CREWS Report 98-36

submitted to

4th International Symposium on Requirements Engineering, RE'1999.

Human Errors and System Requirements

by

Alistair Sutcliffe, Julia Galliers and Shailey Minocha

Centre for HCI Design
School of Informatics
City University
Northampton Square
London EC1V 0HB
United Kingdom

+44 (0)171 477 8412
{a.g.sutcliffe, **Error! Bookmark not defined.**
jrg@csr.city.ac.uk

Human Errors and System Requirements[1]

Alistair Sutcliffe, Julia Galliers and Shailey Minocha

Centre for HCI Design,
School of Informatics,
City University,
Northampton Square,
London EC1V 0HB, UK
E-mail: a.g.sutcliffe@city.ac.uk
Tel: +44-171-477-8411
Fax: +44-171-477-8859

Abstract

This paper reports a method of assessing the implications for human error on system requirements, a topic not usually considered during requirements engineering (RE). In our previous work, we proposed a taxonomy of influencing factors that might contribute to human error. This paper takes the taxonomy and elaborates it to suggest generic requirements to deal with problems in different layers of the taxonomy. Components of the taxonomy are combined into a causal model for error, represented as a Bayesian Belief Net (BBN). BBNs model the error influences arising from user knowledge, ability, and the task environment. These are combined with factors describing the complexity of action and user interface quality in scenarios of projected system usage. The BBN model predicts probabilities of slips and mistakes. These are assessed according to action types in the scenario to suggest generic requirements to prevent the error or to deal with its consequences. The method is illustrated by a service engineer support system application.

1 Introduction

Few methods advise on how to investigate the consequence of human errors for system requirements. Analytic methods have been reported in the safety critical literature ranging from use of design rationale and formal techniques for failure analysis (Johnson 1996) to more general approaches with guidelines to prevent human error (Leveson 1995, Hollnagel 1993). In RE, Fields et al. (1995) have investigated tasks as a means of linking error analysis with requirements investigation. The Inquiry Cycle (Potts et al. 1994, Hsi & Potts 1995) takes a wider ranging view of problems, called obstacles, which give rise to system requirements by analysing user-system dependencies; however, no detailed guidance for requirements engineers has been reported. In our previous work, we proposed a Scenario based Requirements Analysis Method (SCRAM) (Sutcliffe 1995, 1997) that employed scenario scripts in a walkthrough method that validated design options for 'key points' in the script.

In safety critical systems accurate foresight is a pressing problem, so we used taxonomies of human error (Reason 1990, Hollnagel 1993), to investigate scenarios of future system use (Sutcliffe et al.,1998). Although we could draw the analyst's attention to the potential sources of human (and system) error that may need to be considered during requirements analysis, this advice still required considerable interpretation, and

[1] This research has been funded by the European Commission ESPRIT 21903 'CREWS' (Co-operative Requirements Engineering With Scenarios) long-term research project.

did not predict which action-steps in a scenario may be more prone to error. This paper reports work we have undertaken to address this problem by employing Bayesian Belief Networks (BBNs) to analyse scenarios for human error and identify requirements to deal with them. The paper is organised in three sections. First, we introduce our method for assessing the possible impact of human error on system requirements. In section 3, we describe application of BBNs as a method and tool support for error-related scenario analysis. Section 4 illustrates the method and BBNs with a case study taken from a service engineer support system application. The paper concludes with a brief discussion of the potential of BBNs in RE.

2 Framework and Method for Error Analysis

The method is based on three simple ideas: first, that requirements are analysed by creating scenarios as threads of behaviour though a use case, adopting an object - oriented approach (see Jacobson 1992). Second, scenarios are walked-through asking if an error may occur at each stage given a set of influencing factors that contribute to the likelihood of errors. Generic requirements are proposed to deal with the consequence of errors and to mitigate any problems that may arise due to the influencing factors. Generic requirements are reusable knowledge that can be used as an agenda of issues pointing towards areas for further requirements investigation, or requirements that can be refined with more domain-specific knowledge, or high level design solutions that may be added directly to a requirements specification. The components of the method are given in Figure 1.

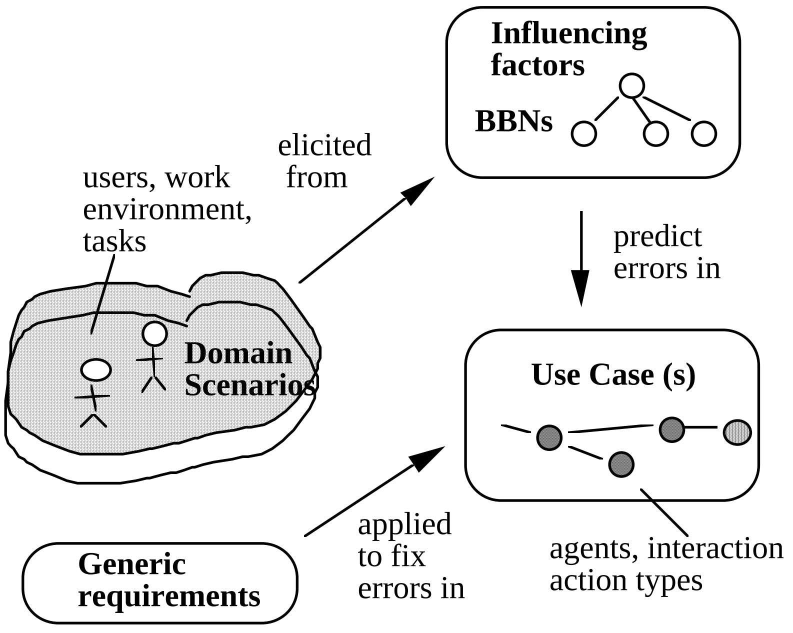

Figure 1 Components of the method: Influencing factors are elicited from domain scenarios and predict human error in use cases. Generic requirements are proposed to prevent errors or as remedial measures

The method described in this paper uses two types of scenarios. First, 'domain' scenarios that capture information about the system, its users, workplace context, and other environmental information. Such scenarios are descriptions of the current system, its environment with ’day in the life of’ stories of use. Secondly, scenario ’scripts" as test data threads generated from a use case and requirements are refined through investigation of errors that originate in the environment from users or other agents.

2.1 Method Stages

The method stages are summarised in Figure 2. The method is iterative, so with a set of first cut use cases, all the other stages proceed concurrently leading to refinement of use cases and the requirements specification.

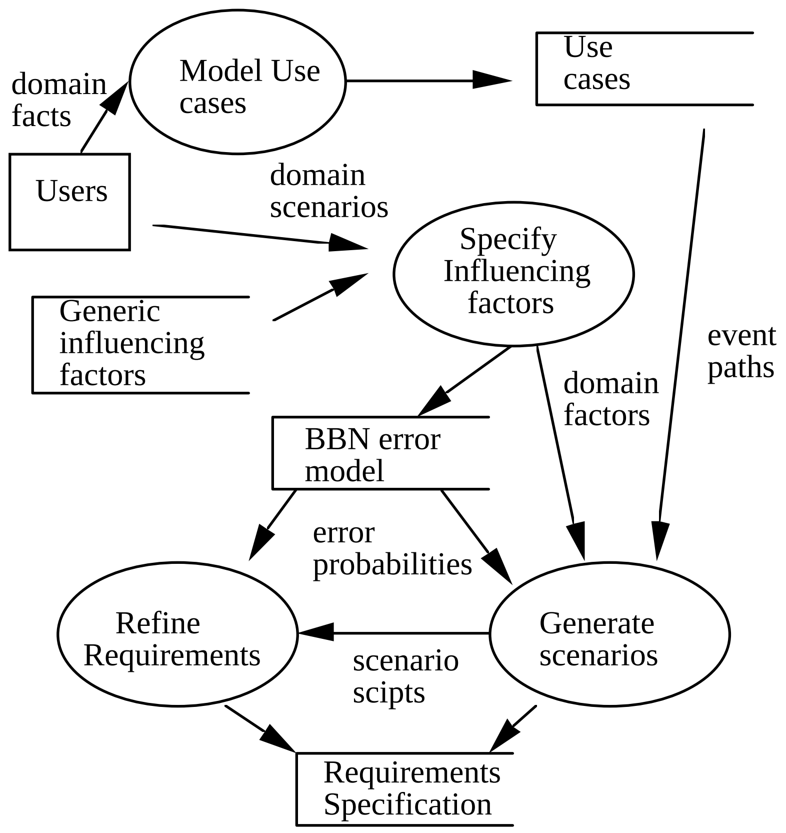

Figure 2 Lower level method stages and associated models.

Stage 1 Use Case Modelling

Use case modelling follows standard OO procedures (Jacobson 1992), so it is not described with in this paper. A preliminary system investigation identifies the major agents, their activities, and goals of the system. Use cases are created for system goals, and describe the agents involved and the event/information flows between them. Use cases are expressed as interaction diagrams that map out the sequential dependencies between human and system agents, following standard UML (UML 1997) notation.

Stage 2 Specify Influencing factors

A set of influencing factors are provided, separated into different layers for task-workload, personnel and training, and the environment. Factors within each layer are investigated to see if they are relevant to the application domain being modelled. Domain scenarios describing the working environment are a good source of information on influencing factors. Tables of influencing factors, their probable consequences and generic requirements to prevent or counteract the consequent errors are provided to help the requirements engineer deal with issues in each layer. Each factor is rated on a 3 point scale (high, medium, low) to estimate the severity of its potential impact on errors. Factors are aggregated to give an overall likelihood of error using BBNs. The influencing factors are used to estimate the overall probability of error emanating from the users and the system environment.

Stage 3 Generate scenarios

This step generates scenarios by walking through each event sequence or pathway in the use case. Each pathway becomes a scenario. Thus one use case will give rise to many scenarios if there is any non-determinism or concurrency in the use case model. Scenario generation is supported by the CREWS-SAVRE tool (Sutcliffe et al. 1998) which automatically identifies all possible pathways through the use case. The BBN influencing factor analysis is then used to 'walkthrough' a scenario script and assess which events and user behaviours may be prone to failure. Every scenario is walked through and the analyst inquires whether an error may occur at each stage.

Stage 4 Refine requirements for human error

Influencing factors are used to determine which types of error (slips and mistakes) are more likely to occur. The analyst identifies the actions which are more error-prone using the BBN analysis with a description of the cognitive and physical complexity of each action-step in the scenario. Actions are typed as input/output or decision support. A set of generic requirements linking error to action types are provided to stimulate further requirements analysis. High probability errors indicate a branch in the original scenario hence a new 'abnormal course' scenario is created to describe error recovery actions. The outcome is a set of formatted use cases, scenarios and elaborated requirements specifications.

2.2 Influencing Factors

To help the software engineer anticipate when exceptions may occur and assign probabilities to abnormal events, a set of influencing factors are provided. Influencing factors analysis has two purposes, first it supplies high level generic requirements associated which each set of factors, and secondly the factors form input for a causal model of error that is used in the BBN analysis (see section 3).

In a previous paper (Sutcliffe et al. 1998), we reported four groups of factors (environmental conditions, management, task/domain and user/personnel qualities) that affected human internal variables such as fatigue, stress, workload and motivation. In this paper we only deal with two layers of influencing factors for the task/domain and user/personnel. Modelling event causality is complex, and the analysis effort that is warranted will depend on the safety criticality system, so two approaches are offered.

First is to use the method as a paper-based ‘tool for thought’. Alternatively, the factors may be entered into the BBN tool to perform a scenario analysis.

Domain scenarios, describing the working environment, the people who will operate, control and manage the system, are used to capture influencing factors. Such scenarios can either be taken from real-life or made up to cover a variety of organisational and work situations that may occur in the domain. The implications of user factors are summarised in Table I. Some of these factors can be measured objectively by using psychological questionnaires. For instance general ability and accuracy/concentration can be measured by intelligence aptitude scales, decision making and judgement by locus of control scales, while domain and task knowledge can be measured by creating simple tests for a specific task/domain. The main implications of the personnel factors are for personnel selection and training. while generic requirements indicate the need for computer based intelligent assistants, critics, and aide memoir information displays.

Table I User / Personnel Factors with problems for slip and mistake type errors and generic requirements to counteract these problems

Personnel Qualities	**Skilled Task Problems**	**Decision / Problem Solving Tasks**	**Implications / Generic Requirements**
General Ability	More slips especially complex tasks, longer learning time	Inability to deal with unexpected events	Personnel selection appropriate for task / job description
Knowledge of domain	More slips, longer learning time	Slow performance, failure in complex problems	Improve training on the job experience, scenario-based training
Skill / task knowledge training	Slow operation, more slips	Mistakes, slow performance	Improve training, help manuals, aid memoirs, mentors
Judgement / decision-making	-	poor initiative, more mistakes, wrong decisions	Select personnel appropriate for task
Concentration / accuracy	More slips, more ordered events	more mistakes, poor decisions, poor checking	Discipline and training to improve performance, select appropriate personnel
Motivation	More slips, slow operation, short cuts	Increase mistakes, slow performance, short cuts	Improve incentives, job satisfaction, select appropriate personnel

Human error, following the distinction drawn by Reason (Reason 1990) is divided into slips, which result from lapses in attention and sensory-motor co-ordination, and mistakes which are more complicated failures in reasoning. The impact of poor personnel factors will be worse skilled performance and action-slip errors, as shown in column 2; while for tasks that require decisions and problem solving there will be worse performance and more mistakes. These are illustrated in column 3, with generic requirements and recommendations for training are listed in column 4. In addition, user personnel factors are used with assessment of task complexity (see Table II), to match users' abilities to tasks of appropriate complexity, while also giving people sufficient challenge and responsibility in their job. Task allocation and work design are complicated issues, beyond the scope of this paper, however see (Bailey 1982) for more details.

Table II Task / Domain factors with implications for errors and generic requirements.

Task factor	Implications	Requirements / Training issues
High Volume	Delays, bottlenecks, performance degrades, fatigue	Buffer & smoothing workloads, automate if possible, design breakpoints, batch schedules
Complexity	User fatigue and stress, mistakes, problem solving failure	Match complexity to user abilities and experience, decompose task, simplify procedures
Repetitiveness	Boredom, poor attention, slips, missed events	Automate if possible, provide task variety, swap operators frequently
Interruptions	Attention slips, missing and mal-ordered events, capture errors	Provide aid memoirs, agendas, status / progress checks, screen-out unwanted events
Time pressure	Late events, slips, capture errors, omissions	Smooth event arrival, automate task, give user time to think, provide holding actions
Multitasking / task switching	Attention slips, missing and mal-ordered events, losing thread problems	Aid memoirs, agenda managers, clear mode / status indicators, schedule switching if possible

Task/domain factors are elicited by questions about the task complexity and environmental constraints on operators. As with the other factors, the complete documentation of the method (Sutcliffe & Minocha 1998) provides a comprehensive list of questions and elicitation techniques. Implications for task/domain factors for user performance and errors are listed in column 2 of Table II, with generic requirements and user training recommendations in column 3. Increases in interruptions, and more task switching make slip type errors more likely, whereas higher volumes and time pressures lead to more errors and delays. Complexity and repetitiveness, on the other hand, have implications for task allocation and increased mistakes if users are not given tasks that match with their abilities, or training to carry out their allocated tasks. It should be noted that many task performance problems also have 'knock-on' effects on users through increased levels of stress and fatigue.

Domain specific models may be created by selecting a sub-set of the influencing factors that apply to an application. Many combinations of influencing factors are possible and each domain requires a particular model, hence we provide a general modelling tool that can be instantiated with domain specific information. The generic model takes a sub set of the key factors that apply to most domains. The tool allows influencing factors to be entered as a rating on a three point scale (e.g. high, medium, low complexity) and then calculates the event probability from the ratings. The model is configured by assigning a set of probabilities for each combination of influencing factors. A set of default formulae for inter-factor weights are provided, but these can be adjusted depending on the user's knowledge of the domain.

3 Bayesian Belief Network Analysis

In this section, we introduce BBNs as a means of combining the influencing factors into a more formal and predictive model of human error. BBNs are graphical networks that represent probabilistic relationships between variables. They offer decision support for probabilistic reasoning in the presence of uncertainty and combine the advantages of an intuitive representation with a sound mathematical basis in Bayesian probability (Pearl, 1988). BBNs are useful for inferring the probabilities of events which have not as yet, been observed, on the basis of observations or other evidence that have a causal relationship to the event in question. For example, a doctor might have observed a variety of symptoms in his patient. Using a BBN, s/he can determine the probabilities that these symptoms are caused by each of the several possible alternative diseases, and hence further complications that might arise.

A BBN is made up of *nodes* and *arcs*. The nodes represent variables and the arcs represent (usually causal) relationships between variables. The example in Figure 3 is a fragment of the net described in more detail in the following section of this paper. It shows the complexity of an action as a variable that is affected causally by two factors - the level of physical detail and the cognitive complexity of the task. Variables with either a finite or an infinite number of states are possible in a BBN, so the choice of measurement scale is left to the analyst's discretion. In the case study we assign these variables to one of the three possible states: *high, medium* or *low*.

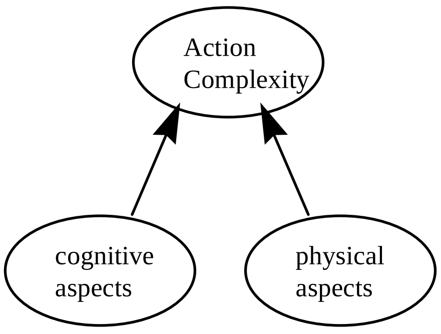

Figure 3 Simple BBN fragment, the two prior nodes have a causal influence on the posterior node of task complexity

In the above fragment, if we know that both the cognitive and physical aspects of a particular action are high, then the probability of the overall complexity being high is greater than if we know the action has a low level of physical detail and involves little cognitive ability. In the BBN we model this by filling in a node probability table (NPT). Table III shows the NPT for the *task complexity* node.

Table III Node Probability Table (NPT) for the *task complexity node*

	Cognitive	high			medium			low		
	Physical	high	medium	low	high	medium	low	high	medium	low
action complexity	high	0.9	0.7	0.6	0.8	0.5	0.4	0.7	0.2	0.1
	medium	0.1	0.2	0.3	0.15	0.3	0.3	0.2	0.3	0.2
	low	0.0	0.1	0.1	0.05	0.2	0.3	0.1	0.5	0.7

Nodes with incoming arcs are associated with a table of conditional probabilities as shown in Table III. Each arc and tables of conditional probabilities represent knowledge about one node that is useful for predictions about another node. Hence in Table III, column 1 the requirements engineer has asserted that if cognitive aspect of the action is high and the physical detail of the action is high then the probability of overall action complexity being high is 0.9, and medium 0.1 with a zero probability of being low. The table is configured by estimating the probabilities for the output variables by an exhaustive pairwise combination of the input variables. BBNs can accommodate both probabilities based on subjective judgements (elicited from domain experts) and, probabilities based on objective data.

When the net and NPTs have been completed, Bayes theorem is used to calculate the probability of each state of each node in the net. The result of this calculation is a probability distribution for the states of each node. Then, if evidence is available to determine the states of particular nodes from particular scenarios, the values entered are propagated through the network, updating the values of other nodes. These calculations are entirely automated by a BBN tool - *Hugin Explorer* (see Hugin A/S web site, *www.hugin.dk*). The result is a network from which predictions can be made regarding the probability of certain variable(s) being in particular state(s), given the combination(s) of evidence entered.

The generic model of influencing factors that give rise to human error is illustrated in Figure 4. Three groups of factors (task/domain, user knowledge and personnel qualities) affect 3 other factors (situation, aptitude and expertise). These, in turn, are combined with motivation to influence slips and mistakes. We have modelled all the user factors listed in Table I apart from concentration which varies between individuals and tasks, so it is difficult to estimate. From the task/domain table, multitasking has been merged with interruptions because both disrupt the normal flow of work; repetitiveness and volume have not be used as these were judged to be less important influences on error. Complexity is modelled at the action rather than task level.

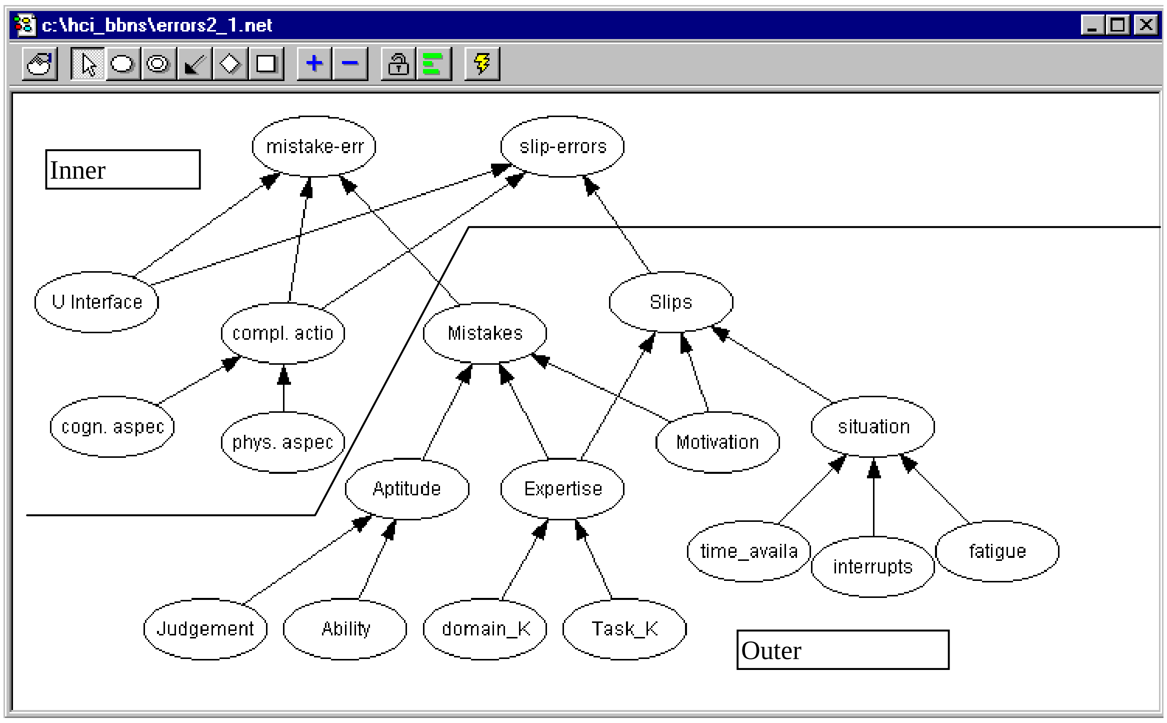

Figure 4 Generic BBN network for error analysis

Time, interrupts and fatigue all describe the environment's situation and the NPT is configured so that it contributes to slips more strongly than to mistakes. User knowledge, composed of domain and task sub-sets, contributes to both types of errors but more strongly to mistakes, while general ability (aptitude and judgement) influences mistakes more strongly. User motivation affects both types of error. So far the BBN has described general properties of the user and task context. This outer layer net is then merged into the inner layer net that describes design qualities of the user interface, and characteristics of each user action in the scenario. User actions are rated according to their cognitive complexity and physical operational detail. These factors in turn affect the probability of human errors as either mistakes or slips which are finally manifest as event exceptions in either human-computer interaction or in human action.

3.1 Impact Analysis Method

Domain scenarios are used to gather inputs for the outer BBN variables (see Figure 4) of the user's knowledge, ability and the situation. The BBN is then run against a scenario script of the user's behaviour with the system, essentially following a use case, or scenario as one pathway through a complex use case. Scenario scripts can also be generated from interaction diagrams of use cases. The BBN is run to produce an error probability for each action in a scenario script for a particular environment scenario (see Section 4).

Scenarios are run with the NPT-configured BBN by setting the values for the input nodes. For instance, in the service engineer application, the variables describe a generalised model of service engineers. In one scenario, the domain and task knowledge are set to high, as training and experience should equip all engineers, apart from trainees, with knowledge of the equipment they service. Ability and judgement are set to medium reflecting the usual educational qualifications and abilities of personnel recruited into these jobs. Motivation is set to medium as company loyalty was not judged to be high, while time pressure and interruptions were high as many calls have to be completed to tight deadlines and interruptions in customer premises are common. This scenario is then run against a scenario script for diagnosing faults in photocopiers.

For each run the following estimates have to be made at the action level in the script:

(a) The action has to be assessed as high/medium/low cognitive complexity. Any actions requiring complex decision-making, problem-solving or judgement are rated as high complexity. Conversely simple physical actions and computer I/O are rated low.

(b) Physical complexity is rated for each action-step. Complex manipulations involving precise movements and detailed co-ordination are rated high, whereas single action-step, discrete actions (e.g. stop/start machine) are rated low.

(c) If the action involves a human computer interface then its usability is assessed. Measures for usability can be acquired from evaluation of users' observed problems. Alternatively, usability can be rated by answering the following questions, which also point to generic requirements for the user interface:

- does a command or function exist for the user to achieve their goal ? If not, then a missing requirements is indicated.
- can the used find the command or action to achieve their goal ? If not, then the menus or the user interface metaphor needs attention

- is it clear how to carry out the action with the computer ? If it is not, then the prompts, cues and user interface metaphor need to be improved.
- can the user perceive and understand the effect of their action ? If not, then feedback should be added or made clearer.

Usability evaluation is a complex subject and the above questions provide the minimum guidance, for more details the reader is referred to (Sutcliffe et al., in press) or (Monk et al. 1993). If a prototype interface does not exist the usability score is set to medium for all potential human computer actions and low (=no design problems) when actions are unlikely to involve human computer interaction.

In use cases, human actions tend to be complex cognitive, and are frequently not analysed in detail, whereas user system interactions are decomposed into lower levels of detail and tend to be simple and physical. It is important, therefore, to ensure human-cognitive actions are explicitly modelled in the use case, e.g. in diagnosing a problem, the user's reasoning is modelled as a complex, cognitive action rather than the simple action 'find fault' that may be the outcome of such reasoning. The BBN is run using ratings of influencing factors representing the user and task/domain that are taken as constants in the application, in combination with the other input variables representing the different environment scenarios. This produces an error probability for each action-step in the scenario script derived from the use case.

3.2 Linking probable errors to Requirements

Slip and mistake errors predicted by the BBN analysis are linked to generic requirements by analysing the type of action, or interaction, for each step in the scenario. To specify the errors that may occur a list of exception types is provided, as summarised in Table IV. They are divided into two groups. First abnormal event patterns drawn from Hollnagel's (Hollnagel 1993) event 'phenotypes' classification, and secondly, information abnormalities that refer to the message contents (e.g. the information is incorrect, out-of-date, etc.). Each exception type is associated with one or more generic requirements that propose high level solutions to the problem.

Table IV Summary of Exception types for errors prone actions and generic requirements to deal with these problems.

Exception type	**Generic Requirements**
event does not happen - omitted	time-out, request resend, set default
event happens twice (not iteration)	discard extra event, diagnose duplicate
event happens in wrong order	buffer and process, too early - halt and wait, too late - send reminder, check task
event not expected	validate vs. event set, discard invalid event
information - incorrect type	request resend, prompt correct type
incorrect information values	check vs. type, request resend, prompt with diagnosis
information too late (out of date)	check data integrity, date/time check, use default
information too detailed	apply filters, post process to sort/group

information too general	request detail, add detail from alternative source

The recommendations or general requirements in the method provide a basis that can be developed with more specific domain knowledge. The exception types are employed with the following heuristics that associate different types of actions, BBN predicted errors and exceptions:

- Input actions which are prone to slip-errors are likely to result in omissions or wrong order exceptions. Besides the generic requirements listed in Table IV, other defences are to provide editing facilities to correct input slips, spelling checkers to detect typographical errors, and validation checks on input fields.
- Input actions with high mistake probabilities have different consequences. Mistakes are not usually detectable in input as these reflect valid events which are not what the user intended. Remediation facilities need to be added to the requirements specification such as undo functions, status indicators for selected options and clear feedback on the user’s progress.
- Output actions are the responsibility of the system so any event omissions point to missing requirements. If an output action is prone to error by the user who receives the data, then slips will be manifest as mis-reads of data, transposition errors or failure to see the output. Defences are to make key data salient by highlighting techniques. A high tendency for mistakes when using or interpreting output will result in incorrect actions or decisions being taken when data is misinterpreted. Requirements to mitigate such problems are to improve data presentation by sorting, grouping, summarising and prioritising information. Information presentation is a complex subject in its own right so the requirements engineer is referred to more detailed sources of guidelines (Galitz 1993).
- Decision actions may contain a mixture of input and output interaction, so many of the above requirements also apply. If user slips are probable, validation checks, editors to correct input, forgiving command and confirmation of dangerous actions, are advised. For mistakes, the decision support system requirements are to check the likelihood of alternative decisions taken against available data, to make dangerous decisions difficult and add warnings. The ability to monitor human decisions depends on the ability to capture and interpret input. If the decision remains within the human domain this may not be possible. Other generic requirements to mitigate the effects of mistakes are to make the assumptions and dependencies of decisions clear and to provide an accurate and realistic model of the decision domain for the user. This improves the user's 'situation awareness' in decision making.

Generic requirements are necessarily high level, but they do draw the requirements engineers' attention to problems which need to be considered. Moreover, the method gives pointers to richer sources of guidelines in the literature. Our work is progressing to improve the quality of requirements advice by investigating classes of user task, so requirement to remediate slips and mistakes can be suggested for diagnosis, planning, scheduling and other generic tasks (see Sutcliffe and Carroll 1998).

4 Case study

The case study is a service engineer support system. We report analysis of the use case 'Diagnose fault & repair' (see Figure 5). Sixteen environmental scenarios were run to model different permutations of engineer training (novice v. expert), their motivation (high or low) and the work situation (good or bad). We report four of the 16 scenarios, an optimistic one with well trained, motivated engineers in a good situation, the inverse pessimistic scenario and two intermediates. Each environment scenario was run against actions in the use case which were classified into four types:

Simple: low cognitive and physical complexity- 10 out of 13 actions
Physical: low cognitive but high physical complexity, action 9, collecting spare parts
Cognitive: high cognitive but low physical complexity, action 11, checking the machine performance
Complex: high cognitive and physical complexity, action 10, repairing the machine.

In all cases the user interface property was set to medium. The results are given in Figure 6 (a-d). Mistake and slip errors will be high for novices in the most pessimistic scenario (Figure 6a), and increasing the novice's motivation makes little difference (Figure 6c). This highlights that the work situation is probably more important for novices. Experts with poor motivation perform better than novices for mistakes but for slips the difference is not great (Figure 6b). When experts are well motivated and the work situation is favourable, in the most optimistic scenario, performance is good and 'low' mistake errors are predicted on all action types apart from complex ones; however, some slips are predicted for all action types, apart from the simple category. When combined with other scenarios which are not illustrated, the main implications of the analysis were to reduce bad situation factors (i.e. give engineers plenty of time for repairs and reduce interruptions), and that investment in training was vital to keep novice mistakes to an acceptable level. Although the BBN analysis may over-estimate the error probabilities, it is the differences between the scenarios that point out issues for the requirements engineer's attention.

The interpretation of the resulting probabilities for errors in the bands 'high, medium, low' has to be calibrated for the domain. In this example low was taken to be an absolute frequency which would not significantly impede normal system operation, i.e. <0.05 % errors per 100 task operations. Medium was estimated as a range from 0.05 to 5% errors/100 task operations with high as >5.0%. The results, shown in Figure 6 (a-d) predict that mistakes will be a problem for novice engineers for actions 9-11, but are also of considerable concern for simple actions. The implication is that thorough training is important. Slips will be a problem for all actions when the situation is bad, as is often the case when engineers work under time pressure with interruptions, or may be under fatigue. This is particularly noticeable for the physical action 9 and the more complex actions, so additional effort needs to be placed into refining requirements to prevent slip errors in actions 9-11 and ways of minimising the physical complexity of searching for spare parts should be found. The generic requirements in Table IV indicate the issues that should be investigated. Clear informal displays for diagnosis are a priority and development of intelligent assistants or tutoring systems may be justified for novice engineers.

Figure 5 Interaction Diagram for the Use Case - Diagnose Fault and Perform Repair

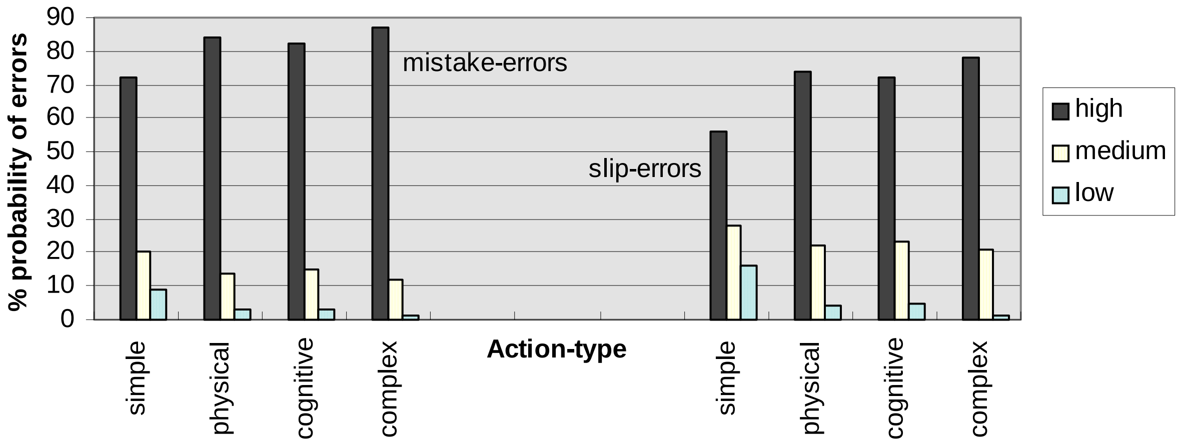

Figure 6a Error probabilities for slips and mistakes, for the Novice engineer scenario with low motivation and a poor situation

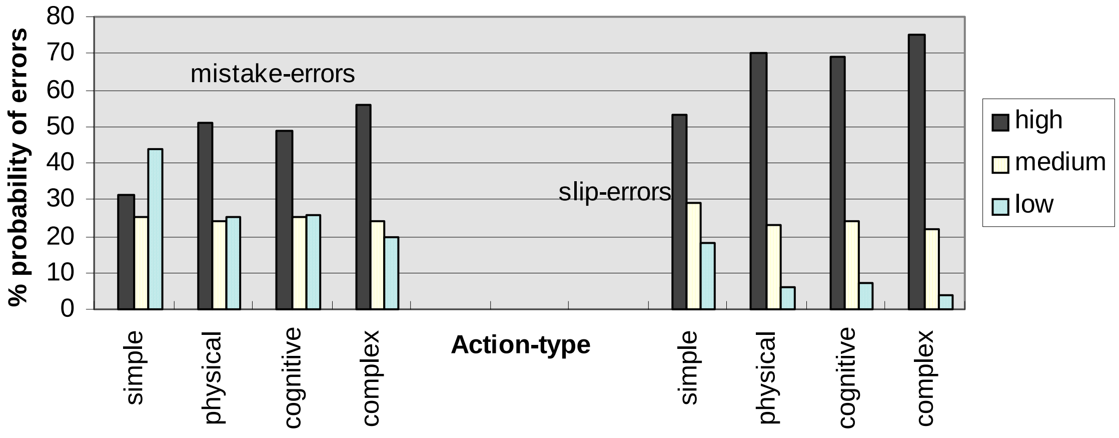

Figure 6b Error probabilities for slips and mistakes, for the Expert engineer scenario with low motivation and a poor situation

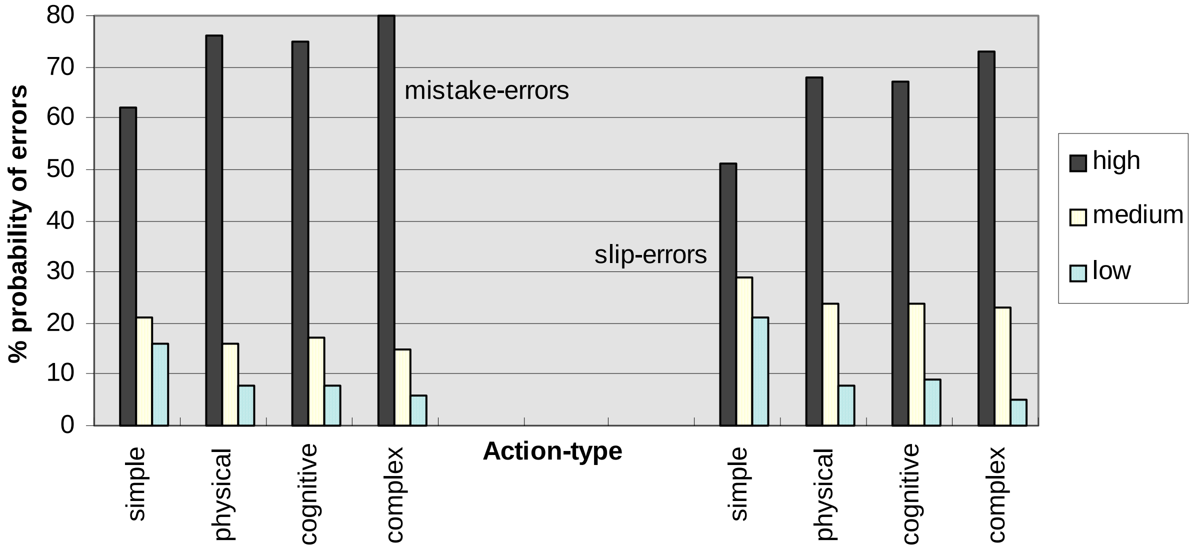

Figure 6c Error probabilities for slips and mistakes, for the Novice engineer scenario with high motivation and a poor situation

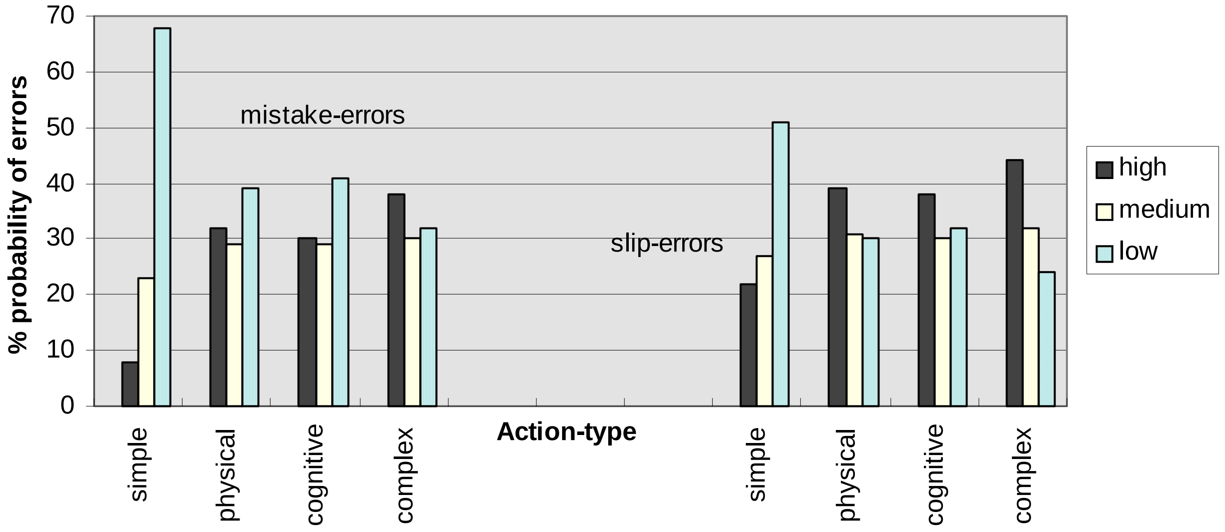

Figure 6d Error probabilities for slips and mistakes, for the Expert engineer scenario with high motivation and a good situation

5 Discussion

This paper has reported modest progress towards improving the quality of requirements elicitation to deal with problems arising from human error. BBNs are a promising approach, nevertheless, there are disadvantages with the set up effort required. Configuring the tables with probabilities is time consuming and the combinatorial explosion of variables makes creating tables with more than 3 input variables per node difficult. The network we have produced is a plausible, although an incomplete model of the psychological influences on human error, and it will benefit from further tuning of the network probability table. However, it is a reusable analytic resource which, we believe, accounts for the more important causal influences on human error.

We have demonstrated an analytic technique to direct requirements analysis effort towards certain actions of a scenario, with the generic requirements that provide advice on solutions for slips or mistakes according to the type of action. However, the quantitative analysis of BBNs has several other benefits; for instance, different scenarios of user interface design quality and user training can be run to inform trade-offs between investing in user training or software development to improve the user interface and hence reduce errors. BBNs highlight the actions in use cases where failure is a critical concern prompting the requirements engineer to invest more time in defining user support and error recovery. Finally, with a prototype design, the BBNs can be used as a quality assurance tool to benchmark designs with a known user population, task and prototype user interfaces to predict the probability of performance errors. In safety critical systems such assessments can provide useful evidence in safety cases.

Quantitative techniques in RE have received little attention to date; however, we note that when trade offs need to be made, metrics have been used by Boehm and In (Boehm & In 1996) in the QAARC tool and in product procurement (Jacobs and Klethers 1994). The approach we have reported provides a more formal means for quantitative assessment of system requirements which, we believe, will return increasing benefits for the effort involved when tuned to particular domains. In our future work we will tune the BBN network by comparing its predictions with data on observed errors and improve the quality of the generic requirements by investigating how predictions can be tied to models of generic tasks.

Acknowledgements

This research has been funded by the European Commission ESPRIT 21903 long term research project 'CREWS' - Co-operative Requirements Engineering With Scenarios. The project partners include RWTH-Aachen (project co-ordinator), City University, London, University of Paris I, France, FUNDP, University of Namur, Belgium.

References

(Bailey 1982) R. W. Bailey, 'Human Performance Engineering', Prentice Hall, 1982.

(Boehm & In 1996) B. Boehm and In Hoh, 'Identifying Quality-Requirement Conflicts', IEEE Software, pp. 25-35, Mar. 1996.

(Fields et al. 1995) R. E. Fields, P. C. Wright, and M. D. Harrison, ‘A Task Centered Approach to Analysing Human Error Tolerance Requirements’, IEEE International Symposium on Requirements Engineering (RE’95), pp. 18-26, 1995.

(Galitz 1993) W. O. Galitz, ‘It’s time to clean up your windows: Designing GUIs that work’, New York, Wiley-QED Applications.

(Hollnagel 1993) E. Hollnagel, 'Human Reliability Analysis Context and Control', Academic Press, 1993.

(Hsi & Potts 1995) I. Hsi and C. Potts ‘Towards Integrating Rationalistic and Ecological Design Methods for Interactive Systems’, Georgia Institute of Technology, Graphics, Visualisation and Usability Centre *Technical Report*, 1-15, 1995.

(Jacobson 1992) I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, *‘Object-Oriented Software Engineering: A Use-Case Driven Approach’*, Addison-Wesley, 1992.

(Jacob & Klethers 1994) S. Jacobs and S. Klethers, ‘Improving communication and decision making within quality function deployment’, Proc. First International Conference on Concurrent Engineering, Pittsburgh, 1994.

(Johnson 1996) C. W. Johnson, ‘Documenting the design of safety critical user interfaces’, Interacting with Computers, vol. 8, no. 3, pp. 221-239, 1996.

(Leveson 1995) N. G. Leveson, ‘Safeware: System Safety and Computers’, Addison-Wesley Publishing Co., 1995.

(Monk et al. 1993) A. Monk, P. Wright, J. Haber, and L. Davenport, ‘Improving your Human Computer Interface’, Prentice Hall, 1993.

(Pearl 1988) J. Pearl, ‘Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference’, Morgan Kaufman, 1988.

(Potts et al. 1994) C. Potts, K. Takahashi and A. I. Anton, 'Inquiry-Based Requirements Analysis', IEEE Software, vol. 11, no. 2, pp. 21-32, 1994.

(Reason 1990) J. T. Reason, 'Human Error', Cambridge University Press, 1990.

(Sutcliffe 1995) A. G. Sutcliffe, 'Requirements rationales: integrating approaches to requirements analysis', in Proceedings of Designing Interactive Systems (DIS'95), ACM Press, New York, pp. 33-42, 1995.

(Sutcliffe 1997) A. G. Sutcliffe, 'A Technique Combination Approach to Requirements Engineering', Proceedings 3rd IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 65-74, 1997.

(Sutcliffe & Carroll 1998) A. G. Sutcliffe and J. M. Carroll, ‘Generalising Claims and Reuse of HCI Knowledge’, People and Computers XIII Proceedings of the BCS-HCI Conference, Sheffield, Eds. H. Johnson, L. Nigay, and C. Roast, pp. 159-176, Springer Verlag, 1998.

(Sutcliffe & Minocha 1998) A. G. Sutcliffe and S. Minocha, ‘CREWS-Scenarios for Acquisition and Validation of Requirements - The Method’, Tutorial, 1998.

(Sutcliffe et al. 1998) A.G. Sutcliffe, N. Maiden, S. Minocha and D. Manuel, ‘Supporting Scenario-based Requirements Engineering’, to appear in IEEE Transactions on Software Engineering, Nov. 1998.

(Sutcliffe et al. in press) A. G. Sutcliffe, M. Ryan, M. V. Springett and A. Doubleday, ‘Model mismatch analysis: Towards a deeper explanation of users’ usability problems’, to appear in Behavioural and Information Technology.

(UML 1997) UML, ’Unified Modelling Language: Method’, Rational Corporation, 1997.