A PROBABILISTIC LOGIC FOR THE DEVELOPMENT OF SAFETY-CRITICAL, INTERACTIVE SYSTEMS

C.W. Johnson
The Human Computer Interaction Group, The Department of Computer Science,
The University of York, Heslington, The United Kingdom, YO1 5DD.
E-mail: johnson@minster.york.ac.uk, Telephone: (0904) 433376.

ABSTRACT

This paper starts from the premise that the human contribution to risk must be assessed during the development of safety-critical systems. In contrast to previous approaches, discrete numerical values are rejected as means of quantifying the probability of operator 'error' for many different users of many different systems. Numerical probabilities are used to rank the importance that designers attach to particular system failures. Adequate development resources must be allocated so that operators will resolve and not exacerbate high priority failures. In order to do this, human factors and systems engineers must be provided with notations that can represent risk assessments. Many techniques that are in widespread use, such as fault-tree analysis, provide inadequate support for the development of interactive systems. They do not capture the temporal properties that can determine the quality of interaction between operators and stochastic application processes. It is argued that probabilistic temporal logics avoid this limitation. Notations which are built around linear models of time cannot easily capture the semantics of risk assessments. We have developed Probabilistic Computation Tree Logic (PCTL) to avoid this problem. PCTL is built around a branching model of time. Finally, it is argued that PCTL specifications and Monte Carlo techniques can be used to provide faithful simulations of interactive systems. The implementation of the Risklog prototyping tool is briefly described. Partial simulations can be shown to system operators in order to determine whether they are likely to intervene and resolve system failures.

Keywords: Risk assessment; interface design; formal methods; prototyping; Risklog.

1 Introduction

The Commission of the European Community (1984), the Japanese Fifth Generation Initiative (Watson, 1985) and United States' Presidential Task Forces (1981) have all cited human intervention as a primary factor in the cause and exacerbation of accidents in safety-critical systems. Rasmussen (1985) asserts that "designs should only be accepted if the human contribution to risk can be measured". Reason (1990) argues that the potential for 'error' is latent within any man-machine system and that it is impossible to entirely engineer out the human contribution to risk. Instead, designers must strive to identify and protect systems against those risks which are thought likely to pose the greatest threat to safety. In order to do this engineers must be able to represent and reason about operators' responses to periodic system failures. This paper extends the application of probabilistic formalisms from Artificial Intelligence, control theory and statistical analysis to identify techniques which might enable operators to resolve high risk system failures.

1.1 Risk

The first question to be answered by any attempt to improve risk assessment is: what is risk? A number of different definitions have been proposed by bodies such as the British Standard's Institute (1979), the Institute of Chemical Engineers (1985) and the United States' Nuclear Regulatory Commission (Vesely, 1981). The variations in these definitions reflect the differing concerns of such agencies. This paper exploits a more general definition; risk is the "danger of loss, injury or other adverse consequence" (Allen, 1990). 'Danger' can be expressed in terms of probabilities; the likelihood of an adverse consequence occurring. It is seldom feasible to exhaustively test interactive systems in order to verify these probabilities. It is, typically, not feasible to test for errors which are estimated to occur once during ten years of operation. In spite of such limitations, risk assessment continues to provide an important design tool for many industries. Quantitative information about system failures is available from sources such as the European Event Data Recording System (Mancini, 1988), NASA's Aviation Safety Reporting System (Wiener, 1988) and the Atomic Energy Authority's Systems Reliability Service (Hunt and Ramskill, 1985). These sources are maintained because risk analysis provides a focus for the allocation of finite development resources. Designers strive to minimise those risks which are perceived to pose the greatest threat to their system. In other words, risk assessment helps to avoid ad hoc design by directing resources towards important development issues.

1.2 Formal Specification

NASA (1989), the U.K. Ministry of Defence (1991) and companies, including IBM (Jack, 1992) and Mitsubishi (Katsuyama, Sato, Nakakawaji and Mizuno, 1991), are investing considerable resources in the application of 'industrial strength' specification techniques. Formal methods have also been applied to support interface development. Sufrin and He (1990), Harrison and Thimbleby (1989), Took (1991) have used mathematical specification techniques to describe high level requirements for many different interactive systems. Unfortunately, many formal notations abstract away from temporal properties that can determine the quality of man-machine interaction. Delays in response times can lead to frustration and error (Johnson and Harrison, 1992). A further limitation is that many specification techniques do not distinguish between high and low probability system behaviours. This is significant because in manufacturing processes and telecommunications networks, designers can only specify traces of interaction with respect to the probable behaviour of other systems and users. For instance, operator commands to start pumping will not always be successful if pumps fail once every two days. This paper argues that probabilistic temporal logics can be used to support the design of such interactive systems.

1.3 Simulation

Risk assessments, expressed in mathematical specification languages, provide the non-formalist with little idea of what it would be like to interact with an application. We have implemented the Risklog simulation tool in order to determine whether executable subsets of a probabilistic temporal logic can be used to develop partial implementations. It is hypothesised that these prototypes would provide a far better impression of the 'look and feel' of an interactive system.

1.4 The Structure Of This Paper

Section 2 describes the example that is used to illustrate the rest of this paper. Section 3 argues that Probabilistic Computation Tree Logic (PCTL) can be used to represent the products of risk analysis in a form that supports the design of interactive systems. Section 4 argues that this notation can also be used to support prototyping The implementation of the Risklog simulation tool is briefly described. Sction 5 presents the conclusions that can be drawn from this research and identifies areas for further work.

2 An Example Application

The remainder of this paper uses requirements for a water-cooled nuclear reactor to illustrate the application of risk assessment techniques. This example has been chosen because it typifies the complex and stochastic application processes which pose a considerable challenge to interface designers (Worley and Lewins, 1988). A stochastic process is defined to be a series of operations whose results can best be described in terms of probabilities (Davies, 1985). In particular, we are concerned to represent and reason about operator intervention to resolve periodic failures in the emergency core cooling system. This provides short term cooling once the reactor enters an unstable state. Faults can occasionally develop in pumping equipment and so maintenance is required in order to guarantee the integrity of the system. If repair activities are prolonged then operators are required to halt power generation.

3 Representing Risk Assessments

In order to support the operation of stochastic application processes, designers must be able to represent low and high probability system behaviours.

3.1 Failure Modes, Effects And Criticality Analysis

A number of existing risk assessment techniques might be recruited to support interface development. For example, failure mode, effects and criticality analysis might be used to represent those situations that require operator intervention. Table 1 illustrates a product of this analysis. Operators must schedule increased maintenance in order to avoid the dried pump seals that can lead to over-heating. A limitation of these tables is that it can be difficult to identify the knock-on effects that faults have upon other components in the system. For example, Table 1 does not describe the impact which increased maintenance will have upon the production capacity of the plant.

Component	Failure Mode	Failure Cause	Criticality	Improvement
Pump Blow-back	Blockage	Valve collapse	Medium	Maintenance
Pump Seal	Over-heat	Dried	High	More inspection

Table 1: A Failure Mode, Effects And Criticality Analysis Table.

Similarly, it does not describe the effect which increased maintenance activities will have upon the operators who must control the power generation process. Table 1 might be extended to capture these relationships but this would reduce the tractability of the representation.

3.2 Markov Models

Markov models provide a means of representing the ways in which events interact (Leung and Wolfenden, 1983). Figure 1 illustrates an extremely simple model which relates failure and repair rates for pumping equipment. Using this model it is possible to derive the following formula which relates the probability

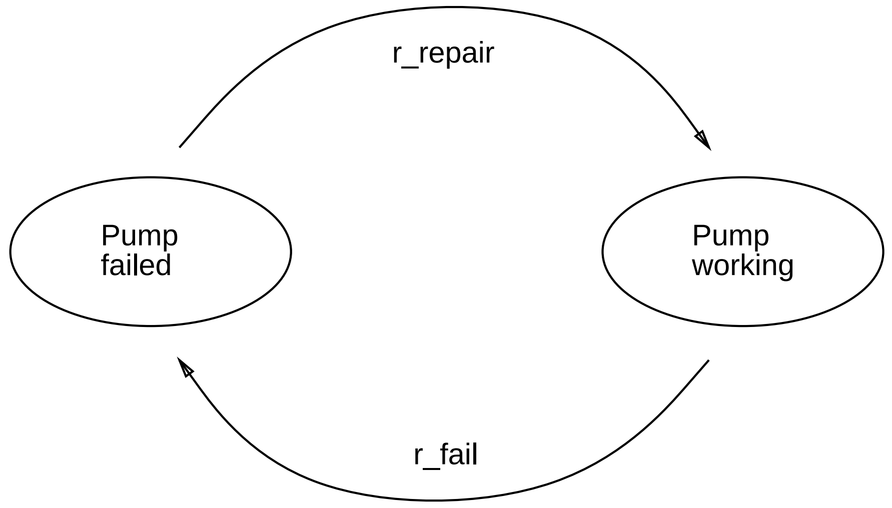

Figure 1: A Simple Markov Model

that a pump is working to its failure and repair rates at time t (Davies, 1985). This formula can be applied to deduce that if a coolant pump fails six times every year and can be repaired once every two days then there is a 0.96 probability that the component is working at any particular time:

$$|p\_working|_t = |\frac{1}{r\_fail + r\_repair} r\_repair + r\_fail\ e^{-(r\_fail + r\_repair)}|_t \qquad (1)$$

A limitation of this approach is that operator responses to system failures are represented implicitly in the transitions between system states. Users must intervene to transform a coolant pump from the failed to the working state. Explicitly representing operators as components of a Markov model begs the question: what states can a user be in? In anticipation of the results of current research in the field of cognitive science, designers must seek alternative approaches (Rasmussen, 1988).

3.3 Fault Trees

Fault-trees are in widespread use amongst systems engineers (Vesely, 1981). For instance, the fault-tree notation used in Figure 2 conforms to guidelines that have been drawn up by the European Federation of Chemical Engineers (1985). The bottom-right branch of this fault-tree represents the risk that an operator will attempt to withdraw control-rods whose interlocks have been disabled. The probabilities and the method of propagation are derived from Watson (1985). There are, however, limitations which restrict the utility of this notation for the design of interactive systems. The European Federation of Chemical Engineering's International Study Group On Risk Analysis concludes:

"Fault-trees have difficulties with event sequences... parts of systems where sequence is important are, therefore, usually modelled using techniques more adept at incorporating such considerations" (European Federation of Chemical Engineering, 1985).

Gate conditions are assumed to hold simultaneously or at some time after their branch conjunctions and disjunctions. The leaves of a tree are assumed to hold simultaneously or at some time before their parents. This lack of temporal information is a significant limitation because event sequences affect the quality of interaction between an operator and their system. Determining the order in which displays are presented and commands are issued forms an important stage in the development of man-machine interfaces.

3.4 Probability Logic

Logic offers a number of advantages for the design of man-machine systems. It can be used to abstract away from the complexity of low level device handling. These details can gradually be introduced as development progresses. There is a close correspondence between specifications written in logic and programs which satisfy those specifications implemented in PROLOG. As early as 1938, Shannon proposed that logic might also be applied to support risk analysis for electrical control systems. Rescher describes one means of achieving this:

"A measure function Pr is presupposed as given, which assigns some real value $Pr(p)$ to each and every member p of the domain of statements at issue" (Rescher, 1969).

The underlying idea behind this approach is that a likelihood value can be assigned to statements. The Pr function is assumed to satisfy constraints imposed by probability theory. For instance, the following are typically provided as axioms:

$$0.0 \leq Pr(p) \leq 1.0 \tag{2}$$
$$Pr(p \vee \neg p) = 1.0 \tag{3}$$

Carnap's *Logical Foundations Of Probability* (1962) reviews a number of semantic interpretations for these clauses. For our purposes, it is possible to apply Rescher's probability logic to represent the risk of system behaviours which require operator intervention. Figures for compressor performance predict 76 failures per operating year (Davies, 1985). Designers might, therefore, specify that the probability of a pump failing in the next second is 0.241×10^{-5}:

$$Pr(pump\_failed_a) = 0.00000241 \tag{4}$$

Such probabilities can be used to represent and reason about the potential behaviour of man-machine interfaces. For example, the following formula states that an error message is received by a user if $pump_a$ fails, a warning is presented and $user_1$ confirms the error message. In order to determine the likelihood of a $pump\_warning_a$, designers must first determine the probability of $pump\_failed_a$. Clause (4) indicates that this is likely to be a rare warning. Additional design resources may, therefore, be allocated in order to ensure that operators can detect and respond appropriately to such displays:

$$\begin{aligned} &l\_error\_received \Leftarrow \\ &\quad pump\_failed_a \wedge pump\_warning_a \wedge user\_warning\_acknowledge_1 \end{aligned} \tag{5}$$

There is no notion of sequence in probability logic; formula (5) would be true if $pump\_warning_a$ was presented after the user acknowledged the warning. This would have a profound impact upon the usability of any implementation and is clearly not what the designer intended. Probability logic, therefore, suffers from the same limitation as the fault-tree notation.

3.5 Linear Time Temporal Logics

Linear time temporal logics provide a means of representing the sequencing information that was absent from clauses such as (5). The following paragraphs describe a language proposed by Manna and Pnueli (1981). Readers who are concerned with the application, rather than the formal basis of the logic, should move onto the next paragraph. Elements of a set of symbols represent variables, constants, propositions, functions and predicates. This set can be partitioned into global and local symbols. Global symbols have a uniform interpretation; they do not change their value or meaning from state to state. Local symbols may assume different meanings and values from state to state. The logic contains the set of boolean connectives: $\wedge; \vee; \neg; \Leftrightarrow; \Rightarrow; \Leftarrow$. The logic also contains the first-order quantifiers $\exists$ and $\forall$ together with the temporal operators: $\bigcirc$ (read as 'next'); $\Box$ ('read as always'); $\Diamond$ (read as 'eventually'); $\mathcal{U}$ (read as 'until'). Terms are built from the application of functions and predicates to constants and variables. Formulas are constructed from the application of boolean connectives, temporal operators or quantifiers to terms. Atomic formulas are propositions and predicates applied to terms which are of an appropriate sort. A model (I, α, δ) for Manna and Pnueli's temporal logic consists of a global interpretation, I, a global assignment, α, and a sequence of states, δ. I specifies a domain corresponding to each sort and assigns concrete elements of that domain to symbols. α assigns a value over an appropriate domain to global free variables and propositions. $\delta = s_0, s_1, ...$ is an infinite sequence of states where each s_i assigns values to local free variables and propositions. The following provides an inductive definition of the truth value of temporal formula w in a model (I, α, δ). I is implicitly assumed, the value of a subformula or term, τ, is denoted by $\tau \mid_{\delta}^{\alpha}$:

- For a local variable or proposition, y: $y \mid_{\delta}^{\alpha} = y_{s_0}$. The value of y in state s_0 is that assigned in the first state of δ.
- For a global variable or proposition u: $u \mid_{\delta}^{\alpha} = \alpha[u]$. The value of u is that assigned to u by α.
- For a constant, c, the evaluation is determined by I: $c \mid_{\delta}^{\alpha} = I[c]$.
- For a function, $f(x_1, ..., x_k)$: $f(t_1, ..., t_k) \mid_{\delta}^{\alpha} = I[f](t_1 \mid_{\delta}^{\alpha}, ..., t_k \mid_{\delta}^{\alpha})$. The value of the application is that of the interpreted function, I[f], applied to the values of $t_1, ..., t_k$ in (I, α, δ).
- For a term t: $\bigcirc t \mid_{\delta}^{\alpha} = t \mid_{\delta^1}^{\alpha}$. The value of $\bigcirc t$ in $\delta = s_0, s_1, ...$ is given by the value of t in the shifted sequence $\delta^1 = s_1, s_2, ...$
- For a predicate $p(x_1, ..., x_k)$: $p(t_1, ..., t_k) \mid_{\delta}^{\alpha} = I[p](t_1 \mid_{\delta}^{\alpha}, ..., t_k \mid_{\delta}^{\alpha})$.
- For disjunction: $(w_1 \vee w_2) \mid_{\delta}^{\alpha} = true$ iff $w_1 \mid_{\delta}^{\alpha} = true$ or $w_2 \mid_{\delta}^{\alpha} = true$.
- For negation: $(\neg w) \mid_{\delta}^{\alpha} = true$ iff $w \mid_{\delta}^{\alpha} = false$.
- For $\bigcirc$ application: $\bigcirc w \mid_{\delta}^{\alpha} = w \mid_{\delta^1}^{\alpha}$.
- For $\Box$ application: $\Box w \mid_{\delta}^{\alpha} = true$ iff for every $k \geq 0, w \mid_{\delta^k}^{\alpha} = true$. w is true for all suffix sequences of δ.
- For $\Diamond$ application: $\Diamond w \mid_{\delta}^{\alpha} = true$ iff there exists $k \geq 0, w \mid_{\delta^k}^{\alpha} = true$. w is true in at least one suffix of δ.
- For $\mathcal{U}$ application: $w_1 \mathcal{U} w_2 \mid_{\delta}^{\alpha} = true$
  iff for some $k \geq 0, w_2 \mid_{\delta^k}^{\alpha} = true$ and for all $i, 0 \leq i < k, w_1 \mid_{\delta^i}^{\alpha} = true$.
- For universal quantification: $(\forall u.w) \mid_{\delta}^{\alpha} = true$ iff for every $d \in D, w \mid_{\delta}^{\alpha'} = true$. Where $\alpha' = \alpha \circ [u \leftarrow d]$ is the assignment obtained from α by assigning d to u. D is the domain over which u ranges.

- For existential quantification: $(\exists u.w) \mid_{\delta}^{\alpha} = true$ iff for some $d \in D, w \mid_{\delta}^{\alpha'} = true$. Where $\alpha' = \alpha \circ [u \leftarrow d]$.

Linear temporal logics have previously been applied to represent and reason about computer programs (Lamport, 1982, Saake and Lipeck, 1988). For our purposes they might also be used to represent temporal properties of interactive systems. The $\bigcirc$ operator can explicitly represent the temporal sequencing that was missing in (5). The following clause states that an error message is received by a user if in the present interval $pump_a$ fails and in the next interval a warning is presented and in the next again interval $user_1$ confirms the error message:

$$\begin{aligned} &ltl\_error\_received \Leftarrow \\ &\quad pump\_failed_a \wedge \bigcirc(pump\_warning_a \wedge \bigcirc user\_warning\_acknowledge_1) \end{aligned} \tag{6}$$

The $\Diamond$ operator can be used to represent liveness properties. These must eventually be satisfied in order to ensure safe and successful operation. Clause (6) is extremely deterministic. The operator must respond to the warning in the next interval after it is presented. Given divided attention and a noisy working environment it may only be possible to ensure that the operator eventually responds to the warning:

$$\begin{aligned} &ltl\_eventual\_error\_received \Leftarrow \\ &\quad pump\_failed_a \wedge \bigcirc(pump\_warning_a \wedge \Diamond user\_warning\_acknowledge_1) \end{aligned} \tag{7}$$

The $\Diamond$ operator represents non-deterministic requirements; it can be evaluated as true in the present interval, in the next interval, in the next next interval and so on. Many safety-related properties are invariant; they do not change throughout interaction. These can be represented using the $\Box$ operator, read as 'always'. Designers might use $\Box$ to specify that it is never the case that $pump\_warning_a$ is presented and $pump_a$ has not failed:

$$\begin{aligned} &ltl\_always\_display\_warning \Leftarrow \\ &\quad \Box\neg(pump\_warning_a \wedge \neg pump\_failed_a) \end{aligned} \tag{8}$$

The introduction of temporal operators into a logic notation can also be used to express temporal properties of probabilities. For example, the following clause states that there is less than a 1% chance that $pump_a$ is down in the present interval. It is possible that at some future interval this probability will increase but it is less than 0.01 now:

$$Pr(pump\_failed_a) < 0.01 \tag{9}$$

In contrast, designers might use the $\Box$ operator to specify that the probability of $pump_a$ failing is always less than 0.01:

$$\Box(Pr(pump\_failed_a) < 0.01) \tag{10}$$

These clauses might be used to inform interface development. For any operator, the failure of *pump_a* will be a rare event. This might persuade interface designers to present diagnostic information in addition to $pump\_warning_a$. In Norman's terms, sufficient context must be provided 'in the world' to ensure that operators are not forced to rely upon 'in the head' recollections of emergency procedures for rare failures (Norman, 1990). It is vital that such requirements should emerge early in development when sensing devices can be allocated to meet the informational requirements of system operators. If the requirement specified in (10) were violated and the probability increased beyond 0.01 then displays might be re-designed in order to reflect the greater experience which operators would have of this error condition. Less contextual information may be required in order for them to diagnose and resolve the failure.

Linear temporal logic can also be used to explicitly represent the ways in which risk assessments change during particular traces of interaction. For example, it might be specified that in the present interval $pump_a$ has failed and the risk of an unstable reactor is greater than 0.01 and in the next interval the operator intervenes to shut the system down and the risk of an unstable reactor is reduced to 0.0005:

$$\begin{aligned}&ltl\_changing\_risks \Leftarrow \\ &\quad pump\_failed_a \wedge Pr(reactor\_unstable) > 0.01 \wedge \\ &\quad user\_reactor\_close\_down \wedge Pr(reactor\_unstable) = 0.000005 \qquad (11)\end{aligned}$$

There is a major problem with probabilistic extensions to linear temporal logics. It is possible to provide a number of alternative semantic interpretations for the Pr function (Carnap, 1962). In particular, Pr may be viewed either as a measure of confirmation or as a measure of frequency. The former interpretation resembles the Bayesian view; probability is contingent upon the observation of certain evidence. This approach is implicit in (11); there is a 0.01 probability of reactor instability due to pump failure. The frequency view is closer to the use of probability in clause (4). The likelihood of pump failure was determined by the observation of previous errors and was specified without any implicit contingencies. Branching time temporal logics provide a means of avoiding such distinctions by explicitly specifying that the semantics of the Pr function should be based upon the frequency view of probability.

3.6 Branching Time Temporal Logics

There are a number of reasons why designers might prefer the frequency view of probability over measures of confirmation. The most important of these is that the published sources for reliability figures tend to use this approach. Our estimate of pump failures was derived from the maintenance records of existing plants. Measures of confirmation can also be criticised from a philosophical stand-point:

> "... for most, perhaps for practically all, of those authors on probability who do not accept a frequency conception the following holds. (i) Their theories are objectivist (and) are usually only preliminary remarks not affecting their actual working method. (ii) The objective concept which they mean is similar to (the frequency view of) probability." (Carnap, 1962).

This observation does not take into account the pragmatic benefits of techniques such as Bayesian conditioning and Bayesian networks (Van Rijsberg, 1992). It must also be stressed that the integration of Bayesian and frequency approaches is an area of continuing research. Frequency predictions can be bounded by ranges which reflect the degree of belief in the probability of a fact being true (Bacchus, 1990). Further work intends to examine whether the benefits of this integration provide sufficient justification for the additional complexity created by reasoning with bounded probabilities. In anticipation of the results of this research the remainder of this paper exploits a frequency based perspective upon probability.

In order to provide a semantics for the Pr function it is first necessary to describe a non-probabilistic branching time temporal logic. Readers who are concerned with the application, rather than the formal basis of the logic, should move onto the next paragraph. Clarke and Emerson's Computation Tree Logic (CTL) represents future time in terms of a number of possible sequences of states (Clarke and Emerson, 1982). This contrasts with Manna and Pnueli's logic where the notation is built around a single sequence of future states. Following Clarke and Emerson, our description of CTL uses p to denote an atomic proposition. p is a formula, f_i denotes a formula or sub-formula. $f_1 \wedge f_2$ is a formula denoting conjunction, $\neg f$ is a formula denoting negation. $EX\ F_j f$ is a formula; f is true in the next state of possible future F_j. $A[f_1 \mathcal{U} f_2]$ is a formula; there exists a prefix for all futures such that f_2 is true in the last state of the prefix and f_1 holds at all other states in that prefix. $E[f_1 \mathcal{U} f_2]$ is a formula; there exists a prefix for a future such that f_2 is true in the last state of the prefix and f_1 holds at all other states in that prefix. The semantics of CTL are represented by a model $M = (S, F_1, ..., F_k, L)$ where: S is a countable set of states; $F_i \subset S \times S$ is a relation on S giving the transitions over a possible future i; L is an assignment of true atomic propositions in each state. Let $F = F_1 \cup F_2 ... \cup F_k$ and $\forall s \in S, \exists s' \in S(s, s') \in F$. A *trace* is an infinite sequence of states $(s_0, s_1, s_2, ...) \in S^\omega$ where $\forall i(s_i, s_{i+1}) \in F$. To any structure M and state $s \in S$ of M there corresponds a tree of possible future states with root labelled s_0 such that $s \rightarrow s_i$ is a branch in the tree iff $(s, s_i) \in F$. M, $s_0 \models f$ denotes that formula f is true in state s_0 of future tree M. The M is omitted if it is understood by the preceding context. The $\models$ relation can be defined inductively:

- $s_0 \models p$ iff $p \in L(s_0)$;
- $s_0 \models \neg f$ iff not$(s_0 \models f)$;
- $s_0 \models f_1 \wedge f_2$ iff $s_0 \models f_1$ and $s_0 \models f_2$;
- $s_0 \models EX\ F_k f$ iff $\exists s_1 \in S(s_0, s_1) \in F_k \wedge s_1 \models f$;
- $s_0 \models A[f_1 \mathcal{U} f_2]$ iff for all traces $\exists i[i \geq 0 \wedge s_i \models f_2 \wedge \forall j(0 \leq j \wedge j < i \wedge s_j \models f_1)]$;
- $s_0 \models E[f_1 \mathcal{U} f_2]$ iff for some trace $\exists i[i \geq 0 \wedge s_i \models f_2 \wedge \forall j(0 \leq j \wedge j < i \wedge s_j \models f_1)]$;
- $f_1 \vee f_2 \equiv \neg(\neg f_1 \wedge \neg f_2)$;
- $f_1 \Rightarrow f_2 \equiv \neg f_1 \vee f_2$;
- $EFf_1 \equiv E[true \mathcal{U} f_1]$ states that for some trace there exists a state in which f_1 is true;
- $AGf_1 \equiv \neg EF \neg f_1$ states that f_1 is true for all states in all traces;
- $AXf \equiv \neg EX \neg f_1$ states that f_1 holds in all next states.

Designers can use CTL's X operator (read as 'next') to represent the sequential information that was missing in formula (5). The A operator can be read as 'in all traces of interaction'. For example, the following clause specifies that an error is always received if pump a fails, a warning is presented and under all possible futures $user_1$ acknowledges the warning in the next state:

$$\begin{aligned} &ctl\_always\_receive\_error \Leftarrow \\ &\quad pump\_failed_a \wedge display\_pump\_warning_a \wedge AX\ user\_warning\_acknowledge_1 \end{aligned} \tag{12}$$

Similarly, it is possible to represent the liveness property described using the $\diamond$ operator in (7). This requirement can be specified using the E operator (read as 'in a possible trace of interaction') and the F operator (read as 'in a future state'). $user_1$ responds to the warning in a future state of at least one possible course of interaction:

$$\begin{aligned} &ctl\_weak\_receive\_error \Leftarrow \\ &\quad pump\_failed_a \wedge display\_pump\_warning_a \wedge EF\ user\_warning\_acknowledge_1 \end{aligned} \tag{13}$$

The previous clause does not require that the warning display should continue to be presented until the user acknowledges it. Such presentation requirements can be represented using the $\mathcal{U}$ (read as 'until') operator:

$$\begin{aligned} &ctl\_continue\_display \Leftarrow pump\_failed_a \wedge E[display\_pump\_warning_a\ \mathcal{U} \\ &(user\_warning\_acknowledge_1 \wedge user\_start\_stand\_by\_pump_1)] \end{aligned} \tag{14}$$

Designers might also exploit the $\mathcal{U}$ operator to represent requirements for multi-user systems. For instance, it might be specified that a number of operators should coordinate their responses before stand-by pumping equipment is started. Cooperation occurs if pump a fails, a warning is presented and in all possible futures the stand-by pump is not started until both user 1 and user 2 provide input to start that pump:

$$\begin{aligned}&ctl\_cooperation \Leftarrow pump\_failed_a \wedge display\_pump\_warning_a \wedge \\ &A[\neg stand\_by\_pump\_on\ \mathcal{U}\ (user\_start\_stand\_by\_pump_1 \wedge user\_start\_stand\_by\_pump_2)]\end{aligned} \qquad (15)$$

This requirement can easily be generalised to more than two operators using additional clauses to represent $user_3$, $user_4$ and so on (Johnson, 1991).

3.7 PCTL

We have developed PCTL (Probabilistic Computation Tree Logic) in order to represent probabilities within the CTL notation. This can be achieved by extending the syntax of CTL to include an additional path formula. $AX[f\ Pr\ y]$ intuitively means that the probability of f being true in the next state of all possible futures is y where y is a value in the range [0.0,1.0]. The probability of a proposition is given by the number of next possible states in which that proposition is true divided by the number of all possible next states. This approach is similar to that exploited by Hansson and Jonsson (1989). For instance, if $pump_a$ fails in two out of ten possible states then the probability of $pump_a$ failing in the next state is 2/10 or 0.2. More formally, $s_0 \models AX[f\ Pr\ y]$ iff:

$$\frac{\#\{\forall F_k \in F | (s_0, s_1) \in F_k \wedge s_1 \models f \bullet F_k\}}{\#\{\forall F_j \in F | (s_0, s_1) \in F_j \bullet F_j\}} = y$$

It is, typically, impossible to enumerate all possible next states for complex systems. Therefore, the historic frequency $[f\ Pr\ y]$ is taken as a best approximation for $AX([f\ Pr\ y])$. We cannot predict the future and so design is informed by frequency rates which have been derived from previous traces of interaction. The semantics of this frequency view can be represented by altering CTL's model to include a partial function $P \nrightarrow S \times S$ which defines a linear ordering over previous states. In PCTL a trace can be viewed as an infinite sequence that includes a finite sequence of previous states: $(s_x, ..., s_{-2}, s_{-1}, s_0, s_1, s_2, ...) \in S^{\omega'}$ where $\forall i \geq 0 (s_i, s_{i+1}) \in F$ and $\forall i < 0 (s_{i-1}, s_i) \in P$. From this we can now derive the semantics of the frequency view of probability. $s_0 \models [f\ Pr\ y]$ iff:

$$\frac{\#\{\forall s_x \in P | s_x \models f \bullet s_x\}}{\#\{\forall s_x \in P \bullet s_x\}} = y$$

The probability, y, of f being true in the present state is given as the total number of previous states in which f was true divided by the total number of previous states. As mentioned in Section 3.4, figures for compressor performance predict 76 failures per operating year. Such real-time failure rates can be represented in PCTL by linking all state transitions to a clock, $\forall i (s_i, s_{i+1}) \in S$. If state transitions, (s_i, s_{i+1}), occur every second then designers might specify that:

$$[pump\_failed_a\ Pr\ 0.00000241]$$

It is important to note that there are many well developed techniques for the combination and analysis of such probabilities. The interested reader is directed to Bacchus (1990) for a more detailed discussion. In contrast, the following clauses are intended to illustrate the ways in which PCTL might be used to inform the design of man-machine interfaces. The first clause states that an error is acknowledged if pump a fails and a warning is displayed and there is at least one state in a possible course of interaction in which user 1 provides input to acknowledge the failure. The second clause states that an error is acknowledged if pump a fails and a warning is displayed and there is at least one state in a possible course of interaction in which user 2 provides input to acknowledge the failure:

$$
\begin{aligned}
&acknowledge\_error \Leftarrow \\
&\quad pump\_failed_a \wedge display\_pump\_warning_a \wedge EF\ user\_warning\_acknowledge_1 \qquad (16)\\
&acknowledge\_error \Leftarrow \\
&\quad pump\_failed_a \wedge display\_pump\_warning_a \wedge EF\ user\_warning\_acknowledge_2 \qquad (17)
\end{aligned}
$$

From (16) we know that pump failures are unlikely to occur during most shifts. In Section 3.4 it was argued that additional contextual information must be provided in order for operators to correctly diagnose and respond to such rare events. Wickens (1984) argues that users will tailor their polling strategy to reflect the probabilistic behaviour of their system. Less attention will be allocated to the observation of low probability errors. Designers must, therefore, provide sufficient perceptual cues in order to capture the operators' attention. Additional presentation resources might be allocated so that users 1 and 2 are likely to observe $pump\_failed_a$. For instance, designers might exploit both visual and audio alarms. Patterson (1990) argues that warning sounds are essential for the operation of safety-critical processes because hearing is a primary warning sense. These sounds are particularly useful for the presentation of systems whose operators are not continually engaged in controlling application processes. Users may notice audible warnings of periodic failures even if their eyes are closed or they are concentrating on other control tasks. Logic abstractions and PCTL operators might be used to express the duration of such alarms. An audible warning is acknowledged if pump a fails and in all possible futures a warning is presented and an audible warning persists until user 1 acknowledges the alarm:

$$
\begin{aligned}
&audible\_acknowledge\_error \Leftarrow pump\_failed_a \wedge \\
&\quad A[(display\_pump\_warning_a \wedge audible\_alarm\_pump_a)\ \mathcal{U}\ user\_warning\_acknowledge_1] \qquad (18)
\end{aligned}
$$

During the development of complex systems, designers must represent the probability of failure in a range of components. Blow-back valves prevent materials from being forced back into a pump's inlet supply:

$$[fail\_blow\_back_a\ Pr\ 0.00000021] \qquad (19)$$

Tests have shown that the rate of acquisition of warning sounds slows markedly with more than six different alarms (Patterson, 1982). The explicit representation of probabilities using clauses, such as (16), provides designers with a justification for the allocation of alarms. Auditory warnings might be restricted to low probability failures, such as $pump\_failed_a$. From clauses (16) and (19) we know that a *blow_back* failure is more probable than a pump failure. The latter failure might then only be presented using a visual warning:

$$
\begin{aligned}
&acknowledge\_error \Leftarrow blow\_back\_failed_a \wedge \\
&\quad A[display\_blow\_back\_warning_a\ \mathcal{U}\ user\_warning\_acknowledge_1] \qquad (20)
\end{aligned}
$$

To illustrate the importance of combining probabilistic and temporal information during the design of man-machine interfaces we introduce a further clause describing the frequency of a component failure. There is a 0.2×10^{-6} probability of pump pressure escape valve a failing:

$$[escape\_valve\_failed_a\ Pr\ 0.0000002] \qquad (21)$$

We can see that the probability of a $escape\_valve\_failed_a$ is less than that of $pump\_failed_a$. It might reasonably be argued that operators should also be provided with audio warnings to alert them to this low probability failure:

$$
\begin{aligned}
&audible\_acknowledge\_valve\_error \Leftarrow escape\_valve\_failed_a \wedge \\
&\quad A[(display\_valve\_warning_a \wedge audible\_alarm\_valve_a)\ \mathcal{U}\ user\_warning\_acknowledge_1] \qquad (22)
\end{aligned}
$$

If valve and pump failures occur at the same time then both audio alarms might be issued until $user\_warning\_acknowledge_1$. Such solutions can impose considerable demands upon operators who must disambiguate multiple concurrent warnings. These problems can be avoided by altering the rhythm and spectral composition of different alarms. Alternatively, auditory warnings may be ranked using the products of probabilistic risk assessments. It is important to emphasise that other factors must be considered, such as the potential threat of each concurrent failure. PCTL does, however, provide a means of explicitly representing the comparative likelihood of system behaviours in a manner which can be used to inform design. In our example, risk assessments might be used to argue that if $pump_a$ and $escape\_valve_a$ fail in the same interval then operators should be informed of the comparatively less likely $escape\_valve\_failed_a$ than the more frequent $pump\_failed_a$:

$$\begin{aligned} & priority\_valve\_error \Leftarrow \\ & \quad escape\_valve\_failed_a \wedge pump\_failed_a \wedge \\ & \quad A[(display\_valve\_warning_a \wedge display\_pump\_warning_a \wedge \\ & \quad audible\_alarm\_valve_a) \; \mathcal{U} \; user\_warning\_acknowledge_1] \end{aligned} \tag{23}$$

De Keyser (1990) argues that anticipation is vital for decision making in complex environments. Umbers (1979) suggests that a users' control strategy is determined by the predicted behaviour of the system in response to their actions. Bainbridge (1981) argues that operators rely upon predictions about the behaviour of the system when responding to unexpected or novel situations. In other words, the task of controlling a complex system forces users to make their own informal analysis of the stochastic behaviour of application processes. Probabilistic risk assessments might, therefore, be used when developing training material and system documentation. Operators might be required to estimate the probability of failures during competence examinations.

There are a number of limitations with this approach. Firstly, it seems unlikely that the degree of precision represented in clauses such as (21) is required when operating many control systems. This objection might be overcome by asking questions about the relative probability of component failures. For instance, is $pump\_failed_a$ more likely than $escape\_valve\_failed_a$? A second problem is that it may not always be possible for operators to verbalise their knowledge of application behaviour. Broadbent (1990) argues that there is a "negative correlation between the ability to control a system and the score obtained on questions about it". Thirdly, presenting the relative likelihood of system failures in training material may not provide an efficient means of instructing operators about the probable behaviour of a complex system. Simulations might be used to overcome this limitation. Prototypes can be developed to mimic the probable behaviour of application processes. Operators might then be shown partial implementations in order to gain experience of controlling stochastic application processes during the early stages of interface development when full implementations are, typically, unavailable.

4 Simulation

Using PCTL to specify high and low probability behaviours does not guarantee that operators will be able to detect and respond to particular displays, such as $display\_blow\_back\_warning_a$ and $display\_pump\_warning_a$. The Risklog prototyping tool has been developed to provide empirical support for the assumptions that may be embodied within PCTL specifications.

4.1 Risklog And Time

Several research groups have provided executable semantics for temporal logics. For instance, Moszkowski (1986) has developed an interpreter for the Tempura programming language using Lisp. Others have implemented systems using the PROLOG programming environment. This latter approach has been adopted by the developers of Tokio linear temporal logic system (Fujita, Kono, Tanaka and Moto-Aka, 1986). In order to exploit Tokio as a means of deriving prototypes from probabilistic specifications it is necessary to translate PCTL requirements into linear time temporal logic. This is trivial for PCTL clauses which apply over all traces of interaction. Intuitively, if a property holds over all possible future traces then it must hold over that single trace which describes the actual course of interaction. This transformation is illustrated by the following Risklog clause that implements *priority_valve_error* (23):

$$\begin{aligned}&risklog\_priority\_valve\_error : -escape\_valve\_failed_a, pump\_failed_a,\\ &\quad(display\_valve\_warning_a, display\_pump\_warning_a, audible\_alarm\_valve_a)\ \mathcal{U}\\ &\quad\ user\_warning\_acknowledge_1\end{aligned} \tag{24}$$

This approach cannot be applied to clauses, such as *continue_display* (14), which only hold for some traces of interaction. These include the E prefix and do little to direct the execution of a prototype; they represent requirements which may or may not be fulfilled depending upon whether a particular future branch of time is or is not followed. In current implementations of Risklog, designers must either omit such clauses from an executable specification or strengthen them to apply over all future traces of interaction. Our intuition is that this process forms a necessary part of the refinement towards implementation. Future work intends to explore means of retaining these clauses so that Risklog will record whether or not they are satisfied during particular traces of interaction with a prototype.

4.2 Risklog And Probability

A limitation with Tokio as a prototyping tool is that it is extremely deterministic. With clause (24), if a developer asserted that $pump_a$ failed during the first run of a prototype then it would also fail for every other run. Designers must change the facts that hold for many different intervals of interaction in order to ensure that operators are faced with traces that accurately model the probable behaviour of a safety-critical system. Risklog exploits Monte Carlo techniques to avoid such limitations. The term 'Monte Carlo' refers to the use of random numbers to determine whether or not a fact holds during a particular interval of interaction. By analogy, if Risklog is faced with a choice of possible behaviours then it makes its decision by flipping a coin. The use of probabilities is analogous to weighting the coin in favour of a particular outcome. In our example, the probability of $pump_a$ failing, 0.241×10^{-5}, is compared with a random number in the range [0.0, 1.0]. If the random number is less than or equal to the probability then the fact is assumed to be true; the pump fails. If the random number is greater than the probability then the fact is assumed to be false; the pump does not fail. The calculation of a probability and the comparison of a random number are represented using propositions with a *simulate* prefix:

$$\begin{aligned}&risklog\_simulate\_cooperation : -simulate\_pump\_failed_a, display\_pump\_warning_a,\\ &\quad(\neg stand\_by\_pump\_on\ \mathcal{U}\ user\_start\_stand\_by\_pump_a, user\_start\_stand\_by\_pump_2)\end{aligned} \tag{25}$$

If a probability is increased then there is a greater chance that it will be larger than the random number generated during the evaluation of the *simulate* proposition. It is also important to simulate low probability traces, such as pump and blow-back failures. Risklog achieves this by reversing the previously mentioned decision procedure during the comparison of probabilities and random numbers. If the random number is greater than the probability then the corresponding fact is assumed to be true. If the random number is less than or equal to the probability then the fact is assumed to be false.

Figure 3 illustrates the architecture of Risklog. Tokio is used to interpret clauses containing temporal logic operators. The Monte Carlo suite calls the random number generators used by *simulate* clauses. The linear congruent method is used and the algorithm is seeded using the time at which PROLOG first consults the Risklog files (Sedgewick, 1988). These facilities are implemented using an interface between

PROLOG and the C programming language. A companion paper describes how Risklog exploits the Prelog system to implement graphical interfaces for stochastic, multi-user applications (Johnson, 1993).

5 Conclusion And Further Work

This paper has argued that risk assessment techniques can be recruited to support the development of interactive, safety-critical systems. Existing approaches, such as fault-trees, cannot adequately capture the temporal properties that characterise interaction with stochastic application processes. Temporal logics, such as Manna and Pnueli's linear notation, can be used to explicitly represent sequential and concurrent properties of interactive dialogues. A probability function, Pr, can be introduced to represent stochastic behaviour in a similar manner to that exploited in the probability logic described by Rescher (1969). The semantics of the resulting notation are unclear. In particular, it is not possible to differentiate between contingent and frequency views of probability. We have shown how PCTL exploits the branching time model of Clarke and Emerson's CTL to avoid such ambiguity (Clarke and Emerson, 1982). This logic notation can be used to represent probabilistic and temporal properties of man-machine interaction. Its semantics are based upon the frequency approach to probability. Finally, we have argued that executable subsets of PCTL support the development of interactive simulations. The Risklog prototyping tool exploits Monte Carlo techniques to generate low and high risk system behaviours. These simulations can be shown to potential users and are amenable to experimental analysis.

There is an important limitation with Risklog's decision procedure; low probability behaviours are unlikely to occur. It is, therefore, difficult to simulate the mixture of high and low probability failures which often frustrate the use of complex computer systems. Current work is exploring tolerances, or bands, within which low probability propositions will occasionally be evaluated as true irrespective of the random numbers generated by Monte Carlo techniques.

Low probability, high cost failures, typically, have a greater impact upon safety than high probability, low cost errors. Further work intends to enhance Risklog to consider the costs of failures when simulating situations that require operator intervention. Utility functions will be introduced into PCTL specifications (Burns, 1991). We also intend to apply the techniques, described in this paper, to a range of application domains. The intention is to determine whether PCTL can capture generic temporal and stochastic properties of man-machine systems.

6 Acknowledgements

Thanks are due to Professor John McDermid who provided many useful comments on an early version of this paper. Thanks are also due to Professor Michael Harrison and the members of the Human Computer Interaction Group, Department of Computer Science, University of York.

7 References

R.E. Allen (1990). *The Oxford Concise Dictionary.* The Clarendon Press, Oxford, United Kingdom.

F. Bacchus (1990). *Representing And Reasoning With Probabilistic Knowledge.* The MIT Press, Cambridge, United States of America.

L. Bainbridge (1981). Mathematical equations or processing routines. In J. Rasmussen and W. Rouse, editors, *Human Detection And Diagnosis Of System Failures*, pages 259–286. Plenum Press, New York, United States Of America.

British Standard's Institute (1979). *Glossary Of Terms Used In Quality Assurance*, BS 4778, London, United Kingdom.

D.E. Broadbent (1990). Decisions and verbal justifications. In D.E. Broadbent, J. Reason, and A. Baddeley, editors, *Human Factors In Hazardous Situations*, pages 45–54. Clarendon Press, Oxford, United Kingdom.

A. Burns (1991). The HCI component of dependable real-time systems. *The Software Engineering Journal*, 6(4):168–174, July 1991.

R. Carnap (1962). *The Logical Foundations Of Probability.* The University of Chicago Press, Illinois, United States of America.

E.M. Clarke and E.A. Emerson (1982). Design and synthesis of synchronisation skeletons using branching time temporal logic. In D. Kozen, editor, *Logic of Programs 1981 - Proceedings*, LNCS 131, pages 52–71. Springer-Verlag, Berlin, FDR.

A.P. Cox, editor (1985). *Risk Analysis In The Process Industries: The Report Of The International Study Group On Risk Analysis.* EFCE No. 45. The Institute Of Chemical Engineers, Rugby, United Kingdom.

K.R. Davies (1985). Techniques for the identification and assessment of major accident hazards. In J. Burgoyne, editor, *The Assessment And Control Of Major Hazards*, pages 289–308. Pergamon Press, Oxford, United Kingdom.

M. Fujita, S. Kono, H. Tanaka, and T. Moto-Aka (1986). Tokio: Logic programming based on temporal logic and its compilation to PROLOG. In E. Shapiro, editor, *Third International Conference On Logic Programming*, LNCS 225, pages 695–708. Springer-Verlag, Berlin, FDR.

H. Hansson and B. Jonsson (1989), A Framework For Reasoning About Time And Reliability, In *Proceedings Of The 10th IEEE Real-Time System Symposium*, IEEE.

M.D. Harrison and H.W. Thimbleby, editors (1989). *Formal Methods In Human Computer Interaction.* Cambridge University Press, Cambridge, United Kingdom.

D.L.M. Hunt and P.K. Ramskill (1985). The behaviour of tanks engulfed in fire - the development of a computer program. In J. Burgoyne, editor, *The Assessment And Control Of Major Hazards*, pages 71–86. Pergamon Press and The Institution Of Chemical Engineers, Oxford, United Kingdom.

Institute Of Chemical Engineers (1985). *Nomenclature For Hazard And Risk Assessment In The Process Industries*, Rugby, United Kingdom.

International Atomic Energy Agency and The Commission of the European Community (1984). *Critical Survey of Research On Human Factors And The Man-Machine Interaction*, IAEA-SM-26B/29, Vienna, Austria.

A. Jack (1992). It's hard to explain but Z is much clearer than English. *The Financial Times*, page 22, 12 April.

C.W. Johnson (1991). Applying temporal logic to support the specification and prototyping of concurrent multi-user interfaces. In D. Diaper and N. Hammond, editors, *People And Computers VI: Usability Now*, pages 145–156. Cambridge University Press, Cambridge, United Kingdom.

C.W. Johnson (1993). The implementation of a formal prototyping environment for multi-user systems. In P. Dourish, editor, *Implementation Perspectives on CSCW Design.* Springer Verlag, Berlin, Germany, (forthcoming).

C.W. Johnson and M.D. Harrison (1992). Using temporal logic to support the specification and prototyping of interactive control systems. *International Journal Of Man-Machine Studies*, 36:357–385.

K. Katsuyama, F. Sato, T. Nakakawaji, and T. Mizuno (1991). OSI testing environment based on standardized formalisations. In J. Quemada, J. Mañas, and E. Vazquez, editors, *Formal Description Techniques III*, pages 287–294, Elsevier, North Holland.

V. De Keyser (1990). Temporal decision making in complex environments. In D.E. Broadbent, J. Reason, and A. Baddeley, editors, *Human Factors In Hazardous Situations*, pages 121–128. Clarendon Press, Oxford, United Kingdom.

L. Lamport (1982). TIMESETS - a new method for temporal reasoning about programs. In D. Kozen, editor, *Logic Of Programs 1981 - Proceedings*, LNCS 131, pages 177–196. Springer-Verlag, Berlin, FDR.

C.H.C. Leung and K. Wolfenden (1983). Disk database efficiency: A scheme for detailed assessment based on semi-Markov models. *Computer Journal*, 26(1):10–14, February.

G. Mancini (1988). Modelling humans and machines. In L.P. Goodstein, H.B. Anderson, and S.E. Olsen, editors, *Task, Errors and Mental Models*, pages 278 – 292. Taylor and Francis, London, United Kingdom.

Z. Manna and A. Pnueli (1981). 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.

Ministry Of Defence (1991). *Requirements for the Procurement of Safety Critical Software*, MOD DEF-STAN 0055, London, United Kingdom.

B. Moszkowski (1986). *Executing Temporal Logic Programs.* Cambridge University Press, Cambridge, United Kingdom.

National Aeronautic and Space Administration (1989). *Advanced Orbiting Systems - Architectural Specification For The CCSDS Secretariat*, Washington DC, United States of America.

D.A. Norman (1990). The 'problem' with automation : Inappropriate feedback and interaction not 'over-automation'. In D.E. Broadbent, J. Reason, and A. Baddeley, editors, *Human Factors In Hazardous Situations*, pages 137–145. Clarendon Press, Oxford, United Kingdom.

R.D. Patterson (1982). Guidelines for auditory warning systems on civil aircraft. Technical Report 82017, United Kingdom Civil Aviation Authority, London, United Kingdom.

R.D. Patterson (1990). Auditory warning sounds in the work environment. In D.E. Broadbent, J. Reason, and A. Baddeley, editors, *Human Factors In Hazardous Situations*, pages 37–44. Clarendon Press, Oxford, United Kingdom.

President's Task Force On Aircraft Crew Compliment (1981), United States' Government. *Report On Aircraft Crew Compliment*, Washington DC, United States of America.

J. Rasmussen (1985). Trends in reliability analysis. *Ergonomics*, 28(8):1185–1195.

J. Rasmussen (1988). Cognitive engineering, a new profession? In L.P. Goodstein, H.B. Anderson, and S.E. Olsen, editors, *Task, Errors and Mental Models*, pages 325 – 334. Taylor and Francis, London, United Kingdom.

J. Reason (1990). *Human Error.* Cambridge University Press, Cambridge, United Kingdom.

N. Rescher (1969). *Many-Valued Logic.* Mc Graw-Hill, New York, United States of America.

C.J. Van Rijsberg (1992). Probabilistic retrieval revisited.
The Computer Journal, 35(5):291–298.

G. Saake and U.W. Lipeck (1988). Using finite-linear temporal logic for specifying database dynamics. In E. Börger and H. Kleine Büning and M.M. Richter, editors, *Workshop On Computer Science Logic - 1988 Proceedings*, LNCS 385, pages 288–300, Springer-Verlag, Berlin, FDR.

R. Sedgewick (1988). *Algorithms.* Addison Wesley, Wokingham, United Kingdom.

C.E. Shannon (1938). A symbolic analysis of relay and switching circuits. *Transactions Of The American Institute Of Electrical Engineers*, 37:713–723.

B. Sufrin and J. He (1990). 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.

R. Took (1991). Integrating inheritance and composition in an objective presentation model for multiple media. In F.H. Post and W. Barth, editors, *EUROGRAPHICS '91*, pages 291–303. Elsevier Science Publications, North Holland, Netherlands.

I.G. Umbers (1979). Models of the process operator. *The International Journal of Man-Machine Studies*, 11(2):263–284.

W.E. Vesely (1981). The fault tree handbook. Technical Report USNRC NUREG 0492, United States' Nuclear Regulatory Commission, Washington DC, United States of America, January.

I.A. Watson (1985). Review of human factors in reliability and risk assessment. In J. Burgoyne, editor, *The Assessment And Control Of Major Hazards*, pages 323–337. Pergamon Press, Oxford, United Kingdom.

C.D. Wickens (1984). *Engineering Psychology And Human Performance.* C.E. Merrill Publishing Company, London, United Kingdom.

E.L. Wiener (1988). Cockpit automation. In E.L. Wiener and D.C. Nagel, editors, *Human Factors In Aviation*, pages 433–461. Academic Press, London, United Kingdom.

N. Worley and J. Lewins, editors (1988). *The Chernobyl Accident And Its Implications For The United Kingdom - Report Number 19 Of The Watt Committee on Energy.* Elsevier Applied Science, London, United Kingdom.

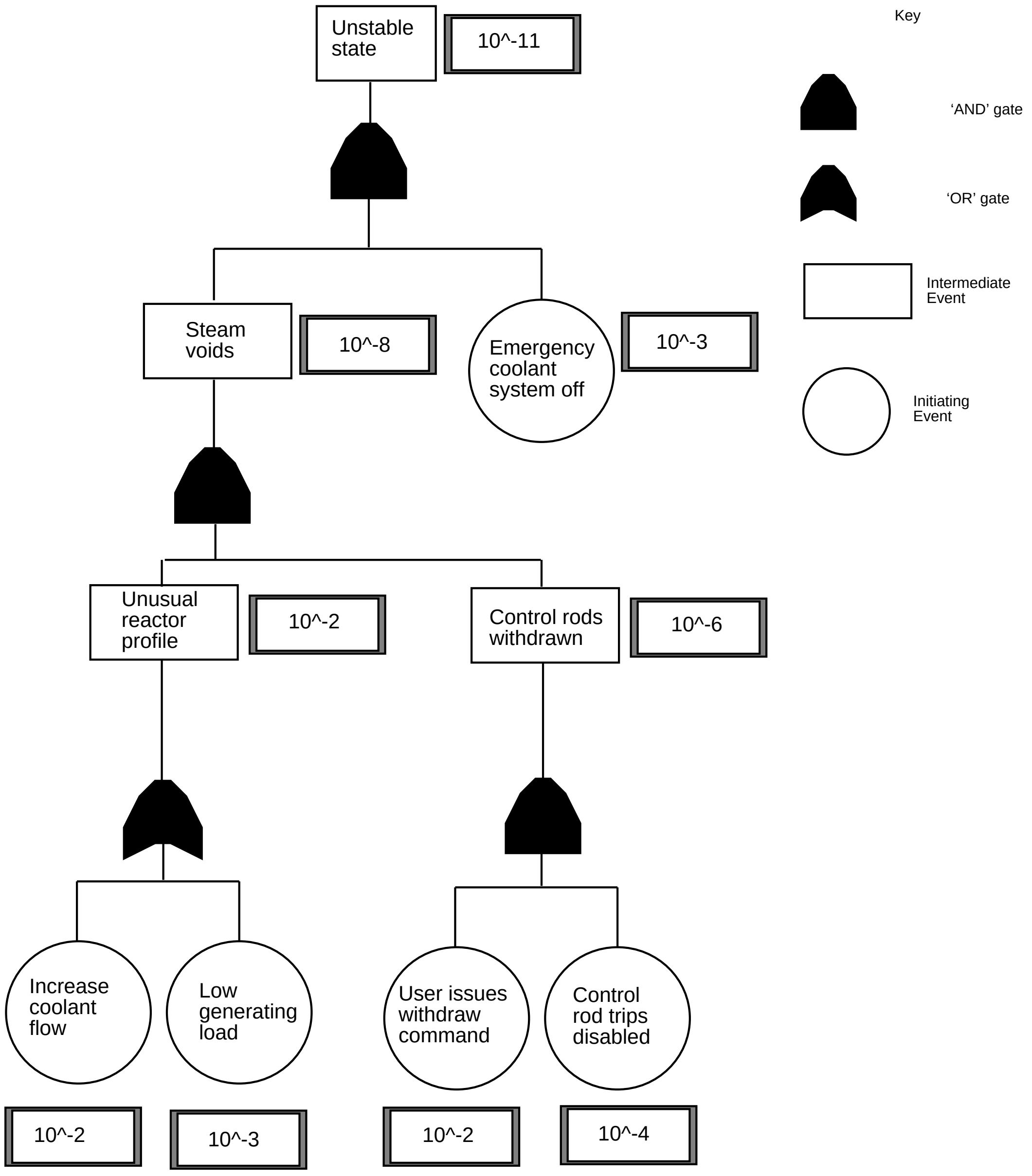

Figure 2: Fault-tree Representing Quantified Probabilities

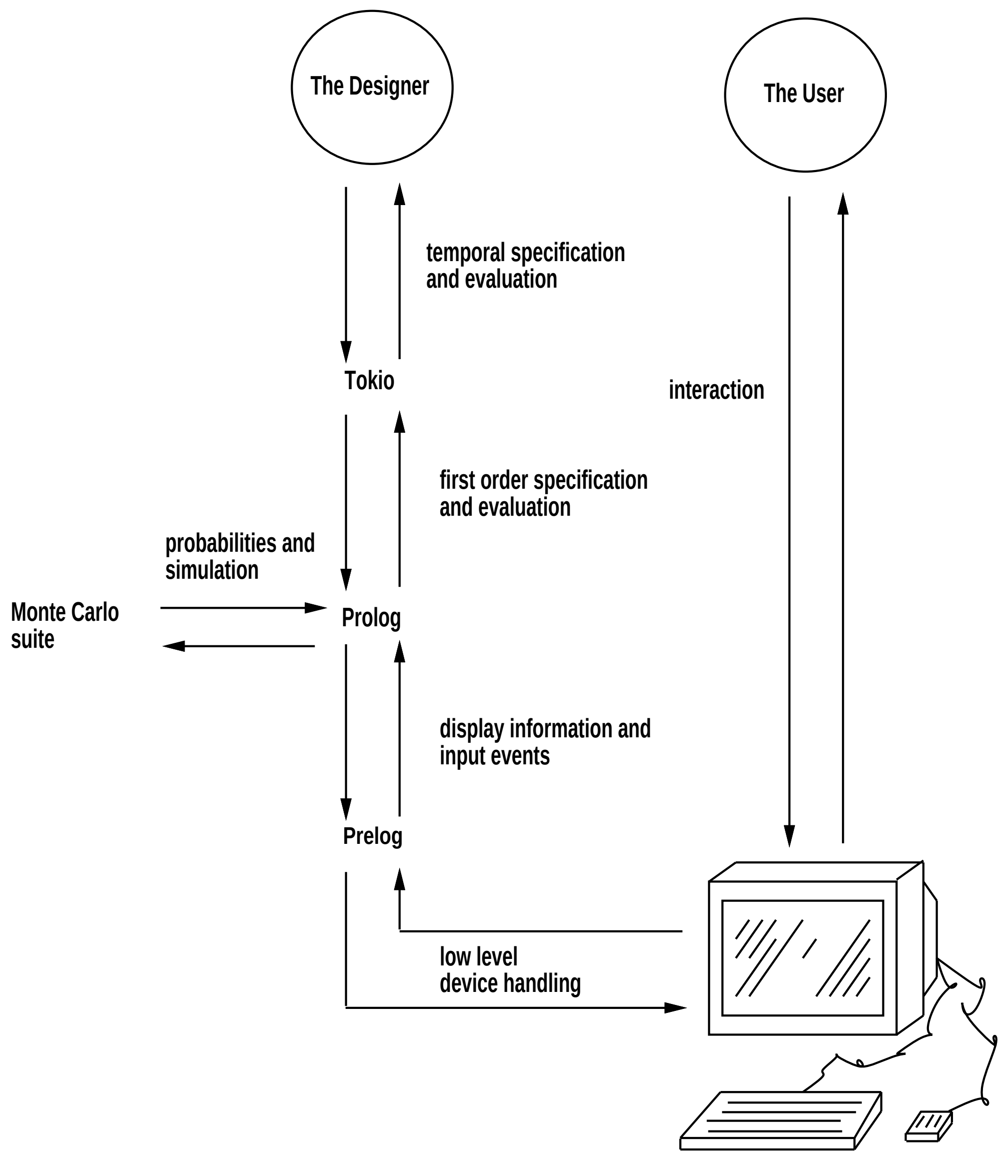
The Designer
The User
temporal specification and evaluation
Tokio
interaction
first order specification and evaluation
probabilities and simulation
Monte Carlo suite
Prolog
display information and input events
Prelog
low level device handling

Figure 3: The Risklog Architecture