The Role Of Error-Tolerant Design In Minimising The Impact Of Risk

C.W. Johnson,
Department of Computing Science, University of Glasgow,
Glasgow, United Kingdom, G12 8QQ.
E-mail: johnson@dcs.glasgow.ac.uk,
Telephone: +44 41 339 8855, Fax: +44 41 339 4913.

ABSTRACT

The introduction of safety devices does not always lead to decreases in the frequency of accidents. Operators adjust their response to technological improvements. Protection mechanisms may be used to support 'unsafe' working practices so that the net risk of failure remains unchanged or may even rise. Regulatory bodies have reacted against this risk compensation. Rules and procedures have been imposed upon the day to day operation of protection equipment. Unfortunately, these constraints often fail to preserve the safety of an application. Inattention, fatigue, poor training and willful neglect can lead to rules being broken and guidelines being ignored. In this paper we explore alternative means of reducing the threats posed by risk compensation behaviour. In particular, it is argued that designers must exploit an integrated approach to the development of high risk applications. By this we mean that both human factors and system engineering must be recruited to support the development of protection equipment.

1 Introduction

The introduction of safety features into working environments has not necessarily seen any reduction in the number of accidents. The availability of protection devices enables people to alter or relax their behaviour so that the net likelihood of an accident actually rises [38]. There are numerous examples of this risk compensation. For instance, the United States' National Transportation Safety Board acknowledges that whilst additional safety devices have reduced

the number of mechanical failures in jet accidents, the proportion of human errors has increased [32]. Similarly, improvements in the road-handling features of cars have done little to reduce the number of accidents involving those vehicles [11]. From a systems engineering point of view this raises a number of important issues. Costly risk assessment techniques may be of little value if improvements in systems safety are outweighed by the risks introduced by changes in operator behaviour. This problem might be avoided if designers remove safety features. The consequences of high risk operating techniques would then not be mitigated by the intervention of safety devices. This approach seems to be ethically unacceptable. Alternatively, protection mechanisms might be hidden from system operators until they are needed. McKenna terms this 'psychological invisibility' [22]. The problem with this approach is that once safety devices have been activated, it becomes difficult to hide their presence from system operators. Given that safety-devices are being developed what measures can we adopt to avoid the dangers implied by risk compensation?

1.1 Market Regulation And Legislation

Regulatory and legislative sanctions provide means of ensuring that operators do not compromise the safety of their protection systems. The Three Mile Island emergency provides an example of this approach to risk compensation. Following an earlier incident on the site, warning lights were installed to indicate that pressure operated relief valves had been activated to protect the reactor from a loss of coolant. The operators assumed that these lights indicated that the valves were actually shut. They actually indicated that an instruction had been issued to close the valve. When the operators did intervene, their actions actually exacerbated the situation [29]. Following the Presidential investigation into the Three Mile Island accident, the United States' Nuclear Regulatory Commission (NRC) reacted to the problems of risk compensation by adopting the principle of minimal intervention [27]. The safety of the system should not be dependent upon operator responses to system warnings. This rule was embodied in the terms and conditions of operation that were imposed upon plant management. It was stipulated that automatic systems must run high-head charging pumps, part of the emergency cooling equipment, for at least twenty minutes after reactor scrams. A scram occurs when neutron absorbers are inserted into a reactor in order to slow the reaction process. It was assumed that such legislation would ensure minimal operator intervention. The dangers of risk compensation were to be avoided by preventing users from interfering with the protection systems. The 1979 North Anna incident illustrates some of the problems which can occur when regulations govern

the operation of complex systems. Changes in the generation process employed by the North Anna reactor led to dangerous temperature profiles following a scram. The operators were faced with a difficult choice. If they obeyed NRC regulations then the safety of the plant would be threatened. If they disobeyed the regulations then the plant could be saved but they would break the NRC conditions of operation. Fortunately, plant management chose to disregard the principle of minimal intervention. A pump was taken off the coolant circuit and the emergency was resolved. Duncan observes that this incident underlines the dangers of trying "to prescribe regulations, procedures or algorithms, especially when these prescriptions are backed by legal sanctions" [9].

1.2 Human Factors Approaches

The guidelines embodied in plant regulations can be reinforced by operator training. This provides an additional means of countering the threats posed by risk compensation. In U.K. nuclear power generation, all Operating and Maintenance enginners are trained to understand the justification for Station Operating Instructions, Plant Item Operating Instructions and Plant Maintenance Instructions [40]. The training is intended to ensure that protection devices are not relied upon to support dangerous operating procedures. Training does not guarantee the safety of an application. This is illustrated by the events leading to the Chernobyl accident. Plant management initiated a test that was intended to determine whether the reactor could continue to supply power once it had begun to be run down. If successful, the pumps which supplied emergency coolant to the reactor could be powered by that reactor until diesel generators were started. Similar tests had failed on two previous occasions because the power level had fallen too rapidly. Operator intervention provided the immediate causes of the accident: they disabled the emergency core cooling systems and made several errors in the way that they conducted the tests [30]. In particular, they relied upon control rods to shut down the reactor when it became unstable. The design of RBMK rod assemblies created the potential for positive void coefficients. Once control rods had been withdrawn beyond a certain point, their reintroduction increased rather than decreased reactivity. The Watt committee report into the Chernobyl accident found that:

> The operators should not have continued to operate the reactor under these conditions; however, there was no automatic system to prevent them." [40].

As this quotation suggests the design and provision of system inter-locks might have preserved the safety of the application when established training and operating procedures were ignored.

1.3 Systems Engineering

Protection devices can prevent operators from exacerbating systems failures. It is difficult, however, to guarantee that such devices will preserve safety. This point is illustrated by attempts to guard against chemical spillages. For example, an operator was required to fill a chemical tank with sufficient raw material to last the day's production [17]. They were supposed to monitor the level of the substance until the tank reached 90% full. This operation was successfully performed for many years until the operator allowed the tank to overflow. A high level trip was installed to switch off the pump automatically when the level in the tank exceeded 90%. Less than twelve months later the accident recurred. The operator decided to rely on the trip to halt the pumping. Both the manager and the foreman knew this was happening but sanctioned the behaviour because the operator's time could then be re-deployed. Paradoxically the safety features threatened safety because users relied too much on blow-back protection. Operators optimised their finite cognitive and perceptual resources by allocating control tasks to automated systems. Problems arose because this allocation had not been intended by the engineers who had developed the automated equipment. The trip was being used as a process controller and not as a safety feature. They did not realise that trips failed approximately once every two years. In other words, systems engineering had provided additional features without considering the human factors of control.

1.4 Integration

Given the problems associated with regulatory solutions, systems engineering and human factors techniques, it seems appropriate to consider ways in which an integrated approach might reduce the impact of risk compensation. Legislation and regulation fail because detailed edicts can be too inflexible to govern processes where human judgement must preserve the safety of an application. In contrast, training techniques can be too flexible. Users may disregard routines and procedures under the commercial pressures of day to day operation. Alternatively, systems engineering can be used to limit operators' scope for intervention. Interlocks can make it hard for users to interfere with safety devices. The problem with this approach is that the inter-locks which help to guard protection devices also help to create the conditions for risk compensation; operators routinely rely upon these features to support high risk control strategies. This paper

argues that the integration of human factors and systems engineering techniques reduces the problems of risk compensation.

1.5 Outline Of The Paper

This section has introduced the problem of risk compensation. Solutions which rely solely upon legislation, training or systems engineering have been criticised as unreliable and possibly dangerous. An integrated approach to systems development has been advocated. Section 2 argues that many different barriers prevent companies from exploiting an integrated approach to systems development. As a result it can be difficult to identify areas in which operators might compound the risks of systems failure. Section 3 argues that existing risk assessment techniques cannot easily be applied to reduce the problems of risk compensation. Techniques such as fault trees and Markov models provide only limited support for the integration of human factors and systems engineering. Section 4 argues that abstract, mathematical notations can be used to avoid these limitations. It is argued that such formalisms must express temporal and stochastic properties of human-machine interaction if they are to capture the conditions for risk compensation. Section 5 criticises this approach and argues that abstract notations provide a poor impression of what it would be like to interact with an application. The Risklog prototyping tool has been developed to avoid this limitation; it uses Monte Carlo techniques to directly derive simulations from logic specifications of human-machine interfaces. Section 6 presents the conclusions that can be drawn from this work. Areas for future work are identified.

2 The Problems Of Integrated Design

A number of problems frustrate an integrated approach to risk compensation. Systems engineering and human factors issues are often addressed during different stages of the development cycle. This prevents human factors designers from tailoring an interface to discourage abuse of protection devices. Conversely, it can be difficult for systems engineers to introduce additional safety features late in the development cycle when the dangers of risk compensation are identified during interface development.

2.1 The Development Cycle

The techniques traditionally used to develop complex systems frustrate attempts to avoid risk compensation. Interaction issues may only be considered after an application has been delivered.

Designers assess the ways in which operators interact with safety devices after a system has been installed:

> "Experience shows that when autoclaves or other batch reactors are fitted with drain valves, they may be opened at the wrong time and the contents tipped onto the floor, often inside a building. To prevent this, the drain valves on a set of reactors were fitted with interlocks so that they could not be opened when the pressure was above a preset value. Nevertheless, a drain valve was opened when a reactor was up to pressure and a batch emptied onto the floor. The inquiry disclosed that the pressure measuring instruments were not very reliable. So the operators had developed the practice of defeating the interlocks either by altering the indicated pressure with the zero adjustment screw or by isolating the instrument air supply. One day the inevitable happened. Having defeated the interlock, an operator opened a drain valve by mistake instead of a transfer valve." (Kletz, [17]).

Inter-locks were only fitted in response to repeated accidents involving the inadvertent evacuation of pressurised vessels. Human factors problems were identified during the accident investigation. Once the inter-locks had been fitted, designers again failed to predict the ways in which operators would respond to inaccurate sensing devices. Such incidents reveal an accident driven approach to systems development. The problems of risk compensation are only considered once problems have occurred. In retrospect, as Kletz observes, the failure seems to be 'inevitable'. In order to combat risk compensation, the development cycle must enable designers to identify and remove the conditions which create the potential for failure.

The iterative refinement advocated by Carroll and Kellog's task-artifact cycle provides a framework for the integration of human factors and systems engineering [3]. This cycle can be decomposed into the three phases shown as boxes in Figure B. Systems engineers implement or modify a control system. The introduction and use of this system creates new tasks which can be identified by human factors engineering. Observations of use can be interpreted to assess the strengths and weaknesses of new safety systems [2]. This process of interpretation is termed hermeneutics; designers must identify and justify the human factors claims that are embodied within the systems. For instance, human factors teams might be called upon to assess the claim that a protection device does not introduce the dangers of risk compensation. Design progresses as the veracity of these claims is confirmed or denied. Lewis' agenda for research in human computer interaction cites hermeneutics as one of the five most powerful techniques currently

available to human factors practitioners [20].

It is difficult to see how Carroll and Kelogg's abstract notion of development can be mapped down onto the detailed pragmatics of systems design. For instance, many applications are open to simultaneous input from more than one source. This openness impairs the successful application of hermeneutics because it can be extremely difficult for designers to validate claims which are made about an operator's interaction with external processes. The Flixborough Nypro disaster illustrates some of the problems which can arise when applying hermeneutic to avoid risk compensation. The accident occurred because nitrate discharges contaminated coolant supplied to a process by another production-line [17]. Users relied upon safe-guards in the supply process to protect their own application. The system embodied the claim that operators did not need to guard against corrosion cracking in their process components. The veracity of this assumption was not explicitly considered until after the Flixborough accident had occurred. In other words, the task artifact cycle suffers from the more general problems associated with the traditional development cycle. Risk compensation occurs because human factors and systems engineering are not completely integrated. Human factors engineers are only used to assess a system once it is in operation. At this point the artifact is already in use and may already be endangering lives.

Figure B presents a more detailed model of development than that proposed by Carroll and Kellogg [39]. Initial design is intended to elicit and record the requirements which are to be imposed upon a human-machine system. Formative evaluation validates those requirements. Summative evaluation ensures that an implementation satisfies its requirements. This model avoids some of the criticisms made about hermeneutics; systems engineering is woven into the development of the interface during stages two and three. The interpretation of an artifact is not left until after the system has been installed; prototyping and acceptance testing are conducted before implementation.

2.2 Communication

The integrated model presented in Figure B does not provide a panacea for risk compensation. In particular, it does not suggest ways in which the findings of human factors or systems engineers can be communicated to their partners in the development process. Natural language might be used. The limitations of this approach have been identified in a companion paper [16]. Problems can arise from terms, such as workload, that have a precise technical meaning in one discipline but which also have a more general interpretation. Vernacular descriptions of complex,

multi-user, multi-process interaction quickly become intractable. It can be difficult to represent concurrency requirements. For example, if an aircraft engine is damaged then the crew must rotate its propeller blades in line with the direction of flight. Accident analysis has revealed that confusion amongst the crew will often result in the wrong set of blades being rotated. As a result auto-feathering systems have been developed to automatically perform this rotation. The claims which guided the development of many auto-feathering systems did not adequately account for the group dynamics which govern the operation of these applications. In 1979 a Swift Aire Lines Nord 262 crashed because the automated equipment correctly auto-feathered the right propeller while the operator shut down the left [37]. This is an example of risk compensation. The Captain relied upon the protection systems and did not adequately confirm the target of their commands. These problems can be addressed by voting protocols that require a coordinated response from other members of the crew before important instructions are issued. Specifying the order of such dialogues and the circumstances in which they may be initiated requires many dozen of pages of textual description [16]. It can be difficult for human factors and systems engineers to identify potential usability problems from such documentation. Designers must be provided with means of stripping out low level details in order to focus upon issues such as risk compensation.

Communication problems not only arise from the complexity of interaction with control applications. They also occur because design decisions must be propagated through the various stages of human factors and systems development. It is particularly difficult to ensure that requirements are fed from an initial design into the formative evaluation of safety-critical system. The problems of risk compensation might be forgotten amongst the many other issues that must be considered during these initial stages of development. It is a non-trivial task for designers to determine whether a partial implementation satisfies the mass of requirements which emerge during the subsequent design of multi-user applications. For example, the symptoms of risk compensation may only emerge after many hours of operating an application [36]. Tools which enable designers to reduce the problems of risk compensation must, therefore, support communication between human factors and systems design teams during the various stages of the development cycle.

3 Existing Tools

A number of existing design notations might be recruited to support communication during the various stages of human factors and systems engineering. These formalisms could be used to represent and reason about the problems of risk compensation before accidents threaten safe and successful production.

3.1 Fault Trees

Fault-trees are widely used by systems engineers working in many different domains [34]. The notation shown in Figure B conforms to guidelines that have been drawn up by the European Federation of Chemical Engineers [7]. The bottom-left branch of this fault-tree represents the risk that there will be a water ingress into the plant if there are wet weather conditions and there is a fault in the air exchanger [8]. Nodes in the tree are annotated with probabilities. The method of propagation is derived from Watson [35]. Figure B shows how fault trees might be used to represent the way in which risk homeostasis might jeopardise the utility of safety devices. The bottom left branch quantifies the risk that a new reserve exchanger fails. This reduces the likely hood of a water ingress from 0.001 per year to 0.00001 per year. This reduction in risk is offset by the corresponding risk in the likelihood that an operator will not detect the fault. Increased reliance on the additional protection system increases this risk from 0.1 per year to 10 per year. The overall risk of a chlorine reaction rises in consequence from 0.00005/year to 0.0005/year. Such graphical representations have an intuitive appeal. They can be shown to members of concurrent design teams in order to validate the assumptions that are made about operator responses to new safety devices. For instance, human factors designers might challenge the probability associated with operators failing to detect the fault.

There are a number of limitations which restrict the application of fault trees to support software development. Hollywell, in this volume, criticises the independence assumptions that lie behind fault trees. The International Study Group On Risk Analysis also 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" [7].

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 for software

development. Different sequences of events have different costs for control systems. The consequences might be very different if an operator noticed the equipment failure before or after water ingressed into the plant in our example. Event sequences can also determine the quality of interaction between an operator and their system. Identifying the order in which commands are issued and warnings are displayed forms an important stage in the development of man-machine interfaces.

3.2 Markov Models

Markov models provide a means of representing the ways in which events interact [19]. Figure B 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 that a pump is working to its failure and repair rates at time t [8]. This formula can be applied to deduce that if a carbon tetrachloride 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:

$$probability(pump_working)_t = \mid \frac{1}{r_fail + r_repair} r_repair + r_fail\, e^{-(r_fail + r_repair)} \mid_t \quad (1)$$

This approach might be used to represent and reason about risk compensation. For instance, Figure B models the introduction of a stand-by pump into the system. The labels `r_fail_working_A` refer to the rate of failure for pump A from a working state. The labels `r_fail_stand_by_B` refer to the rate of failure for pump B in a stand-by state. Operator reaction to the introduction of stand-by equipment might be represented by decreases in the repair rate for pump A, `r_repair_A`; there might be less urgency if there is a stand-by pump.

A number of problems restrict the application of Markov models to avoid risk compensation. In particular, Figure B and B present highly simplified examples. For any large-scale system the number of potential states will be extremely large. It would be a non-trivial task to transition rates with these states. As the number of states rises in a Markov model there is a corresponding increase in the complexity of the mathematics required to derive failure probabilities. Computer support can be recruited to aid this task but it can be difficult to validate the data obtained from such models. A further limitation for interactive control systems is that operator responses to system failures are represented implicitly in the transitions between system states. Users must intervene to transform a pump from a failed state to a working state. It is not clear how other forms of operator intervention might be represented. For instance, should additional states be

introduced to represent the situation that might arise if operators failed to observe a warning because they relied upon a protection system? Explicitly representing operators as components of a Markov model begs the question: what states can a user be in? Hidden Markov models provide an important area for further research in this area. Designers can use this approach to construct partial models to represent potential traces of interaction [4]. In anticipation of this research, alternative approaches must be investigated [28].

3.3 Failure Mode, Effects And Criticality Analysis

Table 1 is a product of failure mode, effects and criticality analysis. It represents the requirement that more inspections must be scheduled in order to avoid the dried pump seals that can lead to over-heating. Maintenance activities must be improved in order to avoid blow-back valve blockage. An important advantage of failure mode, effects and criticality analysis is that

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

the resulting tables are accessible to non-specialists. These representations could, therefore, be recruited to record and justify decisions during human factors and systems development. Such tables might be used to represent and reason about the problems of risk compensation. For instance, automated testing equipment might be developed to warn operators that maintenance is necessary on blow-back valves. If designers rely upon users noticing and responding to the warning then valve collapses might be more likely; inspections would no longer be a scheduled part of routine maintenance. Table 2 illustrates this problem and also shows how these represen-

Component	Failure Mode	Failure Cause	Criticality	Improvement
Pump Blow-back	Blockage	Valve collapse	Medium	Testing device
Operator	No blow-back maintenance	Fatigue	Medium	New shift allocation

Table 2: Table Showing Solution To Risk Compensation

tations can be used to document potential solutions. If fatigue is assumed to be an important factor in operators missing valve collapse warnings then new shift patterns must be devised.

These would partially offset the gains from reduced maintenance activities.

A limitation of failure mode, effects and criticality tables is that it can be difficult to identify the knock-on effects which faults have upon other components in the system. For instance, the failure of a blow-back valve can have consequences for many different operators in a number of associated processes. The introduction of additional safety features might result in compensatory behaviour by those users. The table might be extended to capture these relationships but this would reduce the tractability of the representation. This is an important limitation for control systems which must counter such chains of behaviour. It is also possible to criticise the indices used to represent utility, or criticality, in these tables. Assessments such as `High`, `Medium` and `Low` are at too coarse a level to differentiate between the costs of failures whose consequences range from minor equipment damage to multiple fatalities. Finally, failure mode, effects and criticality tables do not provide software engineers with means of representing probabilistic information. U.K. legislation on the Control of Industrial Major Accident Hazards includes a requirement for quantified risk assessments [13].

4 Formal Methods

Mathematical specification techniques are increasingly being used to support the engineering of complex, high-risk applications. Formal development techniques have been recommended for the design of command and control systems commissioned by the U.K. Ministry of Defence [23]. NASA are using mathematical specifications to assess the design of their Advanced Orbiting System [25]. These techniques have not been used to address human factors problems, such as risk compensation. This threatens user-centred design. Formal methods are intended to minimise the modifications that are necessary once a specification has been refined towards implementation. Human factors considerations are, therefore, relegated to an after-thought in the development process. Some notable attempts have been made to apply formal methods to support the development of user interfaces. Harrison and Thimbleby are using formal methods to specify high-level requirements for many different interactive systems [12]. Sufrin and He have extended the Z schema calculus to represent interaction with a single-user text-editor [33]. Unfortunately, a number of problems limit the utility of formal notations as means of addressing risk compensation. Although it is possible to identify some usability problems using abstract analytical notations, others cannot be identified until partial and full implementations are available. Glendon, in this volume, emphasises the point that risk homeostasis theories

cannot predict the ways in which operators will compensate for the introduction of safety devices. He also presents results which suggest that evidence of risk homeostasis may depend upon the fidelity of a simulation. Such conflicts make it particularly important that designers have some means of assessing the impact which abstract, formal requirements will have upon the implementation of safety-critical systems.

4.1 CTL And Time

First order logic provides a common language for the specification and prototyping of interactive systems [14]. There is a close correspondence between specifications written in first order logic and programs which satisfy those specifications implemented in Prolog [5]. This provides systems engineers with a means of constructing designs in terms of high level abstractions rather than low level device primitives [15] Human factors engineers can then derive Prolog prototypes from the logic specifications in order to assess the potential for risk compensation. For instance, systems engineering might introduce a stand-by pump as shown in Figure B. The following clause represents the requirement that a back up pump is used to resolve an error if *pump_1* fails, a warning is presented and the *stand_by* pump is on:

$$back_up_solution \Leftarrow pump_1_failed \wedge$$
$$display_pump_1_warning \wedge stand_by_pump_on \quad (2)$$

A back-up pump is used to resolve an error if *pump_1* fails, a warning is displayed and the stand-by pump is on.

Human factors engineers might identify the potential for risk compensation; additional safety devices enable operators to reduce sampling frequencies for pump warnings. The same logic notation can also be used to specify solutions to such problems. Operators must confirm the observation of pump warnings:

$$warning_received \Leftarrow$$
$$back_up_solution \wedge user_1_warning_acknowledge \quad (3)$$

A warning is received if *pump_1* fails, a warning is displayed, the stand-by pump is started and *user_1* acknowledges the warning.

A number of problems restrict the application of logic to address the problems of risk compen-

sation. In particular, first order logic suffers from similar problems to fault-trees. There is no notion of time in classical logic; formula (2) would be true if *user_1* responded before the error occurred and the warning was presented. This is clearly not what the designer intended. This limitation can be avoided by using formalisms which explicitly represent time, such as Clarke and Emerson's Computation Tree Logic (CTL) [6]. Appendix A briefly introduces the syntax and semantics of CTL. Intuitive readings for the logic will be given throughout the remainder of the paper.

Designers can use CTL's X (read as 'next') operator to represent the sequential information that was missing in formula (2). The A operator can be read as 'in all traces of interaction'. For example, the following clause specifies that under all possible circumstances *user_1* responds in the next state after pump *1* fails:

$$receive_error \Leftarrow$$
$$back_up_solution \wedge AX\ user_1_warning_acknowledge \qquad (4)$$

Information about an error is always received if *pump_1* fails, a warning is displayed and the stand-by pump is started in the present interval and in the immediate next interval of all possible traces of interaction *user_1* acknowledges the warning.

The previous formula shows how CTL can represent assumptions about operator responses to system failures after the introduction of safety devices. Designers can use such clauses to assess the usefulness of such 'solutions' to risk compensation. For example, it seems unlikely that users will always be able to provide an immediate response to pump warnings. Given divided attention and fatigue there may exist future traces in which the response is delayed. Human factors engineers can only specify that there exist certain traces of interaction during which *user_1* can immediately acknowledge the warning. This requirement can be represented using the E (read as 'in some traces of interaction') operator. The following clause specifies that in some traces of interaction *user_1* acknowledges the error in the next state after pump *1* fails:

$$possibly_receive_error \Leftarrow$$
$$back_up_solution \wedge EX\ user_1_warning_acknowledge \quad (5)$$

Information about an error is possibly received if *pump_1* fails, a warning is displayed and the back-up pump is started in the present interval and in the immediate next interval of some traces of interaction *user_1* acknowledges the warning.

The previous clauses illustrate how logic specifications provide a means of constructing a design without forcing commitment to a particular implementation early in the development process. It is possible for designers to devise solutions for risk compensation in terms of high level abstractions rather than low level device primitives. These details can be added gradually as development progresses. For instance, 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:

$$continue_display \Leftarrow$$
$$pump_1_failed \wedge stand_by_pump_on \wedge$$
$$E[display_pump_1_warning\,\mathcal{U}\ user_1_warning_acknowledge] \quad (6)$$

Information about an error continues to be displayed if in some traces of interaction a warning is presented until *user_1* acknowledges the message.

Section 3.3 criticised Failure Modes, Effects and Criticality analysis because it does not quantify the probability of particular system failures. This is a significant omission. Probabilistic risk assessments form an important stage in the human factors and systems engineering of many safety-critical applications. In terms of risk compensation, designers must represent and reason about the probability of *pump_1_failed* in order to determine the likelihood that operators will directly come into contact with protection mechanisms. The stochastic behaviour of application processes also has an impact upon the more general development of interactive systems. If *pump_1_failed* is an extremely rare event then additional presentation resources must be devoted to *display_pump_1_warning* so that the operator is likely to observe the warning. Less presentation resources may be necessary in order to present information about high probability errors which can be anticipated by system operators [36].

4.2 PCTL And Probability

PCTL (Probabilistic Computation Tree Logic) extends CTL in order to represent high and low probability situations which require operator intervention. $[f\ P\ y]$ intuitively means that the probability of f being true in the next state is y where y is a value in the range [0.0,1.0]. Appendix B briefly introduces the syntax and semantics of PCTL. The following clause illustrates the use of this notation. Figures from Associated Octel predict 76 failures per operating year for compressors [8]. If state transitions, occur every second then designers can specify that the probability of pump *1* failing in the next state is 0.964×10^{-6}:

$$[pump_1_failed\ P\ 0.000000964] \tag{7}$$

The probability of pump *1* failing in the next state is 0.964×10^{-6}.

If an interface is being developed for an existing system then designers may already have access to data about the behaviour of application processes. In other circumstances, designers must exploit generic data for system behaviour. 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' work for a more detailed discussion [1]. It is also important to note that probabilities are **not** being linked to user behaviours. A number of authors have sought to develop stochastic models of humans operators [31, 35]. Kennedy's paper in this volume reviews the strengths and weaknesses of this approach as a means of predicting accidents. This approach might be extended to represent the probability of risk compensation within a particular application. There are many problems which must be resolved before such an approach can be recruited to support systems development. Generic models provide a very gross level of detail. Many different factors can affect the probability that an operator will behave in a particular manner: inattention; fatigue; anxiety; age; previous experience.

It is possible to exploit probabilistic information about component failure to support human factors and systems engineering. For example, we know from formula (7) that there is a 0.964×10^{-6} probability of pump 1 failing in the next second. There might also be a probability of one failure every 0.2×10^{5} seconds for the blow-back valves that prevent materials from being forced back into a pump's inlet supply:

$$[blow_back_1_failed\ P\ 0.000002] \tag{8}$$

The probability of *blow_back* valve 1 failing in the next state is 2×10^{-5}.

Both *blow_back* and *pump* failures will be rare events. Considerable perceptual prompts will, therefore, be required to alert users to these failures. The low probability of both events might persuade interface designers to present additional diagnostic information. 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 [26]. 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. The previous clauses also capture the observation that a *blow_back* failure is more probable than a pump failure. Designers might use this information when developing training material and system documentation. Resources may be allocated in proportion to the probability of particular system behaviours. Systems engineers might exploit clauses such as (8) to argue that protection devices should be introduced to guard against blow-back failures. Such problems are more probable than the pump failure represented by (7). If the blow-back valve fails then the system should shut-down the corresponding pump and start up the stand-by pump:

$$blow_back_shut_down \Leftarrow$$
$$blow_back_1_failed \wedge AX(pump_1_off \wedge stand_by_pump_on) \tag{9}$$

A shut down occurs as the result of a blow-back failure if the blow-back valve for pump 1 fails and in all next intervals of interaction pump 1 is off and the stand-by pump is on.

As with pump failures, designers might allocate display resources to warn *user_1* of a blow back failure so that they are alerted to the intervention of the automatic safety features. Human factors engineers can also use probabilistic data to represent and reason about the potential for risk compensation. The low probability of a blow-back failure and the presence of additional safety devices might reduce the frequency with which operators poll such displays. Operators might, therefore, be required to confirm their observation of the blow-back warning:

$$acknowledge_error \Leftarrow blow_back_1_failed \wedge$$
$$AX(pump_1_off \wedge stand_by_pump_on \wedge display_blow_back_1_warning) \wedge$$
$$EF\ user_1_warning_acknowledge \quad (10)$$

This states that an error is acknowledged if blow back valve 1 fails and in all next intervals pump 1 is off and the stand-by pump is on and a warning is displayed and eventually in some trace of interaction user 1 provides input to acknowledge the warning.

Previous clauses can be used to identify further ways in which risk compensation might be considered during interface development. For instance, designers might be faced with a choice between protecting against pump or blow back failures, using the techniques specified in (6) and (9). Initial analysis might suggest that the more probable blow-back failures would demand the allocation of scare development resources. It can be argued, however, that more probable failures would carry the greatest effect in terms of risk compensation. Until these effects could be identified, designers might consider allocating resources towards the less probable pump failure in the anticipation that correct operating practices would safeguard against the more probable failure.

5 Prototyping

The previous paragraph speculated about the potential impact of risk compensation following the introduction of safety devices for high and low probability failures. The use of a formal notation, such as PCTL, does not guarantee that such speculation will be correct. The Risklog prototyping tool has been developed to provide empirical support for the assumptions that may be made during systems development.

5.1 Risklog And Time

Design requirements, expressed in formal notations, provide the non-formalist with little idea of the 'look and feel' of an interactive system. Prototypes provide a far better impression of what it would be like to interact with a final implementation. They can be shown to system operators and are amenable to experimental analysis. They can be used to determine the cognitive, perceptual, physiological and sociological demands that a control system might place upon its users. Several research groups have investigated means of deriving prototypes from temporal

logic specifications [24]. For instance, the Prolog logic programming environment has been used to develop the Tokio meta-interpreter [10]. Tokio can be used to implement *receive_error* (4) using the following clause:

$$\begin{aligned} &\mathit{risklog_receive_error} :- \\ &\quad \mathit{pump_1_failed}, \mathit{display_pump_1_warning}, \\ &\quad \mathit{stand_by_pump_on}, \bigcirc\, \mathit{user_1_warning_acknowledge} \end{aligned} \tag{11}$$

Information about an error is received if *pump_1* fails, a warning is displayed and the stand-by pump is started in the present interval and in the next interval *user_1* acknowledges the warning.

In the previous clause, $\bigcirc$ represents the AX quantification of PCTL which can be read as the next interval in all possible courses of interaction. This is appropriate because Tokio attempts to prove that the quantified proposition does in fact hold for the next interval. If it fails then the entire clause is false, in this case the operator does not acknowledge the error. There are, however, some PCTL clauses which cannot be executed so easily. For instance, *continue_display* (6) only holds for some traces of interaction. Such clauses represent weak requirements which may or may not be fulfilled during a particular trace of interaction. They do little to direct the execution of a prototype. Designers must either omit such clauses from an executable specification or strengthen them to require that they must be satisfied by an implementation. Our intuition is that this process forms a necessary part of the refinement towards implementation.

5.2 Risklog And Probability

In logic prototyping environments, if a designer asserted that *pump_1* failed during the first run of a prototype then it would also fail for every other run. This is a significant limitation for the analysis of risk compensation. Risk homeostasis theory postulates long term effects for the introduction of safety devices. It is important, therefore, that repeated use of a prototype should closely follow the anticipated behaviour of application processes. If many different traces of interaction are to be evaluated than engineers must manually edit hundreds of assertions about their system. Such changes would have to be carefully controlled in order to accurately simulate the probable behaviour of an application. Risklog uses Monte Carlo techniques to avoid these limitations. The prototyping tool uses random numbers to determine which of a number of possible paths a simulation must follow. 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 path or paths. In the previous example, the likelihood of *pump_1* failing, 0.964×10^{-6}, is compared with a random number in the range [0.0, 1.0]:

$$risklog_simulate_receive_error :- \\ simulate(pump_1_failed), display_pump_1_warning, \\ stand_by_pump_on, \bigcirc\, user_1_warning_acknowledge. \quad (12)$$

Information about an error is received if risklog uses Monte Carlo techniques to simulate *pump_1* failing and a warning is displayed and the stand-by pump is started in the present interval and in the next interval *user_1* acknowledges the warning.

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. By increasing a probability there is a greater chance that it will be larger than the random number; the corresponding paths of behaviour will more readily be chosen by Risklog.

5.3 Risklog And Interaction

Logic clauses provide an extremely poor impression of the 'look and feel' of potential implementations. This is a problem for the evaluation of risk compensation. Glendon, in this volume, argues that observation of risk homeostasis may depend upon the fidelity of a simulation. The divide between abstract clauses and high-fidelity prototypes can be bridged by the introduction of presentation details. Structured graphics systems can be used to implement complex images which are represented in terms of their component parts. For example, a *coolant_error* display might consist of an icon representing a *rod_trip* as well as a message to *display_pump_1_warning*:

$$part(coolant_error, rod_trip_icon). \quad (13)$$

$$part(coolant_error, display_pump_1_warning). \quad (14)$$

The first clause states that the *rod_trip_icon* is part of the *coolant_error* display. The second clause states that the *display_pump_1_warning* is part of the *coolant_error* display.

Figure B illustrates the composition of the image in terms of regions: a background region; a text region and a *rod_trip_icon* region. The *rod_trip_icon* region might be further decomposed into sub-regions containing lines, squares or circles. Each region has properties, such as size and position, attributes, such as font and pattern, and a behaviour, such as whether or not it is selectable. Elements of the resulting region structures can be directly incorporated into Risklog simulations:

$$\begin{aligned} &risklog_simulate_receive_error :- \\ &\quad simulate(pump_1_failed), stand_by_pump_on, \\ &\quad display(coolant_error), part(coolant_error, display_pump_1_warning), \\ &\quad \bigcirc\, user_1_select(display_pump_1_warning). \end{aligned} \quad (15)$$

Information about an error is received if risklog uses Monte Carlo techniques to simulate *pump_1* failing and the stand-by pump is started and a coolant error is displayed and a pump 1 warning is part of the coolant error display and in the next interval *user_1* acknowledges the message by selecting the pump 1 warning.

The resulting Risklog architecture is illustrated in Figure B. In many applications it is difficult or impossible to evaluate a potential interface within its final working environment. Future research might build upon McCrobie's work on the integration of prototype control systems into full-scale simulations of working environments [21]. Alternatively, techniques developed during the laboratory evaluations of human factors research might be scaled up to support the more complex analysis of industrial applications.

6 Conclusions And Further Work

This paper has argued that risk compensation is an important problem for the development of interactive control systems. The benefits of improved safeguards are frequently jeopardised by changes in operator behaviour. This problem can only be avoided by the integration of human factors and systems engineering. Human factors engineers must recruit systems engineers in order to identify the exact tolerances of the safeguards that protect operators. Systems engineers must consult human factors engineers in order to predict the ways in which users will compensate for the introduction of safety devices. Existing techniques, such as fault tree analysis and Markov models, do not adequately support this integration. Logic has been proposed as an alternative. The same formalism can be used to specify human factors and systems requirements. Probabilistic extensions can be used to explicitly represent the anticipated frequencies of system failures. This information can be used to direct the allocation of development resources when introducing additional safety features. These features must, in turn, be considered by human factors engineers when identifying the potential for risk compensation.

7 Acknowledgements

Thanks are due to the members of the Human Computer Interaction Group, Department of Computer Science, University of York where most of this paper was written.

A The Syntax and Semantics Of CTL

p denotes an atomic proposition and 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.

Clarke and Emerson introduce a model for the semantics of CTL [6]. $M = (S, F_1, ..., F_k, L)$ 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.

B Syntax and Semantics of PCTL

The syntax of PCTL extends CTL with an additional path formula. $[f\ P\ y]$ 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]. Informally, the probability of a proposition is given by the number of possible futures in which that proposition is true divided by the number of all possible futures. For instance, if *pump_1* fails in two out of ten possible states then the probability of *pump_1* failing in the next state is $2/10$ or 0.2. More formally, we can state that $s_0 \models [f\ P\ y]$ iff:

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

Real-time failure rates are represented in PCTL by linking all state transitions to a clock, $\forall\, i(s_i, s_{i+1}) \in F$. Ladkin has developed similar techniques for grounding Allen's convex interval logic in real-time [18].

References

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

[2] J.M. Carroll and R.L. Campbell. Artifacts as psychological theories: The case of human computer interaction. *Behaviour And Information Technology*, 8(4):247–236, 1989.

[3] J.M. Carroll and W.A. Kellogg. Artifact as theory nexus : Hermeneutics meets theory based design. In K. Bice and C. Lewis, editors, *Proceedings Of The CHI'89 Conference On Human Factors In Computing Systems*, pages 7–14. ACM, New York, United States of America, 1989.

[4] E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. Equations for part-of-speech tagging. In R. Fickes and W. Lehnert, editors, *Proceedings Of The Eleventh National Conference On Artificial Intelligence*, pages 784–789, Massachusetts, United States of America, 1994. MIT Press.

[5] K.L. Clark. The synthesis and verification of logic programs. Technical Report 81/36, Department of Computing, Imperial College Of Science And Technology, London, United Kingdom, 1981.

[6] E.M. Clarke and E.A. Emerson. 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, 1982.

[7] A.P. Cox, editor. *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, 1985.

[8] K.R. Davies. 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, 1985.

[9] K.D. Duncan. Fault diagnosis training for advanced continuous process installations. In J. Rasmussen, K. Duncan, and J. Leplatt, editors, *New Technology And Human Error*, pages 209 – 221. J. Wiley and Sons, New York, United States of America, 1987.

[10] M. Fujita, S. Kono, H. Tanaka, and T. Moto-Aka. 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, 1986.

[11] R. Fuller. A conceptualisation of driving behaviour as threat avoidance. *Ergonomics*, 27:1139–1155, 1984.

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

[13] Health And Safety Executive. *Control of Industrial Major Accident Hazards, Consultative Document And Guidance*, London, United Kingdom, 1983.

[14] C.W. Johnson. Using temporal logic to prototype interactive systems. In G. Cockton D. Diaper, D. Gilmore and B. Shackel, editors, *Human-Computer Interaction—INTERACT'90*, pages 1019–1021. Elsevier Science Publications, North Holland, Netherlands, 1990.

[15] C.W. Johnson and M.D. Harrison. PRELOG - a system for presenting and rendering logic specifications of interactive systems. In C.E. Vandoni and D.A. Duce, editors, *EUROGRAPHICS '90*, pages 469–480. Elsevier Science Publications, North Holland, Netherlands, 1990.

[16] C.W. Johnson, J.C. McCarthy, and P.C. Wright. Using Petri nets to support natural language in accident reports. *To Appear In Ergonomics*, 1994.

[17] T.A. Kletz. *What Went Wrong? Case Histories Of Process Plant Disasters*. Gulf, Houston, United States Of America, 1985.

[18] P. Ladkin. Primitives and units for time specifications. In T. Kehler, S.Rosenschein, R. Filman, and P.F. Patel-Schneider, editors, *Proceedings Of The Fifth Annual Conference On Artificial Intelligence*, pages 354–359. Morgan Kaufman, Los Altos, United States of America, 1986.

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

[20] C.H. Lewis. A research agenda for the Nineties in human computer interaction. *Human Computer Interaction*, 5(2-3):125–143, 1990.

[21] D. McCrobie. Human factors design considerations for military trains. In D.L. Pettigrew, editor, *Proceedings Of The 33rd Annual Meeting Of The Human Factors Society*, pages 536–540. Human Factors Society, Santa Monica, United States of America, 1989.

[22] F.P. McKenna. Do safety measures really work? an examination ofg risk homeostasis theory. *Ergonomics*, 28:489–498, 1985.

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

[24] B. Moszkowski. *Executing Temporal Logic Programs*. Cambridge University Press, Cambridge, United Kingdom, 1986.

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

[26] D.A. Norman. 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, 1990.

[27] R.M. Pew, D.C. Miller, and C.E. Feehrer. Evaluating nuclear control room improvements through analysis of critical operator decisions. In R.C. Sugarman, editor, *Proceedings Of The 25th Annual Meeting Of The Human Factors Society*, pages 100–104. Human Factors Society, Santa Monica, United States of America, 1981.

[28] J. Rasmussen. 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, 1988.

[29] J. Reason. Cognitive aids in process environments : Prosthesis or tools? In E. Hollnagel, G. Mancini, and D.D. Woods, editors, *Cognitive Engineering In Complex Dynamic Worlds*, pages 7–14. Academic Press, London, United Kingdom, 1988.

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

[31] W.B. Rouse and S.H. Rouse. Analysis and classification of human error. *I.E.E.E. Transactions On Systems, Man and Cybernetics*, SMC-13(4):539–549, 1983.

[32] R.B. Stone and G.L. Babcock. Airline pilot's perspective. In E.L. Wiener and D.C. Nagel, editors, *Human Factors In Aviation*, pages 529–560. Academic Press, London, United Kingdom, 1988.

[33] 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.

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

[35] I.A. Watson. 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, 1985.

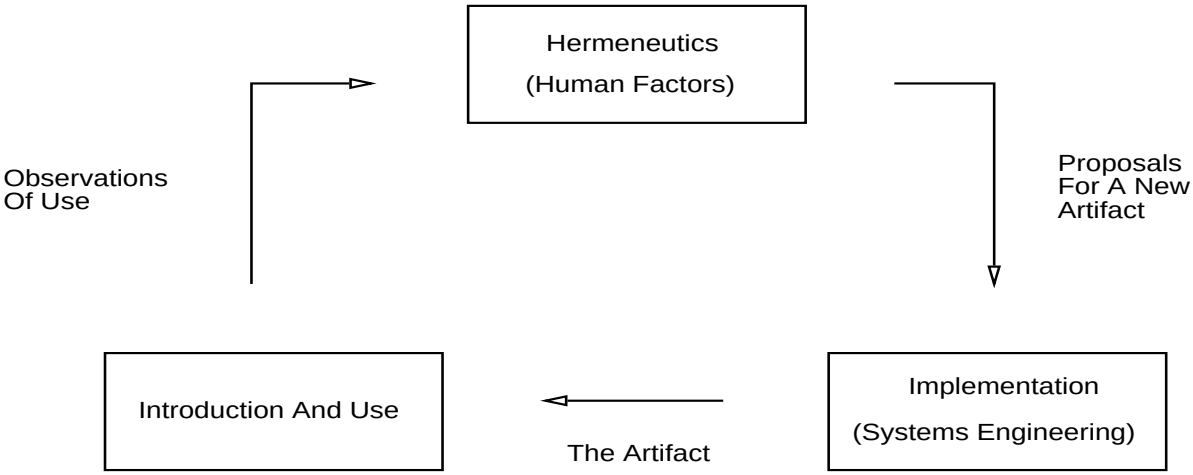
[36] C.D. Wickens. *Engineering Psychology And Human Performance.* C.E. Merrill Publishing Company, London, United Kingdom, 1984.

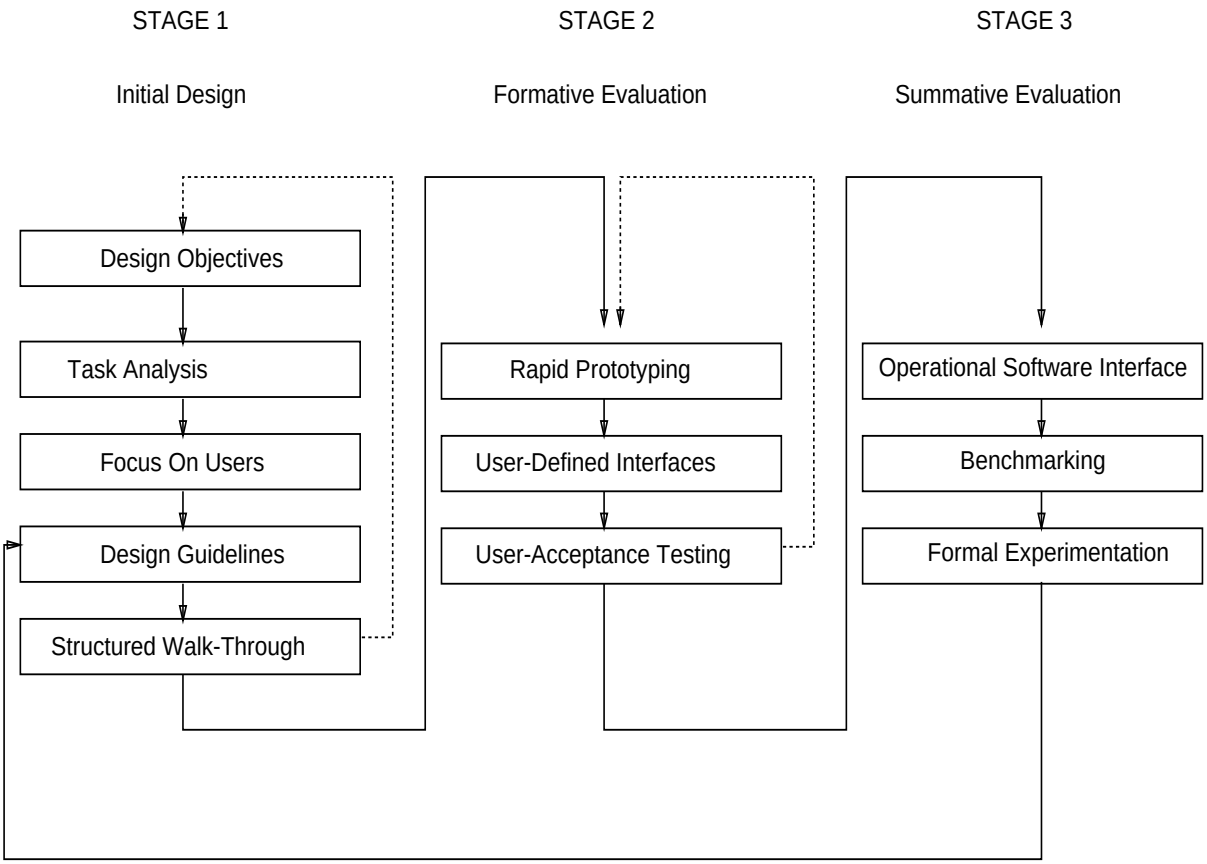
[37] E.L. Wiener and R.E. Curry. Flight deck automation : Promises and problems. *Ergonomics*, 23(10):995 – 1011, 1980.

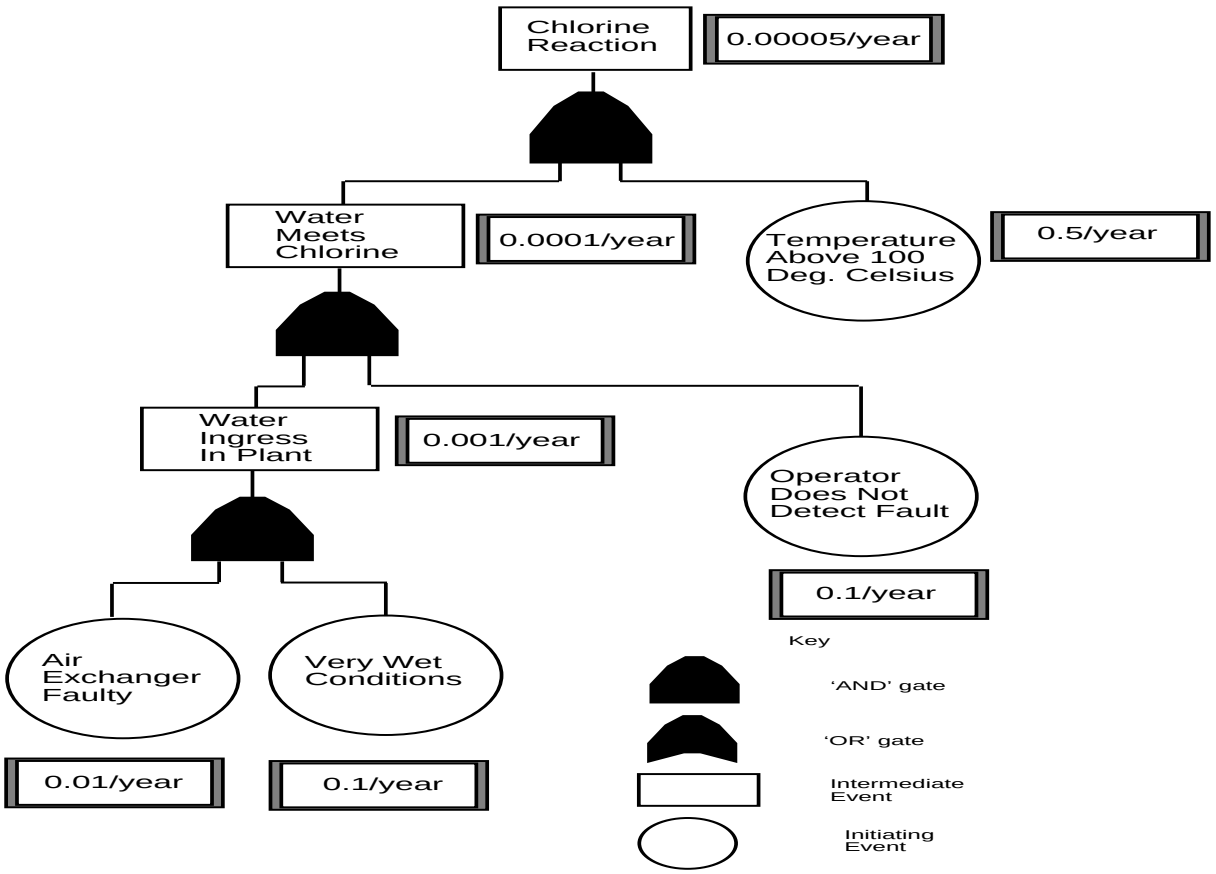
[38] G.J.S. Wilde. The theory of risk homeostasis: Implications for safety and health. *Risk Analysis*, 2:209–225, 1988.

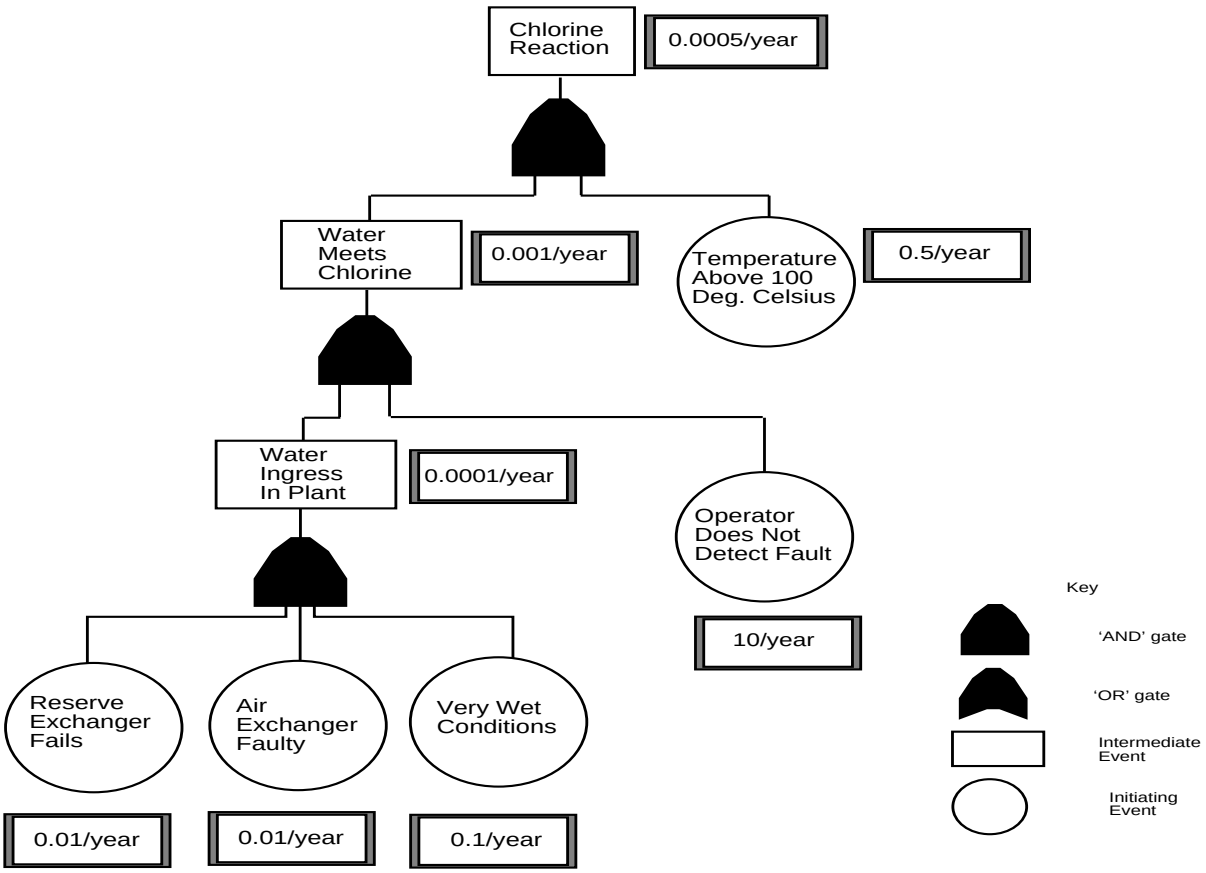
[39] R.C. Williges, B.H. Williges, and J. Elkerton. Software interface design. In G. Salvendy, editor, *The Handbook Of Human Factors*, pages 1416–1449. Wiley, New York, United States of America, 1987.

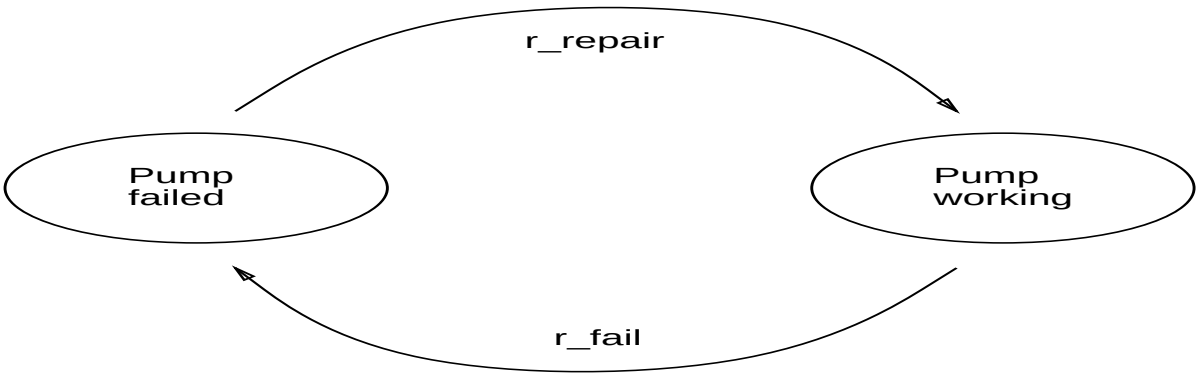
[40] N. Worley and J. Lewins, editors. *The Chernobyl Accident And Its Implications For The United Kingdom - Report Number 19 Of The Watt Committee on Energy.* Elsevier Applied Science, London, United Kingdom, 1988.

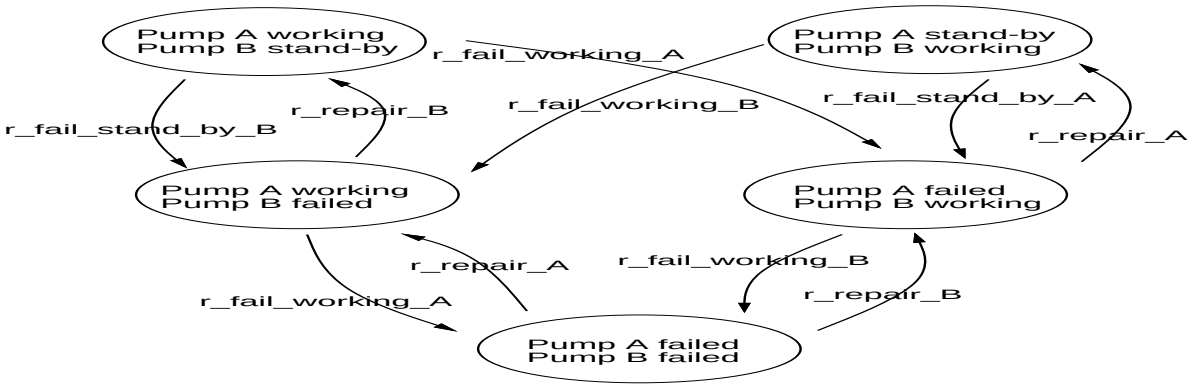

Hermeneutics
(Human Factors)
Observations
Of Use
Proposals
For A New
Artifact
Introduction And Use
The Artifact
Implementation
(Systems Engineering)

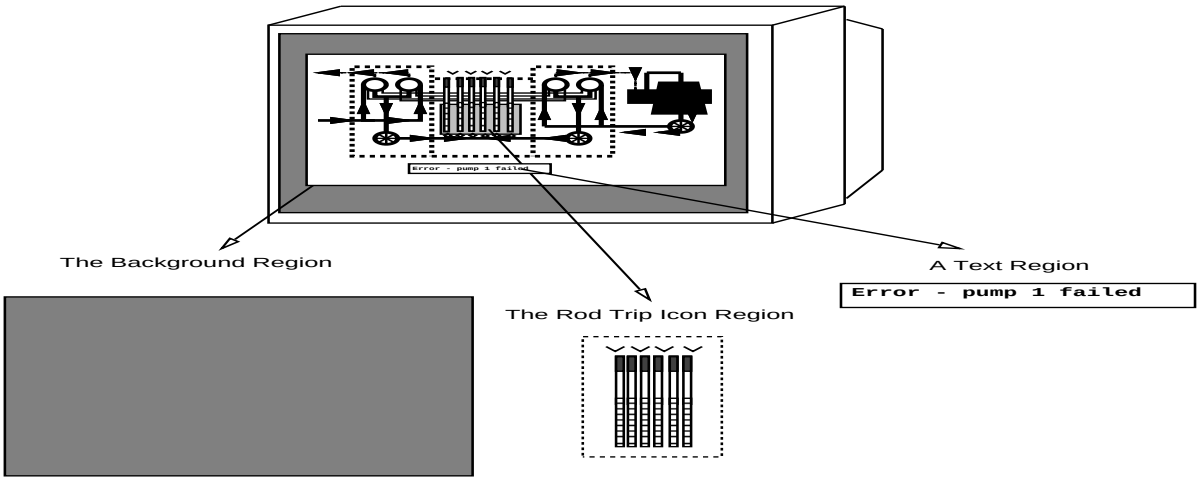

STAGE 1
Initial Design
STAGE 2
Formative Evaluation
STAGE 3
Summative Evaluation
Design Objectives
Task Analysis
Focus On Users
Design Guidelines
Structured Walk-Through
Rapid Prototyping
User-Defined Interfaces
User-Acceptance Testing
Operational Software Interface
Benchmarking
Formal Experimentation


Chlorine Reaction
0.00005/year
Water Meets Chlorine
0.0001/year
Temperature Above 100 Deg. Celsius
0.5/year
Water Ingress In Plant
0.001/year
Operator Does Not Detect Fault
0.1/year
Air Exchanger Faulty
0.01/year
Very Wet Conditions
0.1/year
Key
'AND' gate
'OR' gate
Intermediate Event
Initiating Event


Chlorine Reaction
0.0005/year
Water Meets Chlorine
0.001/year
Temperature Above 100 Deg. Celsius
0.5/year
Water Ingress In Plant
0.0001/year
Operator Does Not Detect Fault
10/year
Reserve Exchanger Fails
0.01/year
Air Exchanger Faulty
0.01/year
Very Wet Conditions
0.1/year
Key
'AND' gate
'OR' gate
Intermediate Event
Initiating Event


r_repair
Pump
failed
Pump
working
r_fail


Pump A working
Pump B stand-by
Pump A stand-by
Pump B working
r_fail_working_A
r_fail_working_B
r_fail_stand_by_A
r_repair_B
r_fail_stand_by_B
r_repair_A
Pump A working
Pump B failed
Pump A failed
Pump B working
r_repair_A
r_fail_working_B
r_fail_working_A
r_repair_B
Pump A failed
Pump B failed


The Background Region
The Rod Trip Icon Region
A Text Region
Error - pump 1 failed

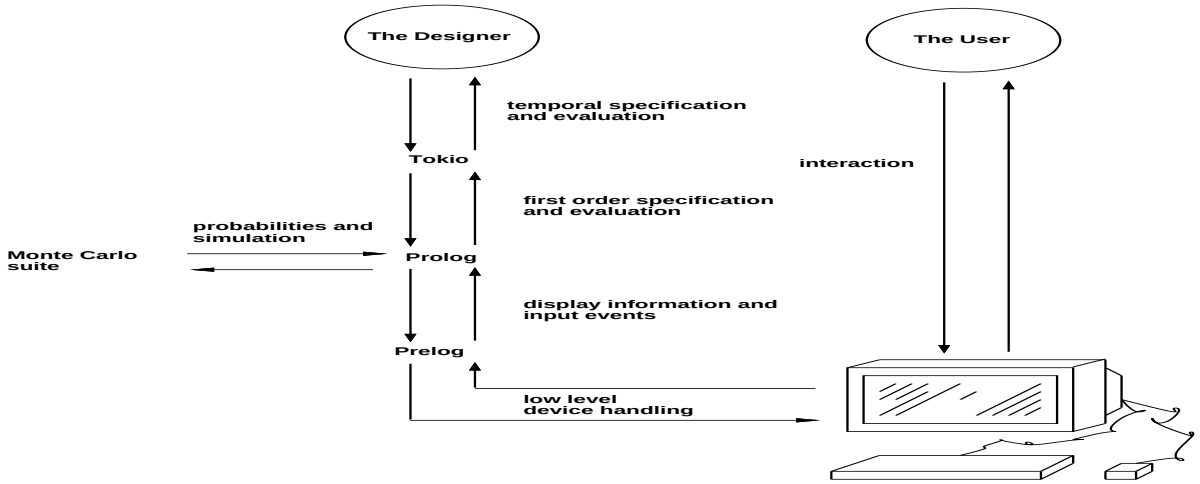

The Designer
The User
temporal specification
and evaluation
Tokio
interaction
first order specification
and evaluation
probabilities and
simulation
Monte Carlo
suite
Prolog
display information and
input events
Prelog
low level
device handling

Figure 1: The task-artifact cycle

Figure 2: A three stage model of interface development

Figure 3: Fault-tree Representing Quantified Probabilities

Figure 4: Fault-tree Representing Risk Homeostasis

Figure 5: A simple Markov model

Figure 6: A Markov model showing a stand-by pump

Figure 7: The region decomposition for part of the *coolant_error*

Figure 8: The extended Risklog architecture