DECISION THEORY AND SAFETY-CRITICAL INTERFACES

C.W. Johnson,

Dept. of Computing Science, Univ. of Glasgow, Glasgow, United Kingdom, G12 8QQ.
E-mail: johnson@dcs.glasgow.ac.uk Telephone: +44 41 3308855 Fax: +44 41 3304913.

Risk assessment techniques are being used to support the engineering of complex, safety-critical systems. They help to identify those situations that pose the greatest threat to an application. They can also be used to inform interface development. For example, operators must be presented with additional contextual information during infrequent, high-cost failures. Many risk assessment techniques cannot represent the temporal properties that affect usability. Few techniques provide means of reasoning about user input or output. This paper presents means of overcoming these limitations. It is argued that temporal logic can be used to represent sequential and concurrent properties of interaction. The same notation can also be used to specify presentation requirements. Temporal logic does not, however, provide means of representing risk. Decision theory is recruited to introduce probability and cost into the notation. The logic is then applied to reason about the different effects that low frequency, high cost failures have upon a user interface. A limitation of this approach is that the extended logic provides little impression of what it would be like to interact with a stochastic application. Simulation tools have been developed to avoid this limitation.

Keywords: *Risk assessment; cost; probability; temporal logic; decision theory; safety-critical interfaces.*

1 Introduction

Regulatory bodies, such as the United Kingdom's Health and Safety Executive and the United States' Nuclear Regulatory Commission, must ensure that complex production processes conform to stringent safety standards. Risk assessment techniques provide companies with means of demonstrating that applications fulfill these requirements. Quantitative reliability data about previous failures can be obtained from sources such as NASA's Aviation Safety Reporting System, the European Event Data Recording System and the U.K. Atomic Energy Authority's Systems Reliability Service. This data can also be used to guide interface design. Development resources must be allocated to provide operators with the information needed to respond to high probability system failures. It is important to realise, however, that risk cannot merely be represented in terms of frequency data. For instance, U.K. safety legislation is based around the assumption that: risk = frequency $\times$ consequence [5]. The costs or threats posed by system behaviours have important implications for interface development. Low frequency, high cost failures, typically, have a greater impact upon safety than high frequency, low cost faults.

Recent studies have identified the limitations of risk analysis techniques, such as fault trees or failure mode and criticality analysis, for interface design [6]. The following pages exploit a mathematical notation to support the development of human-machine interfaces. NASA are applying formal specifications to represent and reason about on-board communications systems [13]. Formal analysis is a necessary part of many design projects commissioned by the United Kingdom's Ministry of Defence [12]. Previous work into the use of mathematical specifications for interface design has focused on text-editors, drawing packages and conferencing systems [10]. Previous approaches have not considered the stochastic or probabilistic behaviours that characterise interaction with safety-critical processes. The following pages recruit work in economics and decision theory to avoid these limitations.

An interface for a chlorine recovery plant is used to illustrate the remainder of this paper. This system removes impurities from chlorine gas and improves its liquefaction efficiency [3]. The recovery process involves the compression of gasses vented during liquefaction. Figure 1 outlines the

main stages in this process. We have chosen this application because it suffers from many of the problems that frustrate the design and operation of safety-critical control systems. It suffers from periodic hardware faults. Software control systems must respond to such failures; operators must be alerted. The chemical properties of chlorine are well understood but its corrosive nature can make its handling especially difficult. Such hazards raise significant problems for the detection and presentation of process information.

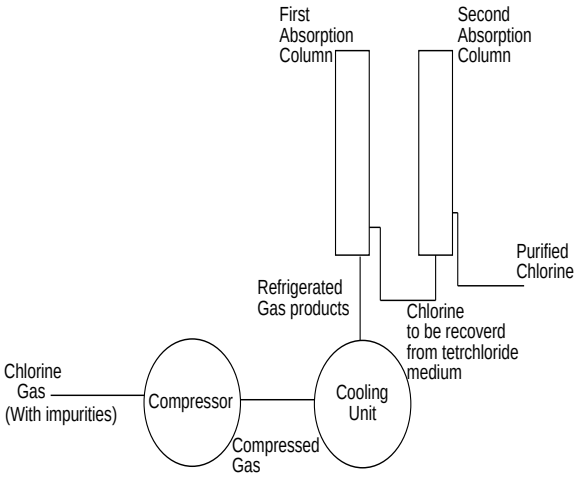

Figure 1: Block Diagram For The Recovery Process

2 Probability

Probabilistic Computation Tree Logic (PCTL) has been developed to demonstrate that logic can support risk analysis. The syntax and semantics of this notation are presented in [6]. In contrast, this paper focuses upon the application of the formalism to guide interface development. For instance, reliability data for our exemplar system predicts 76 failures per operating year for reciprocating chlorine compressors [3]. Similar data is available for analyser failures in the first absorption column:

$$[pump_error\ Pr\ 0.00000241] \quad (1)$$
$$[analyser_error\ Pr\ 0.0000000003] \quad (2)$$

The probability of the pump failing in the next second is 0.964×10^{-6}. The probability of a column analyser failure is 0.3×10^{-9}.

From these formulae, designers can draw a number of conclusions about the behaviour of a potential interface. For any user, *analyser* faults and *pump* failures will be rare events. This 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 [14]. 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. Considerable perceptual cues will also be required to alert users to these rare failures. Operators tailor their polling strategy to reflect the probabilistic behaviour of an application; less attention is allocated to the observation of low frequency errors [17]. Interface designers might, therefore, provide audible warnings to ensure that operators are informed about the loss of their pumping equipment. The following PCTL clause represents this presentation requirement at an extremely high level of abstraction, $display_pump_warning$ and $audible_pump_alarm$. It strips out the low level device details, such as bitmap structures, that can obscure important properties of an interface during the early stages of development. It also describes an input requirement, $user_warning_acknowledge$. This requirement might be satisfied by implementations that support mice, keyboards, dedicated buttons or touch sensitive screens. A previous paper describes how these details may be gradually introduced as development progresses [9]. The $\mathcal{U}$ (read as 'until') operator can be used to express the duration of the alarms:

$$\begin{aligned} &audible_acknowledge_error \Leftarrow \\ &pump_error \wedge \\ &\quad A[(display_pump_warning \wedge \\ &audible_pump_alarm)\ \mathcal{U} \\ &user_warning_acknowledge] \end{aligned} \quad (3)$$

An audible warning is acknowledged if the pump fails and a warning is displayed and an audible alarm persists until a user acknowledges the warning.

Empirical evidence is required to ensure that operators will detect and respond to particular presentation formats within acceptable time limits. Executable subsets of formal notations can be used to validate such requirements. Sections 4 will describe simulation tools that can be used to implementation

prototypes from PCTL specifications [6].

3 Decision Theory

Clauses (1) and (2) capture the observation that a *pump* failure is more probable than an *analyser* error. Designers might use this information when developing training material and system documentation. Resources may be allocated in proportion to the probability of particular faults. Risk assessments must, however, consider both the frequency and severity of system behaviours. Design resources are allocated to those situations that pose the greatest threat to safety even if they are relatively unlikely. Research in economics and decision theory has developed means of representing cost and probability [15]. This can be applied to interface design by using decision vectors, $(f^1, p^1; ...; f^n, p^n)$, to associate probabilities with failures. In economics these vectors are used to associate probabilities with the returns on investment decisions. A relation, $\succ_{risk}$, can then be defined to rank the threats posed by fault vectors. This relation hides an underlying model of cost, determined by a utility function u, in the same way that PCTL's temporal operators, such as AX (read as 'in all next states'), hide a model of time:

$$\forall s \in S : V(s) = (\sum_{i=1}^{n} p^i) u(f^1 \wedge ... f^n) \quad (4)$$

$$\forall s, s' \in S : s' \succ_{risk} s \leftrightarrow V(s) > V(s') \quad (5)$$

The first equation presents a valuation function that applies a utility function, u, to the conjunct of the propositions contained in the vector, $(f^1 \wedge f^2 \wedge ... f^n)$. The second equation uses the valuation function to derive a preference relationship between proposition vectors.

A number of propositions can be included in these vectors to describe situations in which concurrent failures exacerbate control problems. This is important because such situations frequently lead to increases in operator workload. Probabilities are not re-calculated to account for failure dependencies or contingent evidence. If necessary, standard probabilistic techniques can be applied to derive such measures from the information contained in propositional vectors [1]. The following clause uses $\succ_{risk}$ to capture the observation that a single high probability, low cost fault poses less of a threat than a lower probability, higher cost compound failure:

$$pump_v(exchanger_error, 0.0000000003, pump_error, 0.00000241) \succ_{risk} analyser_v(analyser_error, 0.0000000003) \quad (6)$$

The compound failure of a pump and the heat exchanger poses a greater threat than a single failure of a column analyser.

It is important to identify such priorities early in the development process. Systems engineers must allocate additional development resources to detect the compound failure. Interface designers must allocate additional presentation resources to warn operators about situations in which both the pump and exchanger fail. The high costs associated with these failures means that they pose significant threats to safety. Their low probability implies that operators may not be aware of an appropriate response. The extended PCTL notation provides a common medium in which to specify these different requirements. Systems engineers must detect the faults, *exchanger_error* and *pump_error*. Interface designers must present warnings, *display_exchanger_warning* and *display_pump_warning*:

$$compound_failure_warning \Leftarrow exchanger_error \wedge pump_error \wedge AX(display_exchanger_warning \wedge display_pump_warning) \quad (7)$$

There is a warning of a compound failure if $pump_a$ and the heat exchanger fail in the present interval and in all next states a warning is displayed about these failures.

The use of a formal notation does not guarantee that designers will produce safe and 'usable' systems. For instance, previous clauses have not considered what might happen if the exchanger and pump failed at the same time as problems developed in the column analyser. The $\succ_{risk}$ construct can be used to reason about operator and system priorities during such situations. Systems engineers could coordinate development resources to resolve those failures that pose the greatest threat. Interface designers might exploit the $\succ_{risk}$ relation to order the presentation of warning displays. This is necessary

when initial failures may cause 'knock-on' problems in related processes. The concurrent presentation of many different warnings can over-load the scarce cognitive and perceptual resources of system operators [17]. From clause (6) we know that the column analyser carries a lower risk than the pump and exchanger failures. Designers might, therefore, specify that the analyser failure is eventually presented and that the pump and column failures are displayed as soon as possible:

$$ordered_failure_response \Leftarrow analyser_error \wedge compound_failure_warning \wedge AX(start_standby_pump \wedge EF(display_analyser_warning \wedge re_route_analysis)) \quad (8)$$

There is an ordered response to failure if in the present interval the column analyser and the exchanger and the pump all fail and in the immediate next interval the stand-by pump is started, a warning is presented for the pump and exchanger. Eventually, the analyser is re-routed and a warning is presented for this failure.

The previous requirement illustrates the need to integrate human factors and systems engineering. Clauses, such as (6), help to establish common objectives for application development and interface design. Exchanger and pump failures are handled and presented before analyser failures. If common objectives are not established then control systems might identify and respond to high-risk failures without notifying the operator. Numerous accidents have been caused when users have not been provided with appropriate feedback about system intervention [14].

4 Simulation

PCTL clauses provide an extremely poor impression of what it would be like to interact with an application. Simulation tools can be used to address this problem. PROLOG has been extended to support our temporal, probabilistic logic. The architecture of the Risklog system is described in more detail in [6]. Current implementations support the development of multi-user graphical interfaces [8]. A limitation with conventional logic prototyping tools, including PROLOG, is that they are extremely deterministic. In clauses such as (8) if a developer asserted that the *pump* failed during the first run of a prototype then it would also fail for every other run. This makes it very difficult to provide operators with a realistic impression of process behaviour. Designers must change the facts that hold for many different intervals in order to accurately model the probable behaviour of a safety-critical system. Risklog exploits Monte Carlo techniques to avoid this burden. Random numbers are used to determine whether or not a fact holds during a particular interval of interaction. From (6) we know that the probability of *pump* failing is 0.241×10^{-5}. This frequency 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.

A limitation with Risklog is that our simulations are too realistic. Low frequency events occur with a relatively low frequency. High probability events occur frequently. In consequence, operators might have to use the prototype for several years before experiencing the pathological failures described in previous sections. Recent work has extended the tool to avoid this. The $\succ_{risk}$ relation is used to guide simulations towards particular areas of the the cost-probability spectrum. Designers can simulate a series of low frequency, high cost failures. This is done by temporarily increasing the probabilities associated with high-risk situations, such as the exchanger and pump failure of (6). During other simulations, standard Monte Carlo techniques can be used to produce more 'typical' traces of interaction.

5 Further Work

Bainbridge argues that interface designers must elicit the justification for system actions rather than simply identifying functional behaviours [2]. Such an understanding is a pre-requisite for the development of displays that support the operation of complex processes. This might be encouraged by building syntactic constructs around the PCTL notation. For instance, the following 'formal rationale' groups a clausal requirement, a propositional vector and a

natural language description to present the justification for a development decision:

`Requirement:`

$$ordered_failure_response \Leftarrow analyser_error \wedge compound_failure_warning \wedge AX(start_standby_pump \wedge EF(display_analyser_warning \wedge re_route_analysis))$$

`Justification:`

$$pump_v(exchanger_error, 0.0000000003, pump_error, 0.00000241) \succ_{risk} analyser_v(analyser_error, 0.0000000003)$$

`Description:`
There is an ordered response to system faults if the analyser, exchanger and pump fail in the present interval. In the next interval, the stand-by pump is started and a warning is displayed for both the pump error and the exchanger failure. Eventually, the analyser is rerouted and an analyser warning is displayed. The justification is that the compound failure of the exchanger and the stand-by pump has a higher cost than the analyser failure.

This style of specification suggests a number of areas for further work. Designers must consider many different factors during the development of safety-critical interfaces. Development decisions cannot be justified purely in terms of relative risk. Regulatory pressures and managerial influences can affect the decision making process [7]. Future research intends to extend the vocabulary of clauses that might appear in the `Justification` section. It is hypothesised that the flexible argumentation structures of graphical notations, such as QOC, might be gradually transformed into more formal representations as development progresses [11]. Proof techniques could then be applied to check for contradictions or inconsistencies in the rationale that supports an interface.

Rasmussen asserts that "designs should only be accepted if the human contribution to risk can be measured" [16]. For instance, Hawkins cites error rates of 1×10^{-2} for push button selection in safety critical systems [4]. He also argues that "under certain circumstances human reliability can improve by several orders of magnitude". Interface designers must be more precise about the particular circumstances that will achieve such a reduction. PCTL could be used to represent the probability associated with particular forms of operator 'error'. It could also be used to specify the techniques that are intended to reduce operator error. Recent research has exploited this approach to derive safety requirements from the products of accident analyses [7]. It is hypothesised that these techniques might provide a means of integrating the reliability models of human factors research into the systems engineering of complex applications.

6 Conclusions

This paper has argued that risk analysis can be used to guide interface design for safety-critical systems. Human factors and systems engineers must coordinate their responses to high-risk failures. The PCTL notation has been extended to support this task. Elements of decision theory have been recruited to represent probability and cost. A $\succ_{risk}$ relation ranks the threats posed by application failures. The extended notation can also represent the system and operator requirements that are intended to combat these failures. A limitation of this approach is that the logic provides little impression of what it would be like to interact with a stochastic application. The Risklog tool has been introduced. This uses PCTL clauses to simulate safety-critical interfaces. Designers can exploit the $\succ_{risk}$ relation to guide simulations towards particular areas of the the cost-probability spectrum. This is done by temporarily increasing the probabilities associated with high-risk situations. During other simulations, standard Monte Carlo techniques can be used to produce more 'typical' traces of interaction.

Acknowledgements

The work described in this paper was conducted while the author was a member of the Department of Computer Science, University of York, York, United Kingdom. This work is funded by the United Kingdom's Science and Engineering Research Council, grant number GR/J07686.

References

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

[2] L. Bainbridge. Ironies of automation. In J. Rasmussen, K. Duncan, and J. Leplatt, editors, *New Technology And Human Error*, pages 271 – 283. J. Wiley and Sons, New York, United States of America, 1987.

[3] 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.

[4] F.H. Hawkins. *Human Factors In Flight*. Gower Technical Press, Aldershot, United Kingdom, 1987.

[5] G.H. Helsby and R.F. White. Criteria for use in the assessment and control of major hazards. In J. Burgoyne, editor, *The Assessment And Control Of Major Hazards*, pages 273–289. Pergamon Press and The Institution Of Chemical Engineers, Oxford, United Kingdom, 1985.

[6] C.W. Johnson. A probabilistic logic for the development of safety-critical interactive systems. *International Journal Of Man-Machine Studies*, 39(2):333–351, 1993.

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

[8] C.W. Johnson and M.D. Harrison. Declarative graphics and dynamic interaction. In F.H. Post and W. Barth, editors, *EUROGRAPHICS '91*, pages 195–207. Elsevier Science Publications, North Holland, Netherlands, 1991.

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

[10] C.W. Johnson and M.D. Harrison. Software engineering for human computer interaction. *SIGCHI Bulletin*, 26(2):46–48, 1994. Editorial For Special Edition On Human Factors And Software Engineering.

[11] A. MacLean, R.M. Young, V.M.E. Bellotti, and T.P. Moran. Questions, options, and criteria: elements of design space analysis. *Human-Computer Interaction*, 6(3-4):201–50, 1991.

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

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

[14] 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.

[15] C. Puppe. *Distorted Probabilities And Choice Under Risk*. Lecture Notes In Economics And Mathematical Systems, No 363. Springer Verlag, Berlin, Germany, 1991.

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

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