactor but the radioactive core continued to give out heat. The pressure relief valve opened and became stuck so that there was a continuous release of radioactive water from the main steel pressure vessel into a drain tank and the containment sumps. Within two minutes the emergency coolant systems started automatically and began to increase the coolant level. This original loss of coolant accident

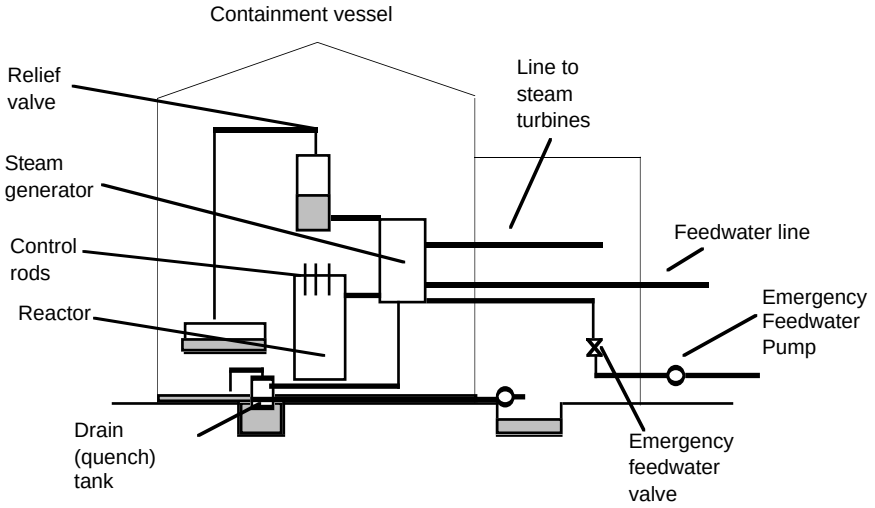

Figure 3: Diagram of the Three Mile Island Pressurised Water Reactor.

(LOCA) was exacerbated by the operators who responded to the pressure warning by manually overriding the two emergency core cooling injection pumps. By fifteen minutes into the accident,

there were almost 40,000 litres of primary coolant on the containment floor as the liquid began to escape from the sumps. Radioactive isotopes of krypton, iodine and xenon escaped to the environment through the building's ventilation system.

1.5 Outline Of The Paper

This section has introduced the argument that is presented in this paper. Section 2 builds on this and describes a high level framework that can be used to represent and reason about the operation of complex, interactive systems. This work has direct parallels with the abstract models that Lamport and others have developed to support the software engineering of concurrent programs. Section 2 then goes on to show how Lamport's model can be used to reason about human error and system failure. Unfortunately, this high level model is too abstract for accident investigations. Section 3, therefore, demonstrates that additional detail can be introduced to represent the specific human factors and system problems that lead to particular accidents. It is then possible to identify a 'catastrophic' subset of these problems which are the primary causes of an accident. Section 4 shows how these can be drawn from the causal sequences that are described in Section 3. It is the identification of these catastrophic failures that, traditionally, forms the heart of any accident investigation. The important point here is that the use of a formal notation provides a precise means of representing the causal relationships between failures that are often buried within the prose of natural language reports. Section 5 then goes on to demonstrate an additional benefit of our approach. Formal proof techniques can be applied to TLA descriptions in order to prove that the causal relationships which hold given a known set of failures and a normative model of system operation. In other words, we can validate our model of system failure by comparing the causal links in our specification to the known outcome of system failure and operator error. Section 6 summarises the conclusions from this work and suggests areas for future research.

2 A High Level Model For Accident Investigations

There are a number of key aims for any accident investigation [11]. In particular, analysts must identify:

- the starting point or initial circumstances for any accident;
- the normative or anticipated behaviour of the system;
- the abnormal events that led to system failures and operator 'error'.

This section shows how a model developed to support the software engineering of concurrent programs can also be used to provide a generic framework for these different activities. Subsequent sections will show how detail can gradually be added to this abstract model in order to identify the ways in which critical events combined during major accidents.

2.1 The Elements of Lamport's Model

Many similarities can be drawn between the development of concurrent programs and the generation of accident reports. Both activities force engineers to analyse asynchronous interaction between complex, autonomous processes. In software engineering, designers must account for the behaviour of programs interacting through asynchronous mechanisms or remote procedure calls. In

the case of accident investigation, analysts must examine the ways in which multiple operators interact with a range of hardware and software systems. These similarities are also apparent if one compares the aims of accident investigation, illustrated by the previous list, to the formal components of Lamport's model for the software engineering of concurrent programs [16]. This model describes the operation of a system, OperatingConditions, in terms of some initial conditions, InitialCondition, some changes to the state of the system, $2 [N]_f$, and a fairness condition, F:

$$\text{OperatingConditions} \triangleq \text{InitialCondition} \wedge \Box[N]_f \wedge F \tag{3}$$

For our purposes, an accident report can be thought of as the OperatingConditions, except that here we are describing the conditions for failure rather than conditions for the successful execution of a concurrent program. These reports must represent the initial conditions that hold before an accident occurs, InitialCondition. They must also consider the behaviour of the application during the accident, $2 [N]_f$. Here f represents the state of the accident components in terms of a number of variables. Finally, the formal model must ensure that each action had equal opportunity to occur | otherwise we would not model accurately the progression of events representing the incident. This is captured by the fairness condition, F, which applies to all the component actions of the next-state relation, N.

The strength of TLA with regard to accident analysis is that it enables refinements of sub-components of the accident scenario to be developed as more information becomes available to accident investigators. For example, even though the demineralizer of the plant is not mentioned in Bigelow and Fortune's account, it would be possible to add a TLA description of it at a later stage, if required. In addition, TLA allows an accident scenario to be described at both the temporal behavioural level and at the functional system level. The actions of the formalism are first-class processes which can model, at the level of detail required, the changes that occurred within various components of the disaster. We can, for instance, describe both simple temporal relationships such as water in the air-circulating cooling system switching off and the levels of temperature and pressure in the primary circuit surrounding the reactor core. For this reason of exibility we have chosen to use TLA over a simple predicate logic.

2.2 State Transitions (N)

In software engineering, designers must establish that certain output relations will eventually hold given a set of pre-conditions, InitialCondition, and some well-defined state transitions, N, [16]. In accident analysis, investigators must show how an accident eventually occurred given the basic operating conditions of the nuclear plant, N, and the set of initiating events described in their report, InitialCondition. This parallel between software engineering and accident analysis can be used to structure the formal analysis of our Three Mile Island example. For example, it is possible to introduce a causal relationship, A ; B (read as 'leads to'), that indicates the intended impact of a sequence of instructions in a concurrent program. These are the transitions described by N. This same relation can also be used to describe the cause-consequence chains that lead to major accidents. The following formula can be read as it is always the case that if A occurs then eventually B will happen:

$$A \; ; \; B \triangleq \Box(A \Rightarrow \Diamond B) \tag{4}$$

It should be noted that the ; operator formalises a relatively primitive notion of causality. It does not distinguish between necessary and su-cient conditions nor does it describe a struc-

ture for the events in causal sequences. Ladkin describes a number of more complex logics that can be used to capture detailed notions of causality [15]. Rather than adopt these models, this paper exploits the simple formalisation; to describe the relationships between critical failures and an eventual accident. In the following, OperatingConditions represents the conditions existing at the plant, prior to and during the accident. This was defined in (3). CatastrophicConditions represents the set of failures that occurred on 28th March 1979. These form the tradition focus of accident analysis:

$$OperatingConditions \Rightarrow (CatastrophicConditions \Rightarrow LossOf\_CoolantAccident) \quad (5)$$

This section has argued that the similarities between the software engineering of concurrent programs and accident analysis can be used to provide a formal structure for accident reports. Lamport's abstract model for the specification of parallel applications has been used to describe the events leading to major failures at an extremely high level of abstraction. This is important because it helps designers to strip away the clutter of low level details that can frequently obscure critical events during an accident. Unfortunately, such a high level of abstraction is inappropriate for many stages of accident analysis. It is, therefore, important that human factors analysts and system engineers can gradually introduce additional details as their investigations progress. This process is illustrated in the following section.

The Enabled predicate is true for an action A and a state if and only if it is possible to take an A step starting in that state. That is to say that there exists some other state which is related to by the action A. This predicate is vital to our formalisation of accident analysis in that we can use it to stipulate which events can occur in the model. In particular we, use it to denote the fact that the catastrophic actions, such as certain valves becoming waterlogged in the TMI plant, could take place.

2.3 Initialisation (Init)

A key stage in any investigation is the identification of a starting point for an accident. This may seem to be a trivial issue but an analysis of the reports that were produced in the aftermath of Three Mile Island indicates that this is not the case. For example, our account of the accident in Section 1.4 followed [7] in stating that the original cause was a demineraliser failure. Other reports blame the accident upon the failure of the air supply to an air operated valve [2]. It is clearly important to reduce such ambiguity if software engineers, hardware developers and interface designers are to co-ordinate the development of future systems. Formal methods can be used to provide an unambiguous representation of the initial circumstances that surround an accident. For example, the elements of TLA can be used to provide additional detail for our formalisation of $Init_C$ conditions. The following clause reflects the second interpretation mentioned above. It formalises the conditions for air supply failure rather than a demineraliser fault. The state of the demineraliser is omitted because in this analysis it is not considered to be a primary cause of the failure. Of course, this information might be included if analysts revised their interpretation of the events leading to the accident.

$$\begin{aligned} Init \triangleq\ & (relief\_valve = closed) \wedge (primary\_flow = normal) \\ \wedge\ & (steam\_turbine = on) \wedge (e\_pumps\_status = online) \\ \wedge\ & (e\_pumps = off) \wedge (air\_circuit\_status = clear) \\ \wedge\ & (secondary\_flow = normal) \wedge (ac\_valves = open) \end{aligned}$$

$$
\begin{aligned}
&\wedge\ (\mathrm{control\_rods} = \mathrm{up}) \wedge (\mathrm{ac\_pumps} = \mathrm{on}) \\
&\wedge\ (\mathrm{water\_runoff} = \mathrm{false}) \hat{=} (\mathrm{water\_temp} = \mathrm{normal}) \\
&\wedge\ (\mathrm{water\_press} = \mathrm{normal}) \wedge (\mathrm{feedwater\_line} = \mathrm{open}) \\
&\wedge\ (\mathrm{relief\_valve\_status} = \mathrm{working}) \hat{=} (\mathrm{hpp\_enable} = \mathrm{online}) \\
&\wedge\ (\mathrm{letdown\_valve} = \mathrm{closed}) \hat{=} (\mathrm{high\_pressure\_pumps} = \mathrm{off}) \qquad (6)
\end{aligned}
$$

The variables in this clause will be the value of f, in $2^{[N]_f}$, at the start of the accident. It should be stressed that this predicate is in itself an abstraction of the mass of more detailed information that may be presented to courts of enquiry or in subsequent accounts [17]. For example, primary_flow and secondary_flow indicate the flow of the water in the primary and secondary circuits. steam_turbine is an abstract representation of the state of the steam turbine. The e_pumps and e_pumps_status variables indicate whether the emergency feedwater pumps were on and whether they were enabled or not. The explicit representation of Initial Conditions is an important stage in the development of accident reports just as it is for the engineering of concurrent programs. In both cases, it helps to frame the subsequent design and analysis by providing a starting point for further consideration.

2.4 Fairness Conditions (F)

The second element of Lamport's model is the fairness condition F. In software engineering terms, this is used to ensure that events which must happen, eventually do happen:

$$
F \stackrel{4}{=} (\Box\Diamond \langle A \rangle_f) \vee (\Box\Diamond \neg \mathrm{Enabled} \langle A \rangle_f) \qquad (7)
$$

At first sight, it might seem strange to consider fairness conditions during the analysis of an accident. However, the previous disjunction precisely characterises the aim of accident reports. These documents are intended to describe how an accident eventually occurred so that it can never happen again. Indeed, we have strong fairness conditions covering both the ordinary actions of the plant, which we denote C, and the catastrophic actions, denoted as Catastrophic Conditions. This ensures that both the catastrophic actions and the operating actions may occur with equal priority in the model of the event. When we further specify that,

$$
\mathrm{Enabled} \mathrm{Catastrophic\_Conditions}
$$

we are saying that the latter part of the fairness disjunction is untrue and, as a consequence,

$$
\Box\Diamond \langle \mathrm{Catastrophic\_Conditions} \rangle_f
$$

This says that the catastrophic actions must occur, which is, naturally, consistent with the disaster scenario that we are formalising.

3 Causal Sequences ($\leadsto$)

One of the ways in which formal notations can add detail to high level accounts of accidents is by representing the causal chains that lead to major accidents. These chains are built from the operating conditions, Operating_Conditions, and the fact that the catastrophic events, Catastrophic_Conditions, that led to the loss of coolant accident, Loss_Of_Coolant_Accident, were enabled. An important technical point is that subsequent sections will elide the Enabled

predication in the interests of clarity. It is important to have an explicit representation of these sequences because it can be difficult for readers to extract causal relationships from the natural language accounts of major accidents. For example, one report includes the following lines:

> ...they [the maintenance crew] allowed some water to enter an air circuit that opened and closed some valves... The affected valves shut off... ([12], p. 12)

This quotation illustrates how the ordering and selection of materials used to imply a causal relationship between water entering the air circuit and the valves being shut off. The following clause makes this relationship explicit: Water_in_Air_Circuit denotes water being introduced into the air circuit and Valves_Switch_Off stands for the valves switching off in the circuit. This illustrates how causal chains are formed from the catastrophic events, Catastrophic_Conditions, such as the introduction of water into the air circuit, and from the operating conditions, Operating_Conditions, that determine the system's response to the failure:

$$\text{Water\_in\_Air\_Circuit};\ \text{Valves\_Switch\_Off} \tag{8}$$

Having constructed such a causal sequence, software and hardware engineers can trace the ways that subsequent events stemmed from these initial failures. For instance, the same report states that:

> The affected valves shut off and the pumps in the same circuit closed down in quick succession.

The same ; relation can be used to represent the consequences of the valve failure which in turn was caused by the water entering the air circuit. In the following clause, Pumps_Closing_Down is the action of the pumps closing down:

$$\text{Valves\_Switch\_Off};\ \text{Pumps\_Closing\_Down} \tag{9}$$

The important point here is that the ; relation forces analysts to consider the ways in which a wide range of failures contribute to major accidents. The initial events might simply relate to the hardware or systems engineering but the knock-on effects of those failures are quickly propagated throughout the application. In our example, the failure of the valves led to the pumps closing down. This led to the feedwater line being closed. The consequences of this was that the automated control systems intervened. They shut down the steam turbine and started up the emergency turbines. As software is increasingly being introduced into such safety-critical applications it is vital that designers understand how such knock-on failures affect the overall requirements for their systems. Unfortunately, it is not easy to extract these requirements from natural language accounts:

> One of the pumping circuits affected [by the valves switching off] was the feedwater line to the steam generation in the secondary circuit... so the safety system shut down the steam turbine and the electric power generator it drove. ...When the pumps supplying water to the steam generator stopped, three emergency pumps for the feedwater started.

As before, formal notations can be used to focus in upon the causal relationships that are embedded within such informal observations. Steam_Feedwater_Down represents the feedwater line to the steam generator being closed. Steam_Turbine_Shutdown represents the shutdown of

the steam turbine and Emergency_Pumps_Start is the action of the emergency pumps starting up:

$$\text{Pumps\_Closing\_Down} \;;\; \text{Steam\_Feedwater\_Down} \tag{10}$$

$$\text{Steam\_Feedwater\_Down} \;;\; (\text{Steam\_Turbine\_Shutdown} \wedge \text{Emergency\_Pumps\_Start}) \tag{11}$$

Thus far, we have shown how a hardware failure led to the intervention of control systems in order to start up emergency pumping equipment. These problems were exacerbated by operator intervention, or lack of it:

> ::: an operator ::: noticed that these pumps were running but he did not notice two particular warning lights on a control panel. They signalled that valves were closed in each of the two emergency feedwater lines and so were preventing water from reaching the steam generators, which soon boiled dry ::: Not until almost eight minutes later did the operators notice the closed valves and open them.

We use Emergency_Valves_Closed to represent the fact that the emergency feedwater line valves were closed and Steam_Generators_Dry stands for the steam generators boiling dry.

$$(\text{Steam\_Feedwater\_Down} \wedge \text{Emergency\_Pumps\_Start} \wedge \text{Emergency\_Valves\_Closed}) \;;\; \text{Steam\_Generators\_Dry} \tag{12}$$

The absence of operator intervention can be represented by the following clause:

$$(\text{Steam\_Feedwater\_Down} \wedge \text{Emergency\_Pumps\_Start} \wedge \text{Emergency\_Valves\_Closed} \wedge \neg\text{Operator\_Opens\_Valves}) \;;\; \text{Steam\_Generators\_Dry} \tag{13}$$

Such formulae require further refinement if they are to guide the future activities of interface designers, systems engineers and programmers. For instance, there must be some explanation as to why the operators did not intervene to open the valves, $\neg$Operator_Opens_Valves. The subsequent sections address this issue in greater detail. In contrast, the remainder of this section continues to build the causal chain of events leading to the Three Mile Island accident:

> Deprived of an outlet for the heat still entering from the reactor core, the water in the primary circuit increased in temperature and pressure: A relief valve on top of the pressuriser opened and steam and water began to flow out to a drain tank ::: To reduce the production of heat in the reactor core the control rods automatically descended into it :::

In the following, Primary_Circuit_Emergency stands for the emergency conditions which resulted in the primary circuit. In particular, there was a rise in the temperature and pressure in the circuit. Control_Rods_Drop denotes the fact that the control rods dropped into the reactor core and Relief_Valve_Open indicates the opening of the relief valve on the primary water circuit. Relief_Valve_Open represents the relief valve being open and Water_Run_Off denotes water run-off into the drain tank from the primary circuit:

$$\text{Steam\_Generators\_Dry} \;;\; \text{Primary\_Circuit\_Emergency} \tag{14}$$

$$\text{Primary\_Circuit\_Emergency} \;;\; \text{Control\_Rods\_Drop} \wedge \text{Relief\_Valve\_Open} \tag{15}$$

$$\text{Relief\_Valve\_Open} \;;\; \text{Water\_Run\_Off} \tag{16}$$

Formula (15) deserves particular attention because it illustrates how a causal chain of failure can have multiple consequences. An increase in temperature and pressure led to the introduction of the control rods and to the opening of relief valves. Such combined actions are often necessary in order to restore 'normative' states:

> The combined effects of lowering the control rods and opening the relief valve brought the temperature in the vessel down to normal: ::

The use of a formal notation would provide few benefits if it could not trace the ways in which automated systems and safety applications successfully intervened:

$$Control\_Rods\_Drop \wedge Water\_Run\_Off_i \; ; \; : Primary\_Circuit\_Emergency \tag{17}$$

This illustrates some of the complexities that can arise during accident investigations. It also emphasises the importance of tracing the full extent of; chains. The fact that the temperature and pressure problems were being reduced; $Primary\_Circuit\_Emergency$ did not mean that the accident was over. One of the relief valves was faulty and did not close as the pressure and temperature returned to normal:

> Through this valve more than one third of the contents of the primary circuit escaped. ::: two high pressure pumps started automatically, triggered by a lowering of pressure in the primary circuit.

The following clause, therefore, states that if there is no pressure emergency in the primary circuit, $Primary\_Circuit\_Emergency$ then water, $Water\_Run\_Off_i$, run-off will lead to a loss of pressure, $Primary\_Pressure\_Loss$. The second clause represents the observation that the high pressure pumps start, $High\_Pressure\_Pumps\_Start$, as a consequence of the loss of pressure if the pumps are enabled, $High\_Pressure\_Pumps\_Enable$:

$$(: Primary\_Circuit\_Emergency) \; (Water\_Run\_Off_i \; ; \; Primary\_Pressure\_Loss) \tag{18}$$

$$High\_Pressure\_Pumps\_Enable) \; (Primary\_Pressure\_Loss; \; High\_Pressure\_Pumps\_Start)_t \tag{19}$$

This final sequence provides the missing link between the initial ingress of water into the air circuit and the eventual loss of coolant accident (LOCA):

> The let-down valve was opened to release what was believed to be an excess of water. The original fall in pressure and the failure of the injection to bring down the temperature should have alerted the operators that Loss Of_Coolant Accident was underway but it did not.

The following clause states that a loss of pressure in the primary circuit, together with opening the let-down valve, Let_Down_Open, led to the $Loss of\_Coolant\_Accident$:

$$Primary\_Pressure\_Loss \wedge Let\_Down\_Open \; ; \; Loss Of\_Coolant\_Accident \tag{20}$$

It is important to re-iterate the main point of this section which is that the complexity of natural language descriptions makes it difficult to reconstruct the causal sequences that lead to major failures. The reader is left to interpret the relationships that are built up from natural language statements such as 'and then', 'at which time', and so... 'in quick succession' and 'they signalled that'. The use of the ; relation avoids such ambiguity. It also provides a concise means of representing the human errors and system failures that led to the loss of coolant accident.

4 'Catastrophic Events' (Catastrophic Conditions)

The previous section has constructed causal sequences of events that characterise the course of a major accident. These sequences are built from a number of failures and from the responses of the system and its operators to those failures (Operating Conditions). Having constructed these causal sequences, it is possible for analysts to extract those critical failures which were essential for the accident to progress (Catastrophic Conditions). For the Three Mile Island example, these catastrophic events can be represented as follows:

$$\text{Catastrophic\_Condition} \stackrel{4}{=} \text{Water\_in\_Air\_Circuit} \wedge \text{Relief\_valve\_Failure} \wedge \text{Emergency\_Valves\_Closed} \wedge \text{Let\_Down\_Open} \wedge \text{Emergency\_Pumps\_Ofi} \quad (21)$$

The accident was caused by the introduction of water into the air circuit and the failure of the relief valve and by the operators failing to close the relief valve and the opening of the let down valve in the primary circuit and the emergency pumps being disabled. In the above, Emergency_Pumps_Ofi refers to the emergency pumps being disabled and switched ofi:

$$\text{Emergency\_Pumps\_Ofi} \stackrel{4}{=} \text{:High\_Pressure\_Pumps\_Enabled} \wedge \text{:High\_Pressure\_Pumps\_Start} \quad (22)$$

These high level abstractions enable analysts to focus upon critical events. Hardware and software engineers are not forced to consider the mass of circumstantial detail that must be presented to accident inquiries.

4.1 The Causes of Catastrophic Events

Catastrophic Condition is the conjunction of the actions that contributed to the accident taking place. The accident occurred because these actions were enabled in some way. For example, (8) stated that the ingress of water into an air circuit caused the valves to switch ofi. This, in turn, led to the pumps being switched ofi. Unfortunately, clause (8) does not state the reasons why water was allowed to enter into the air circuit. This is a significant omission. Reason [20], Perrow [19], Leveson [17] all argue that analysts must trace the organisational factors that create the 'conditions for disaster'. Such problems can be avoided by extending the causal chain back to the 'primitive' conditions for an accident:

$$\text{Air\_Circuit\_Maintenance\_Error} \; ; \; \text{Water\_in\_Air\_Circuit} \quad (23)$$

This raises the problem of when to stop the formalisation process. Additional clauses can be introduced to represent the reasons for the maintenance failure. The ; operator might then be used to represent the causes for the reasons that lie behind the maintenance failure and so on. In practice, the decision to end the formalisation process depends upon the skill and experience of the analyst. The important point here is that the construction of the causal sequence forces analysts to explicitly consider the catastrophic events that frame an accident.

It is also important to note that the elements of Catastrophic Condition make no distinction as to the source of the 'failure'. In keeping with the title of this paper, we are equally concerned to model both human 'error' and system 'failure'. However, it is entirely possible to represent different categories within Catastrophic Condition:

$$\text{Operator\_error} \stackrel{4}{=} \text{Water\_in\_Air\_Circuit} \_ \text{Emergency\_Valves\_Closed} \_ \text{Let\_Down\_Open} \_ \text{Emergency\_Pumps\_Ofi} \quad (24)$$

This categorisation might be further refined to identify those events that are the side-effects of operator tasks, Emergency_Pumps_Off, as well as those that are the result of covert failures, such as Water_in_Air_Circuit, or mistakes based upon a misreading of available evidence, such as Let_Down_Open. Such an analysis would easily map onto Reason's taxonomy of error [20]. It is also possible to extract those critical events that were due to systems 'failures':

$$System\_failure \triangleq Relief\_valve\_Failure \tag{25}$$

The previous clause illustrates the point that TLA does not provide a panacea for accident analysis. It is entirely possible to disagree with our classification of human 'error' and system 'failure'. For instance, the fact that operators could incorrectly open the let down valve might be viewed as a system failure rather than an operator mistake. The notation does not replace the value judgements and skill-based decisions that must be employed to identify critical events from the mass of background detail. Similarly, it does not provide any automatic means of distinguishing the elements of Operator_error from System_failure.

It may also be noted that there is not universal agreement amongst those reporting upon the accident about its causes. For example, Fremlin has cited the possibility of a demineralizer error as the cause of the Three Mile Island accident [7]. This problem occurs within accident reports where different chapters reflect the interests of separate analysts. We investigate inconsistencies between reports in the forthcoming [22]. We are here, however, attempting to formalise and establish consistency within a single report only.

4.2 Refining Catastrophic Events

Clauses such as (21) and (22) are more detailed than the generic model for concurrent programs and accident analysis introduced in (3). Unfortunately, they are still not at a level that human factors analysts and programmers might use to guide the development of future applications. This problem can be addressed by representing the changes that critical events cause to process parameters. For example, the introduction of liquid into the air circuit caused its ducts to become waterlogged. The value of particular variables, such as air_circuit_status, can be directly related to the sensor readings that must be polled by control programs:

$$Water\_in\_Air\_Circuit \triangleq (air\_circuit\_status = clear) \wedge (air\_circuit\_status' = waterlogged)$$

It is important to emphasise that critical events will typically have no impact upon most of the variables in a system. This can be represented by introducing Unchanged(Var-S), to indicate that all variables in the system not in the set S are unchanged by a specified action:

$$Water\_in\_Air\_Circuit \triangleq (air\_circuit\_status = clear) \wedge (air\_circuit\_status' = waterlogged) \wedge Unchanged(Var - \{air\_circuit\_status\})$$

The failure of the relief valve was another catastrophic event that contributed to the accident. This event had no impact upon the rest of the system apart from causing the valve to fail. Again, this illustrates how the abstraction offered by formal notation can be used to scope or frame the impact of critical failures:

$$Relief\_valve\_Failure \triangleq relief\_valve\_status' = faulty \wedge Unchanged(Var - \{relief\_valve\_status\})$$

The previous clause represents the impact of a hardware failure. It is also possible to represent the impact of operator error in a similar manner. For example, the disabling of the high pressure coolant pumps played a major part in the events leading to the accident. The pumps were switched from a state in which they were enabled but off-line to one in which they had been turned off:

$$Emergency\_Pumps\_Off \triangleq (hpp\_enable = offline) \wedge (high\_pressure\_pumps' = off)$$

The fourth of the critical events was the closure of the emergency feedwater valves:

$$Emergency\_Valves\_Closed \triangleq (feedwater\_line = open) \wedge (feedwater\_line' = closed) \wedge Unchanged(Var - ffeedwater\_line)$$

The final critical factor in the lead up to the accident was the opening of the let down valve in the primary circuit. This had the effect of cutting off the primary coolant flow to the reactor:

$$LetDown\_Open \triangleq ((water\_press = low) \wedge (letdown\_valve' = open)) \Rightarrow primary\_flow' = none) \wedge Unchanged(Var - fprimary\_flow; letdown\_valve)$$

Similar definitions can be introduced for all of the events that have been introduced in our formal model of the accident. This is omitted for the sake of brevity and the interested reader is directed to [14]. In contrast, the following section argues that the formal proof techniques can be used to determine whether the causal sequences can be reconstructed from the failures in Catastrophic_Conditions and the operating procedures in Operating_Conditions.

5 Proofs and properties

The formal modelling of major accidents provides a number of benefits for human factors and software engineers. It avoids the ambiguity that can be found in natural language descriptions of causal relationships. It provides a precise means of framing an accident. TLA clauses can be used to describe the initial conditions for failure, in Critical_Conditions. A further advantage is that formal proof techniques can be used to determine whether the causal sequences described by $\rightsquigarrow$, can actually be produced from the known operating conditions of the system, Operating_Conditions, and the catastrophic system failures and human errors in Catastrophic_Conditions. For example, accident investigators might want to prove that if the operating procedures at Three Mile Island were followed then the introduction of water into the air circuit would lead to the valves being switched off. If this were not the case then either our causal analysis is incorrect, something else triggered the valves. Or our understanding of the operating conditions is incorrect:

$$Operating\_Conditions, \neg Enabled\_Water\_in\_Air\_Circuit \rightsquigarrow Closed\_Valves \quad (26)$$

Where Closed_Valves is defined as follows:

$$Closed\_Valves \triangleq ac\_valves = closed$$

As in previous sections of this paper, we can again recruit software engineering techniques to support accident investigation. In this case, we can use Lamport's (SF1) proof rule [16, page

22] to help us establish the previous relationship:

$$\frac{\begin{array}{l} P \wedge [M] \Rightarrow (P' \vee Q') \\ P \wedge \langle M \wedge Ai \rangle_v \Rightarrow Q' \\ 2P \wedge 2[M]_v \wedge 2F \Rightarrow 3Enabled\langle Ai \rangle_v \end{array}}{2[M]_v \wedge SF_v(A) \wedge 2F \Rightarrow (P \rightsquigarrow Q)}$$

The following substitutions can be made between our model of the Three Mile Island accident and the tokens in Lamport's rule:

$$\begin{array}{lll} P \wedge Enabled Water\_in\_Air\_Circuit & M \wedge N & Q \wedge Closed\_Valves \\ A \wedge C & v \wedge f & 2F \wedge SF_f(N) \end{array}$$

The first premise of SF1 can now be written as follows:

$$Enabled Water\_in\_Air\_Circuit \wedge [N]_f \Rightarrow (Enabled Water\_in\_Air\_Circuit' \vee Closed\_Valves') \quad (27)$$

This is satisfied trivially since, in this case, if $Enabled Water\_in\_Air\_Circuit$ is true then $Enabled Water\_in\_Air\_Circuit'$ must automatically be true. The second premise can be re-written as follows:

$$Enabled Water\_in\_Air\_Circuit \wedge \langle N \wedge C \rangle_f \Rightarrow Closed\_Valves' \quad (28)$$

This is true since we are assuming that the initial condition, $Initial Condition$, holds. Consequently, it must follow that air circuit status = water filled and subsequently from WCV that $Closed\_Valves'$ is true. The final premise can be written as:

$$2Enabled Water\_in\_Air\_Circuit \wedge 2[N]_f \wedge SF_f(Enabled Water\_in\_Air\_Circuit) \Rightarrow 3Enabled C_f \quad (29)$$

The above results from the fact that WCV is part of the disjunction which comprises C. Consequently, since we assume here that $Enabled Water\_in\_Air\_Circuit$ is always true then WCV must be enabled at some point. By applying SF1 we then obtain the following:

$$\begin{array}{l} Initial Condition \Rightarrow (2[N]_f \wedge SF_f(C) \wedge SF_f(Catastrophic Condition)) \Rightarrow \\ \quad (Enabled Water\_in\_Air\_Circuit \rightsquigarrow Closed\_Valves) \end{array} \quad (30)$$

Finally, we observe that (30) is isomorphic to

$$Operating Condition \Rightarrow (Enabled Water\_in\_Air\_Circuit \rightsquigarrow Closed\_Valves) \quad (31)$$

Such proof techniques suggest further ways in which our application of Lamport's model might benefit from software engineering techniques. For example, the transitivity of the $\rightsquigarrow$ relation suggests that certain sub-chains in our causal sequences might be omitted. In other words, if $A \rightsquigarrow B$ and $B \rightsquigarrow C$ then it may be possible to omit B altogether. An example of this may be seen in the above formalisation where the action $Closed\_Valves$, and the variable ac_valves, could be omitted. The action of the air circuit pumps switching off would then follow directly from the water being introduced into the air circuit. Such formal analysis might then be used to remove unnecessary parts from an accident report. This is similar to the ways in which pieces of redundant or inefficient computer code may be identified and removed through mathematical analysis. However, this proposal also illustrates some of the dangers that arise when transferring software engineering techniques directly to accident analysis. The purpose of writing an accident report is not to produce a minimal account but rather to produce a document that effectively communicates the weaknesses of previous systems. Removing such transitive relations may reduce the volume of an accident report but it can also hide links that help readers to understand critical sequences of human 'error' and system failure.

6 Conclusions and Future Work

This paper has argued that Lamport's model for the software engineering of concurrent systems can be used to support the generation of accident reports. This provides a number of benefits. The elements of Lamport's model can be used to identify generic requirements for accident investigations. In particular, multi-disciplinary teams must agree upon the starting point for an accident, Initial Conditions, the anticipated behaviour of a system, Operating Conditions, and the abnormal events that lead to failure, Catastrophic Conditions. The elements of the model can also be applied to structure the detailed analysis of the events leading to major accidents. The Temporal Logic of Actions notation can be used to construct the causal sequences of human 'error' and system failure that lead to disasters. The ; relation can be used to explicitly represent the informal observations that are often buried within informal accounts. Finally, proof techniques can be applied to the elements of the model to determine whether the ; relation actually does explain the circumstances leading to an accident.

The work reported in this paper is the initial product of an Engineering and Physical Sciences Research Council project into the formal analysis of human factors and system failure. Much remains to be done. In particular, a powerful application of our approach is that it can be used to detect inconsistencies not only within but also between accident reports. For example, our description of the Three Mile Island accident has been drawn from a number of sources, including [3], [7] and documents issued by the US President's Commission and the US Nuclear Regulatory Commission. These, in turn, were the distillation of evidence presented to the courts of enquiry. Our formalisation is, therefore, the result of a series of interpretations and simplifications. However, the techniques that are presented in this paper provide a precise and concise account of the events leading to major accidents. It is hypothesised that such formalisations might help to reduce the potential ambiguities that frequently arise in these different accounts.

A further benefit of formalisation is that it also provides engineers and analysts with proof techniques that can be used to verify cross-viewpoint consistency. This again illustrates the wider application of software engineering techniques for accident analysis. The need for such consistency has been recognised by a growing number of researchers in the field of software engineering [64]. This work is an important starting point for our future research. It is vital that those who are affected by the aftermath of an accident are left with a consistent view upon the implications of a report. This will mean, for instance, that consistency has to be shown between nuclear and software engineers. It is also necessary to maintain consistency between human-computer interaction specialists and those who are interested in the computational aspects of systems development.

Acknowledgement

Thanks are due to the referee whose comments helped to shape and focus the final version of this paper. Thanks also go to the members of the Glasgow Accident Analysis Group, to the Glasgow Interactive Systems Group and to the Formal Methods and Theory Group in Glasgow University. This work is supported by the UK Engineering and Physical Sciences Research Council grants GR/JO7686, GR/K69148 and GR/K55040.

References

[1] Air Accidents Investigation Branch, Department of Transport. Report On The Incident Involving BAC 1-11G-AYWB and Boeing 737 EI-BTZ On 12 April 1988 At Gatwick Airport, number 2/89, London, United Kingdom, 1989. Her Majesty's Stationery Office.

[2] V. Bignell and J. Fortune. Understanding Systems Failures. Manchester University Press, 1984.

[3] J. Bowen and V. Stavridou. Safety-critical systems, formal methods and standards. Software Engineering Journal, pages 189–209, July 1993.

[4] H. Bowman, J. Derrick, P. Linington, and M. Steen. Cross-viewpoint consistency in Open Distributed Processing. Software Engineering Journal, pages 44–57, January 1996.

[5] A. J. Dix. Formal Methods For Interactive Systems. Academic Press, London, United Kingdom, 1991.

[6] A. Finkelstein and I. Sommerville. The viewpoints FAQ. Software Engineering Journal: Special Issue on Viewpoints for Software Engineering, pages 2–4, January 1996.

[7] J. H. Fremlin. Power Production: What Are The Risks. Adam Hilger Ltd., Boston, United States of America, 1985.

[8] M. D. Harrison and H. W. Thimbleby, editors. Formal Methods In Human Computer Interaction. Cambridge University Press, Cambridge, United Kingdom, 1989.

[9] C. W. Johnson. The formal analysis of human-computer interaction during accident investigations. In G. Cockton, S. W. Draper, and G. R. S. Weir, editors, People And Computers IX. Cambridge University Press, Cambridge, United Kingdom, 1994.

[10] C. W. Johnson. The application of Petri Nets to represent and reason about human factors problems during accident analyses. In P. Palanque and R. Bastide, editors, The Second Eurographic Workshop On The Design, Specification And Verification of Interactive Systems, pages 345–357. Springer Verlag, Berlin, Germany, 1995.

[11] C. W. Johnson. Representing and reasoning about the impact of environmental layout upon human computer interaction. Ergonomics, 39(3):512–519, 1996.

[12] C. W. Johnson. Techniques for the evaluation of design notations. In M. D. Harrison and J. Vanderdonkt, editors, Design, Specification and Verification of Interactive Systems '96, Berlin, Germany, 1996. Springer Verlag.

[13] C. W. Johnson, J. C. McCarthy, and P. C. Wright. Using a formal language to support natural language in accident reports. Ergonomics, 38(6):1265–1283, 1995.

[14] C. W. Johnson and A. Telford. The formal analysis of human factors, errors and system failures. Technical Report TR-1996-6, Department of Computer Science, University of Glasgow, Glasgow, Scotland, 1996.

[15] P. Ladkin. Reasons and causes. Technical report, Technische Fakultat, Universitat Bielefeld, Germany, 1996. Available via http://www.techfak.uni-bielefeld.de/.

[16] L. Lamport. The temporal logic of actions. Research Report 79, DEC Systems Research Center, 130 Lytton Avenue, Palo Alto, California 94301, USA, December 1991.

[17] N. G. Leveson. Safeware: System Safety and Computers. Addison Wesley, Reading, United States of America, 1995.

[18] Z. Manna and A. Pnueli. Verification of concurrent programs: The temporal framework. In R. S. Boyer and J. Strother Moore, editors, The Correctness Problem in Computer Science, pages 215–273. Academic Press, London, United Kingdom, 1981.

[19] C. Perrow. Normal Accidents: Living With High-Risk Technologies. Basic Books, New York, United States of America, 1984.

[20] J. Reason. Human Error. Cambridge University Press, Cambridge, United Kingdom, 1990.

[21] B. Sufrin and J. He. Specification, refinement and analysis of interactive processes. In M. D. Harrison and H. W. Thimbleby, editors, Formal methods in Human Computer Interaction, pages 153–200. Cambridge University Press, Cambridge, United Kingdom, 1990.

[22] Alastair Telford and C. W. Johnson. Viewpoints for accident analysis. Technical report, Department of Computer Science, University of Glasgow, Glasgow, Scotland, 1996. Submitted to Formal Aspects of Computer Science.

[23] M. Thomas. A proof of incorrectness using LP: The editing problem in the Therac-25. High Integrity Systems Journal, (1):49–62, 1993.

[24] M. Thomas. The story of the Therac-25 in LOTOS. High Integrity Systems Journal, 1 (1):3–15, 1993.

[25] D. Woods, L. Johansen, R. Cook, and N. Sarter. Behind human error: Cognitive systems, computers and hindsight. Technical Report 94-01, CSERIAC State of the Art Report, 1994.