Applying Bayesian Belief Networks to System Dependability Assessment

Martin Neil
Centre for Systems
and Software Engineering
South Bank University
London, UK

Bev Littlewood and Norman Fenton
Centre for Software Reliability
City University
London, UK

Abstract

The dependability of technological systems is a growing social concern. Increasingly computer based systems are developed that carry the potential of increasing catastrophic consequences from single accidents. There have been significant research advances in assessment methods. However dependability assessment of computer systems in practice is still a very uncertain and often ad-hoc procedure. Decision making about system dependability is an uncertain affair and must account of failures in expertise and be capable of integrating different sources of evidence. A more meaningful way of reasoning about systems dependability can be achieved by rejecting current ad-hoc dependability assessment methods and replacing them with the idea of dependability argumentation. Bayesian Belief Networks (BBN's) is proposed as the most promising technology to support this kind of dependability argumentation.

1 Introduction

The dependability of technological systems is a growing social concern as more computer based systems are developed that carry the potential of increasing catastrophic consequences from single accidents [Perrow 84, Mellor 94]. A large amount of effort is spent in improving the dependability of these systems and also in assessing this dependability to the satisfaction of regulatory agencies, insurers, owners and the public at large. For systems where accidents may have serious consequences on human life and health and on the environment, assessors apply methods to help assure that such accidents will happen with acceptably low probabilities. A variety of standards, models and risk assessment methods have traditionally been employed to support these activities.

Despite the considerable success in the use of such methods, systems dependability assessment is still a very uncertain and often ad-hoc procedure, especially where systems contain software. Traditional PRA (Probabilistic Risk Assessment) methods of assessing dependability have tended to concentrate on the dangers presented by physical sources of failure rather than by design faults. Engineers when they are asked to deal with uncertain situations, that require considerable judgement, have tended to rely on strictly "objective" (relative frequency) interpretations of accident likelihoods [Apostolakis 90]. The perception is that it is difficult to apply such frequency considerations to design faults so there has been a tendency to either ignore them or be less rigorous in their assessment.

There are however many standards and models that aim to address the dependability problem. These methods provide a wealth of practical advise on which development approach to choose to achieve dependability but are inadequate when it comes to questions of dependability assessment. Software standards lack engineering rigour and are imprecise [Devine et al. 93, Fenton et al. 94], and do not address the fundamental problem of predicting dependability of software. Alternative approaches to safety assessment for software have been presented borrowing from the ideas of fault tree and hazard analysis [Leveson 95, DEF00-58 95]. There is good reason to believe that these approaches might make us more confident in a system's dependability but their rejection of the probability concepts inherent in the original techniques weaken any attempt to assess dependability. There have been a number of attempts at "characterising" software dependability using quality models [Walters and McCall 78, Basili and Rombach 88]. These approaches propose that if we can decompose dependability properties into supposedly measurable attributes then we can indicate the degree of dependability possessed by the system. This reductionist scheme seems promising but we argue that it is flawed and cannot be used to reason about dependability in a meaningful way.

We recognise that making decisions about system dependability is an uncertain affair and must take into account disparate sources of evidence, the most prominent being expert judgement [Littlewood 93]. However the impact of such evidence is riddled with uncertainty. The relative contributions of different factors are often unknown or controversial and difficult to quantify [Littlewood et al. 95a]. Generally a dependability assessment is obtained by relying on the ability of human experts to integrate the evidence, by applying their own judgement, to obtain important conclusions and make predictions.

It is the contention of this paper that a more meaningful way of reasoning about systems dependability can be achieved by rejecting current ad-hoc dependability assessment methods and replacing them with the idea of dependability argumentation. The use of the word argumentation emphasises the key role concepts like uncertainty, judgement, belief, reasoning and evidence play in our deliberations, whereas assessment is suggestive of dispassionate and objective analysis.

We propose that the most promising technology to support this kind of argumentation is Bayesian Belief Networks (BBNs). BBNs can formalise dependability claims, the models employed to make those claims and the evidence collected in a manner open to independent scrutiny [Littlewood et al. 95a].

2 Assessing Dependability

2.1 Overview

Dependability is defined as "that property of a computer system such that reliance can justifiably be placed on the service it delivers" [Laprie 92]. Depending on the intended application of the system dependability is usually expressed as a number of inter-dependent properties such as reliability, maintainability and safety.

There is as yet no definitive and agreed general definition of what constitutes dependability assessment. Commercial companies engaged in assessment activities and standards bodies that produce systems guidelines and standards have evolved different interpretations of what constitutes the act of assessment. The scope of an assessment can be as wide or as narrow as the customer or regulator desires it. For example a company producing safety critical systems might obtain an assessment of their process against ISO9000 criteria [TickIT 92] or if increased confidence is demanded, might be asked by the regulator to produce a formal analysis of the product. What we can and cannot conclude about dependability from an assessment obviously depends on the type of assessment being undertaken. Indeed whether an assessment would even explicitly consider dependability would depend on the assessment approach. What could we say about the dependability of a system given we know it was developed by a company registered under ISO9000? Such evidence might offer some confidence in dependability but any claim that it was sufficient would clearly be disputed.

Systems assessment can be carried out in a number of ways. Pre-deployment assessment involves evaluating products and processes after the system has been produced, but before deployment. In-process assessment involves executing an assessment throughout the development process in order to identify and prevent problems earlier. The third assessment approach, in-field, assesses a system that is already being operated. Clearly the type and extensiveness of evidence obtainable from an assessment, and indeed the validity of resulting dependability claims, will depend on the type of assessment employed. In retrospective assessments data on actual use, failure rates and user experiences all provide a rich set of diagnostic evidence on the dependability of the product. Pre-deployment assessments would tend to use evidence gathered solely from the testing process, with the accompanying problem that testing data might not be enough for high reliability requirements [Littlewood and Strigini 93]. In-process assessments would tend to make dependability claims based on the application of trustworthy design methods, even though there is little empirical evidence to support such reasoning [Fenton et al. 94].

Differing assessment approaches and dependability claims concentrate on one or more sources of evidence:

- *Development process evidence*: Knowing that systems developers are using "best practice" project management and quality management principles, coupled with a defined life-cycle, usually influences our conviction that the eventual system will be fit for purpose.
- *Product evidence*: Specifications, designs and test plans, etc. also form useful sources of evidence. The quality of these components may be determined, in part, by the effectiveness of the processes put in place. Similarly diagnosis of product attributes, by measurement etc., allows us to form a better picture of potential future operational problems.
- *Resource evidence*: Resources are people, methods, tools and machines. The ability of people is often identified as a major factor in systems assessment. Knowing that developers are skilled and competent usually makes us more confident about eventual system dependability. The capability of methods and tools to handle the problem at hand is also of interest.
- *Evidence about the operating environment*: The general environment within which a project operates will have an effect on dependability at all levels. A lax safety culture or a lack of training can negatively affect dependability.
- *Analogy*: We may have the benefit of historical experience such as past cases. These cases can be called analogues and their relevance will be dictated by the degree of similarity between the present case and historical ones. For instance a developer's track record of building similar systems to the one being assessed will inspire confidence. Likewise if similar design solutions exist, and are being reused, this will often strengthen our conviction.

2.2 Standards

Standards are used to provide the criteria upon which assessment is based. Examples of relevant standards are IEC 65A [IEC65A 91], DIN-0801 [DIN-0801 89], and DEF-STAN 00-55 and 00-56 [DEF00-55 91, DEF00-56 91]. The standards specify lists of criteria addressing a variety of requirements. For example "the product shall be complete, consistent and correct" is an example of a standard type criterion. Many of the criteria contained in standards are ambiguous and difficult to assess objectively [Devine et al. 93].

One standard that is of particular relevance to dependability is the IEC 65A draft standard. IEC 65A consists of two parts. The first part [IEC65 92] addresses general requirements, whilst the second part [IEC65A 91] applies to firmware, operating systems, high-level and low-level languages and PLCs.

IEC 65A predominantly focuses on development process “best practices”, like design methods and testing techniques. For high reliability requirements this seems sensible because we cannot solely rely on evidence from testing alone [Littlewood and Strigini 93]. From the developers point of view this advice is obviously welcome. However, advice on best practice might not be enough. Assessors need methods that can predict the system’s dependability with an acceptable degree of confidence, using test *and* process evidence. It is perhaps unwise to solely rely on the argument “trust us we have used best practice” when we are generally ignorant of how much of an advantage some practices provide.

For dependability assessment the IEC 65A standard requires application of a systematic approach using PSA (Probabilistic Safety Assessment) terminology, the most important of which are described below:

- *Safety*: The expectation that a system does not, under defined conditions, lead to a state in which human life, limb and health, economics or environment is endangered.
- *Safety Integrity*: The likelihood of a programmable electronic system achieving its safety functions under all stated conditions within a stated period of time. Safety integrity levels are defined for the system and for software as ordinal rankings, as given in Table 1.

Level	Description of Software Safety Integrity (qualitative label)	Description of System Safety Integrity (probability of failure for continuously operated system, per hour)
4	Very High	$10^{-8} \leq$ p(failure) $< 10^{-7}$
3	High	$10^{-7} \leq$ p(failure) $< 10^{-6}$
2	Medium	$10^{-6} \leq$ p(failure) $< 10^{-5}$
1	Low	$10^{-5} \leq$ p(failure) $< 10^{-4}$
0	Non Safety-Related	$10^{-4} \leq$ p(failure)

Table 1: IEC 65A Safety Integrity Levels for Systems and Software

In IEC 65A uncertainty is represented by “safety integrity levels” in two different ways. Firstly at the system level the standard uses terms familiar to risk assessors in more traditional hardware engineering fields. For a system it uses the concept of "system safety integrity" to indicate four levels of probability expressed for systems operating discretely or continuously. These system safety integrity levels indicate the chance of future failure, expressed as a frequency. However when it comes to software safety such rigour is disregarded. Software safety integrity is instead expressed in imprecise natural language on a range from high to low and non-safety related. Interpretation of these qualitative terms relies considerably on expert judgement. This would be no cause for concern if software was an insignificant component in modern systems - but it is not, such systems often have hundreds of thousands of lines of complex code whose functionality has come to dominate reliability concerns.

This unequal treatment of software in IEC 65A, and other standards concerned with dependability of computer-based systems, raises some significant questions:

- How confident can the public be that a quantified system level dependability target has been achieved when the potential unreliability of the software that makes up much of the system has not itself been quantitatively assessed?
- What is it about software that makes software developers reluctant to apply quantitative estimates of reliability?
- Why do the role and power of expert judgement receive scant attention in standards?

Despite the lack of a quantified approach to software risks, achievement of reasonable reliability targets is clearly possible. However giving a convincing and open justification of *how* they achieved the target is necessary especially when much of an assessment consists of expert judgement. Currently assessments of computer systems are not exercises in expert accountability and argument, because the standards either do not require it or they do not provide methods for doing it.

An answer to the second question has partly been provided by Littlewood et al. [Littewood et al. 95b]. A significant number of practising engineers do not appear to accept the proposition that software can fail (or even that hardware design is a cause of failure), and hence conclude that probability does not apply. We will not go into the detailed arguments against this position here because they are well documented in [Littlewood et al. 95b]. However it is sufficient to say that expressing design reliability using probability *is* meaningful, whether it is for software or traditional engineering artefacts, such as bridges [Blockley 80].

Given the use of subjective labels and the dependence on expert judgement one would expect standards to offer concrete methods to model, express and validate opinions. Apostolakis [Apostolakis 90] says that engineers doing risk analysis are being asked to deal with situations that require extensive judgement, but they are unaccustomed to mixing objective facts with opinion and so feel the exercise lacks rigour. Additionally those engineers with some grounding in statistics find their frequentist reasoning challenged by the need to apply judgement. There is a need therefore to find ways of making judgement more visible and moving away from strictly frequentist views of uncertainty.

2.3 Quality Models

We can think of software dependability assessment as part of the broader issue of software quality assessment. It is therefore worth seeing what the established software quality models have to offer. Quality models are essentially decompositions of important product properties of software systems into a hierarchical tree-structure. A generic example of this is shown in Figure 1.

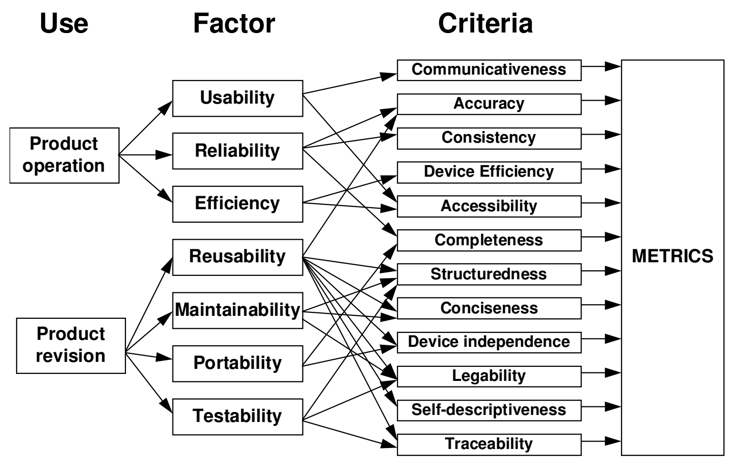

Figure 1: Generic Software Quality Model

The best known approaches are those of Walters-McCall [Walters and McCall 78], and Boehm [Boehm 78] which provide fixed decompositions. Basili and Rombach's GQM (goal, question, metric) model [Basili and Rombach 88] provides a generic, top-down approach. The Walter-McCall model has in fact been adopted as the focus of an international standard [ISO9126 91]. The SCOPE (Software Certification PrOgramme in Europe) project [Rae et al. 95] adopted this standard as its basis for certification. Similarly the IEEE has developed a draft standard based on the same approach [IEEE-1061 91].

Graph-based approaches, such as quality models, may add structure to dependability assessments but by themselves do not, and cannot, go far enough. Little of the work on quality models has concentrated on the semantics of the relationships represented The relationship between criteria, factors and metrics seem to bundle some related concepts together in an ad-hoc fashion. Little or no distinction is made between temporal relations, such as cause and effect The calculus for representing the strength of relations between factors, criteria and metrics is left undecided. This makes reasoning based on quality models difficult and impossible to formalise.

However the popularity of quality models to manage assessment evidence identifies the usefulness and importance of a graph based framework. We will propose one more rigorous than available at present.

2.4 Safety Analysis

From the 1950's, when large complex industrial facilities were first built, studies were performed to determine their safety. These studies had many synonyms, the most well known being PSA (Probabilistic Safety Assessment) and Quantitative Risk Analysis (QRA). Safety analysis studies essentially looked at the degradation of physical components making up industrial plant and the resulting failures. Central to safety analysis is the use of techniques like Failure Modes and Effects Analysis (FMECA), Fault Tree Analysis (FTA), Event Tree Analysis (ETA) and and HAZard Operability studies (HAZOP) which have since become popular in the chemical, transport and nuclear industries.

Quantification of risk and top-down decomposition formed the key principles underlying the success of the safety analysis approach for hardware based systems. Systems analysis allowed detailed and rigorous study of possible failure causes and effects. Quantification made it easier to assess where risk predominated and also helped prioritise fault prevention actions according to the favourability of cost/benefit criteria. In addition to these technical advantages, quantified safety analysis also conferred significant social benefits as well - by comparing the costs of failure with the benefits of operation society had the potential to make informed decisions using rational criteria.

How have these principles and techniques been transferred to the computer domain? It is fair to say that the HAZOP method [DEF00-58 95] and some forms of FTA and FMECA have been widely applied though in a different from that applied for hardware.

The idea of top-down decomposition is easily transferable from hardware to software because modern methodologies encourage block diagram representation of functionality. Where some form of risk quantification has been performed it has been done subjectively, as in the case of Frimtzis et al. [Frimtzis et al. 78] who used a form of FMECA to identify essential system requirements and ranked them according to expected failure.

The fact that software is a logical rather than physical artefact has led some to examine how FTA could be applied to software. Leveson's work in applying Software Fault Tree Analysis (SFTA) to the Ontario Hydro shut-down system is perhaps the most well known [Leveson 95]. Generally, the SFTA approach assumes that the similarity between the logical components that make up an FTA (that is the use of AND and OR connectives to link failure events) and semantics of a program can be usefully exploited. The conjecture is that we can represent a program as a fault tree and demonstrate that certain fault events cannot happen or do not exist by using formal verification techniques. With regard to quantification Leveson says that if SFTA detects a path to an unsound or unsafe output it should be eliminated (designed out) rather than quantified. Eliminating design faults is worthy advice but such reasoning places an absurd requirement on risk quantification. It implies that quantification only targets known problems, when in actuality risk forecasts are primarily concerned with predicting the likely effects of the unexpected or unknown. Unfortunately there is no guarantee of zero defect software, even after known design faults have been eliminated. Applying SFTA would simply help increase our confidence not buy us certainty.

Safety analysis techniques obviously have a strong and useful role in software dependability achievement. The use of such techniques to prove consistency and highlight problem areas would form a valuable input to assessment. The main element that made safety analysis of hardware systems so valuable, however, is missing - rigorous quantification. Existing methods of software safety analysis either prohibit us from expressing our uncertainty or restrict it to imprecise and fuzzy subjective rankings. Even in the case of SFTA where formal verification is used Leveson admits we might still get it wrong when building systems. It is this residual doubt that we need to quantify.

2.5 The Ubiquity of Expert Judgement

We must recognise that making decisions about system dependability is an uncertain affair and must take into account disparate sources of evidence [Littlewood 93]. Such evidence would come from the many models and methods employed during system construction, test and operation. However the impact of such evidence is riddled with uncertainty. The relative contributions of different factors are often unknown or controversial and difficult to quantify [Littlewood et al. 95b]. Consequently engineers fail to combine the evidence in an open and quantified way. Generally a dependability assessment is obtained by relying on the ability of human experts to integrate the evidence, by applying their own judgement, to obtain important conclusions and make predictions. Unfortunately there is ample scientific evidence that human beings are not to be trusted a priori in complex tasks of this kind [Ayton 94]. Unless the experts are known to be reliable (a knowledge that is usually missing for predictions of rare events), means for checking and validating their reasoning, and aiding them in revising their judgements in the light of experience, are absolutely necessary.

It would be rash to view assessment of systems involving new and emerging technologies as simply the accumulation of facts and incontrovertible proofs - but such a view is commonly held amongst many practicing engineers. The limit of scientific and engineering knowledge is a complex issue but we may be able to identify a number of reasons for the misconception that assessments are wholly objective and absolute. Firstly, the traditional perceptions of science and engineering are based on the search for objective truth rather than ascertaining the accuracy of scientific and engineering models. Requirements for strictly objective judgement clearly conflict with the problem of expert fallibility inevitable in dependability assessments. Engineers cannot and do not deal with truth or falsity, the best they can do is to aspire to higher accuracy of predictions. Secondly, because the public is supposedly unable to understand risk issues, responsibility for assessment has inevitably been placed in authorised expertise resident in private companies, institutions and government departments. Institutional expertise of this kind can confer a degree of legitimacy for dependability decisions quite separate from any evidence. In many minds such paternalistic authority can be confused with objectivity and, more worryingly, at its worst can help provide a kind of scientific legitimacy for what may be political decisions [Smith and Lloyd 93].

When we scrutinise the claims made about a system's properties we uncover a set of models and assumptions employed by the expert. Unfortunately these can become out-of-date and dangerously inaccurate if they are not continually questioned and tested against the reality they purport to model. At a personal level, each of us rarely observes and test our own thinking to improve and update the models we base our decisions upon. The degree of uncertainty we will have about the dependability decision will be directly related to how far we trust these models to reflect reality accurately enough for the situation at hand.

These uncertain reasoning mechanisms are implicit in assessment and are uncovered when expert assessors employ particular phrases, such as: *belief*, *judgement*, *inference*, *evidence* and *degree of conformance*. Despite their ubiquity the role of these words in dependability assessment remains obscure. How assessors reason with these words and how they develop assessment conclusions deserves attention.

What do we mean when we say we *believe* something? There are generally two common uses of the word. Firstly we often believe that something is true in an absolute sense. This is most often the case when we have the benefit of hindsight or physical proof. Statements like "a customer requirements elicitation exercise took place" and "a requirements specification document exists" are taken as uncontroversial statements of fact. The second and more common use of the word belief are when we are unsure or are making an uncertain inference. The richness of the natural language reflects the ubiquity of uncertainty. We use words like likely, probable, credible, plausible, possible, chance and odds. Assessors may make statements like "the clients claim that a customer requirements elicitation exercise has taken place but I am not too sure" and "I doubt we can be certain that knowing they have used method X is sufficient to accept that the system is reliable".

Of course, as unavoidable as uncertainty may be, it is natural and beneficial to search for facts. For example it is better to observe first hand what actually went on during a software development process than to receive a third-hand report of what happened from the developer, who may of course be prejudiced. That is why the goal for assessment is to approximate *objective* judgement about a system's dependability.

When we consider assessment reasoning we are really addressing two aspects of *inference*: diagnosis and prediction. Diagnosis involves saying what something is. Stating that "the system software is riddled with GOTO statements "is a diagnosis. On the other hand prediction involves saying something about the future, such as "the MTTF (Mean Time To Failure) of this system is 10^{-6} per hour". Such a prediction would be accurate if it was confirmed by future events.

We have already said that assessors form beliefs on the basis of evidence interpreted through the application of models. We can think of these models containing criteria with which the evidence is compared. The extent to which evidence satisfies the criteria is typically taken to be the *degree of conformance*. We can usefully think of an assessor's degree of conformance as representing one of two things. Firstly it can mean the evidence defines the extent to which some attribute is present - thus being similar to the action of measurement. For example an assessor might measure the size of a module and compare it to some desired limit. The second, and more important, interpretation is that the degree of conformance may mean *degree of belief*. This second interpretation deserves closer scrutiny. Assessors might process a point of evidence and ask what it tells them about some related but uncertain proposition or event. He uses the evidence from one entity, not to tell him about that entity, but to infer or predict the state of some other entity. So knowing that a program had low McCabe metric values [McCabe 76], according to some criterion, may prompt assessors to believe the system has high maintainability.

Expert judgement is a central component of any dependability assessment exercise, especially given the dearth of empirically tested models in software engineering. Such expert judgement pervades all aspects of reasoning in dependability assessment yet its ubiquity stirs little debate. At worst such judgement may wrongly be assumed to be scientific and hence trustworthy, so few demands are made to make assessment more accountable and visible. Nevertheless, some judgements have been shown after the event to be “good” ones - the problem is to know *before* the event that they will be.

3 Challenges in Dependability Assessment

From the preceding analysis it should be clear that a number of serious challenges can be levelled at the ways we currently assess complex computer-based systems. We list them here with the claim that the activity of systems dependability assessment will only become accurate, meaningful and accountable when the research community has accepted and resolved them.

1. *Representing Judgement*: The community needs to find a way of representing human judgement and facts from diverse sources of evidence in a predictive framework. A safety case or dependability case should be presented as an argument rather than as a statement of incontrovertible truth, with means for checking and validating judgement. Use of an uncertainty calculus is essential if judgement is to be made visible and accountable.

2. *Empirical Foundations*: Most ad-hoc dependability approaches have no method for determining whether or not the resulting risk forecast is a successful predictor of system behaviour. The use of imprecise terminology makes confirmation or falsification of such forecasts impossible. A forecast that "the system is on the whole fairly risk-free with respect to the majority of usage situations" leaves the terms "on the whole" "majority"

and "risk free" undecided. We need a way to determine from an observed set of accidents or incidents whether this prediction was indeed accurate. A crucial prerequisite of a risk management system is that it can learn from mistakes and use this feedback to improve performance over time. Information from failures and incidents provides an index of success for risk management. Without numerical counts of such things we cannot learn from mistakes and specify process improvements.

3. *Economic Justification*: Systems development and use is a careful balancing act between safety and productive performance. Central to this task is cost/benefit analysis. Because of their very imprecision, qualitative safety and risk forecasts cannot be manipulated in costs/benefit analysis leaving the quantitatively expressed production figures to predominate. This one-sidedness could lead to a dangerous erosion in the capability of decision making mechanisms to adequately address safety matters.
4. *Evidence Integration*: There is a need to develop an integrating framework where each of the above challenges can be met. We should aim to link dependability claims, engineering models, expert judgement and uncertain evidence in a rigorous framework that promotes accountability.

4 Bayesian Probability and Belief Networks

4.1 Background

We contend that Bayesian Belief Networks (BBNs) offer the most promising technical solution to many of the above challenges. Existing BBN technology, based on probability and decision theory, can potentially improve assessment reasoning under uncertainty. BBNs enable us to analyse and formalise, rather than just mimic, expert judgement and engineering models.

BBNs are known under various synonyms as Graphical Probability Networks, Belief Networks, Causal Probabilistic Networks, Causal Nets and Probabilistic Influence Diagrams. They have been used in a wide variety of applications [Pearl 88] as an appropriate representation of probabilistic knowledge. They have been applied in medicine, oil price forecasting and diagnosis of copier machine faults. In software engineering they have been used to diagnose and find faults during debugging of the Sabre airline reservation system [Burnell and Horovitz 95]. Probably the most famous application is the Answer wizard in Microsoft's Office 95 products, where they are used for automated learning for custom-tailoring help software to a user's work patterns and preferences.

The world-wide interest in BBNs has produced many implementations, most notable amongst these is the HUGIN tool [HUGIN 94] based on the award winning theoretical work of Lauritzen and Spiegelhalter [Lauritzen and Spiegelhalter 88].

A BBN is a directed graph used to represent probabilistic relationships amongst events or propositions. Each node represents a set of alternative events of interest, the arcs represent the probabilistic conditioning of a node's value on that of other nodes. Associated with each node is a conditional probability table that shows the probability of that node state being true given the events represented by the parent nodes. When a node has no parents the conditional probability table is simply the prior distribution. The BBN directed graph with its conditional probability tables specifies a joint marginal distribution of all the events. When the actual state of a node is observed the probabilities of event states are calculated by propagating the "new" knowledge along the arcs in the graph. In this way the probabilities change as our uncertainty and knowledge changes.

4.2 Probability Theory

We need some, preferably formal, way of representing expert judgements about uncertain events. There is a formal theory of probability called Bayesian, or subjective, probability that allows us to do so [de Finetti 74]. In Bayesian probability theory a probability number, lying in the range zero to one, is used to represent an individual's degree of belief in the truth of an event or proposition. Expressed mathematically as $p(A \mid H)$ it is interpreted as the degree of belief in the truth of A given that H is known to be true.

This definition might surprise some readers, but it is objective and scientific because it accurately represents an individual's belief. More specifically, it can be shown to be a necessary consequence of adhering to certain intuitively plausible prescriptive notions of "rationality" - for example that a rational individual should not have circular pair-wise preferences. Of course, the validity of a belief can be questioned, but as a representation of an individual's stated belief a probability number cannot be disputed. It has been claimed this definition is the only way to describe uncertainty and all alternative descriptions are unnecessary, even the frequentist interpretation [Lindley 87]. We would not wish to go so far, but think it attractive that the frequentist interpretations, so popular in risk and safety analysis, can be expressed more conveniently and meaningfully as part of a Bayesian framework. In those circumstances where there is a believable argument based upon frequentist principles - such as the probability of Heads for tossing a fair coin - it would be rational for ones subjective probability to coincide with the frequentist one. Even apparently simple cases like this pose problems, however: how are we to know the coin is fair? We cannot easily execute a number of repetitive trials of coins drawn at random from the local bank in order to determine the proportion of fair ones. Bayesianism offers a more practical resolution, that allows us to express our belief that the coin is fair as a "subjective" probability.

Using probability we can compare beliefs, we can share them and reuse them, and we can build consensus. The narrow frequentist interpretation denies the subjectivity of probability and would restrict decision making to repetitive, statistical events - despite the fact that a perfectly repeatable event is an idealisation rather than a representation of reality [Watson 94].

The key to utilising the above probability concepts to process uncertain evidence is given by Bayes' theorem:

$$p(A/B) = \frac{p(B/A)p(A)}{p(B)}$$

This states that the probability of A, given we know B, is equal to the probability of B, given we know A, multiplied by the ratio of probabilities of A and B.

The importance of the theorem is that it connects two entirely different probabilities concerning the same two events, namely P(A | B) and P(B | A). In the former B is part of the evidence and A is uncertain. In the latter the roles of A and B are reversed allowing Bayes' theorem to propagate the effects of added evidence, through a network of variables in any direction. Unfortunately natural language is often capable of hiding the differences between these two notions.

4.3 Example of a Bayesian Belief Network

Consider the following simple everyday reasoning problem that we can solve using a BBN. A colleague is late for work and you suspect that he has either slept-in or there has been a failure in the London underground system. From experience you know that the probability of failure on the London underground system is high, p(T = true) = 0.3. You also believe that because your colleague is a fastidious timekeeper he is unlikely to sleep in, p(Slept-in = true) = 0.05. You have an important business appointment and had planned to travel there by Tube. Knowing that your colleague is late may make you more likely to believe that the London Underground system has failed, but by how much?

Using this information we can construct a Bayesian network, portrayed in Figure 2, with three nodes, L =Late, T = Tube system failure and S = Slept-in, each with two states: true and false.

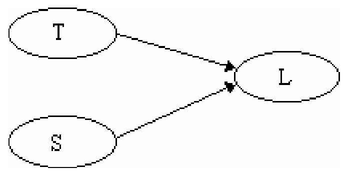

Figure 2: Example BBN

Expressed mathematically the marginal distribution for the example network is:

$$p(L) = p(S)p(T)p(L/S,T)$$

This states that the probability of L is calculated from knowledge about the states of variables S and T and the likelihood of L given these states. This of course assumes that it is reasonable to suppose that S and T are independent.

For each of the nodes in the BBN a conditional probability table is needed. For each of the combinations of the node states we would estimate a probability of that combination being true. Conditional probability values for nodes T and S are given in Table 2 and conditional probability values for node L are given in Table 3:

	true	false
p(T)	0.30	0.70
p(S)	0.05	0.95

Table 2: Conditional probability for T and S

p(L / T, S)	T = true	T = false
S = true	(0.9, 0.1)	(0.7, 0.3)
S = false	(0.6, 0.4)	(0.1, 0.9)

Table 3: Conditional probability table for L

We can calculate the probability that T is true given L is true using Bayes' theorem.

$$p(T=t\,/\,L=t)=\frac{p(L=t\,/\,T=t)\,p(T=t)}{p(L=t)}$$

From Table 2 p(T = true) = 0.3. We can calculate p(L = true) over all possible values of S and T from the marginal distribution and the conditional probability tables, as follows:

$$\begin{aligned}&p(L=t)=p(S=t)p(T=t)p(L=t\,/\,S=t,T=t)+\\&p(S=t)p(T=f)p(L=t\,/\,S=t,T=f)+\\&p(S=f)p(T=t)p(L=t\,/\,S=f,T=t)+\\&p(S=f)p(T=f)p(L=t\,/\,S=f,T=f)+\\&=0.05(0.3)(0.9)+0.05(0.7)(0.7)+0.95(0.3)(0.6)+0.95(0.7)(0.1)\\&=0.2755\end{aligned}$$

Next we need to calculate the likelihood p(L = true | T = true) using the fact that:

$$\begin{aligned}
& p(L=t \,/\, T=t) \\
&= p(L=t \,/\, T=t, S=t)p(S=t) + p(L=t \,/\, T=t, S=f)p(S=f) \\
&= 0.9(0.05) + 0.6(0.95) \\
&= 0.615
\end{aligned}$$

Finally we can calculate p(T=t | L = t) as:

$$p(T=t \,/\, L=t) = \frac{0.615}{0.2755}(0.3) = 0.6697$$

Knowledge about your colleague's lateness has caused the revision, from an initial 0.3 degree of belief that the London Underground system has failed, to 0.67 degree of belief. Travel to the business appointment using London Underground would be ill-advised.

From this very simple example we can see that the BBN formalism offers a number of benefits. Firstly, its graphical nature makes for a powerful, intuitively appealing, knowledge acquisition device. Secondly, the quick and easy propagation of "facts" through the graphs makes it easy to check for coherence and perform "what if" analyses. The greatest benefit is that automatic methods are now available that can be used to propagate the evidence without recourse to tedious manual methods, even for large graphs.

5 Moving from Dependability Assessments to Arguments

5.1 Bayesian Approach to Argumentation

BBNs provide an uncertainty calculus and graphical framework by which dependability arguments can be expressed. The nodes in a BBN can be used to represent the types of evidence required in a dependability argument and the probabilities can model expert opinion or statistical data. Furthermore the maturity of Bayesian probability gives the added benefit of being able to update the BBN to accommodate the accumulation of statistical evidence, for example from repeated test runs. Of great benefit is the capability to develop arguments in two stages: identifying sources of evidence and their interrelationships can be separated from the quantification.

Moving from current ad-hoc approaches towards BBN-based argumentation places expert judgement and the uncertainty of knowledge at the centre of dependability decisions rather than on the fringe. In doing so the opinion of the expert is placed at the centre of scrutiny and frequency data is perceived as playing a supporting role. Such an approach does not imply that there is a single "correct" argument for dependability, rather its use should facilitate an open dialogue and encourage all parties in a dependability decision to develop their own BBNs. Of course such a process would not guarantee safer systems but it would lead to greater accountability and the open exchange of experience.

In Figure 3 we present a model of the structure of dependability arguments. This structure is intended to act as a guide to linking attributes of interest, the relations between them (causes and consequences) and, at the centre, the dependability properties of a system (reliability, safety, etc.). Application of the model would require the use of argumentation templates, for each of the dependability properties, that could be tailored to the particularities of the system in question.

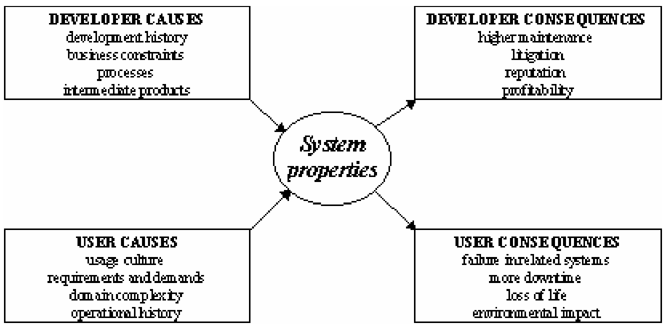

Figure 3: Structuring Dependability Arguments

When assessors are predicting systems dependability they are interested in those factors, under the developers control, which cause the developed process to produce a good system. In Figure 3 these are called the *developer causes*. Effort is concentrated on the quality of the intermediate products, the people and resources used in their production and the processes and methods applied. These in turn are known to be influenced by business constraints, such as budget restrictions, company culture and general engineering capability. Similarly assessors need to investigate the influence the system user will have on system dependability. In Figure 3 these are termed the *user causes*. Domain complexity, the accuracy of requirements and business constraints may have a tangible and immediate effect on dependability. Also the culture, operational history and capability have underlying causal effects on both requirements and operation. For instance a lack of a safety culture could lead to system misuse and accidents. The consequences of system use are the most interesting from the end-users point of view. A dependable system should operate well in its immediate environment. However operators, resources and business processes can be negatively affected by poor system performance. These factors can be classified as *user consequences*. Assessors have to consider the consequences of the system's operation. For example on a fly-by-wire aircraft the user consequences of system failure may have a knock-on effect on other systems and the decisions made by the pilot. Disruption of business may result from undependable operation. Such disruption would most clearly manifest itself as lost profits, higher costs and lower profitability. We can also identify *developer consequences*. The most obvious impact of a poor system dependability will be in maintenance. More serious problems might result in litigation and a loss of reputation. The ultimate consequences would be a decrease in staff morale and poor performance in business operations.

5.2 Dependability Argument Templates

System dependability arguments can now be composed of inter-connected argument templates. Such a template would consist of a BBN developed to predict a single dependability property of the system. Here we present two BBN templates for reliability and correctness.

The templates themselves are simply expositions of the arguments that would be employed to predict or infer dependability and should not be seen as definitive or correct models. For example an alternative to the correctness template has been developed to concentrate on fault density prediction.

5.2.1 Correctness

Users of computer systems want obviously them to be functionally correct. Assessors must therefore attempt to answer two questions when trying to predict correctness before the system has been developed:

- How likely is it that faults have been introduced into the software?
- If there were faults what is the chance that they were removed?

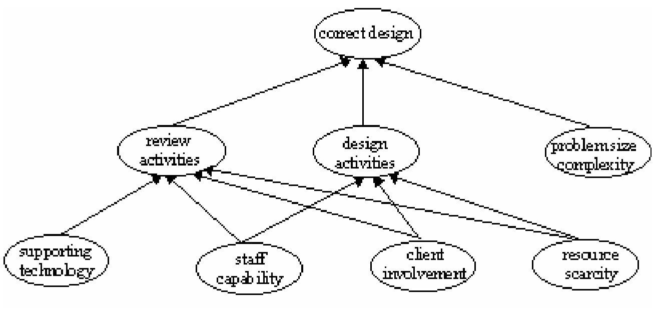

Figure 4: Correctness BBN Template

In order to find some way of predicting the correct design node we would first look at the design and review activities in the software development process. Thus the nodes review activities and design activities enter the BBN given in Figure 4. These two nodes are believed to influence the chance of obtaining a correct design. If a design activity is performed well we would expect that due care would have been taken not to introduce design faults into the system. Good review activities would help find any design faults that slipped through the design process.

The ability to design fault-free systems will be largely influenced by the complexity and size of the problem. So the size and complexity of the application domain form another node predicting correctness.

Now we can take a deeper look at a project and ask what causes good design and review activities? Good review activities are caused by customer involvement, good supporting technology and capable staff. If customer representatives carry out reviews during development, or take part in development review meetings, we would be more likely to expect that faults would be detected. Appropriate supporting technology, such as proof checkers and formal review procedures, would also positively influence our belief about the effectiveness of reviews. Capable staff play a strong role because they influence review and design activities. Any pressure on resources may have a negative effect on the quality of the design and review activities. A consequence of good review activities would be that faults were picked up early in the life-cycle.

If a system specification is functionally correct we could expect a number of consequences. Firstly any implementation made from the specification would be less likely to contain faults. Secondly, because there are fewer faults to be triggered during operation, high operational reliability should result. An assessor might want to predict these consequences for reliability from a diagnosis or prediction of correctness. These consequences are relevant to the next template.

5.2.2 Reliability

An unreliable system can lead to frequent maintenance, the production of work-around procedures to ensure that business processes can operate in some way, and of course failures in other systems. Figure 5 shows a BBN with these factors as consequences of low system reliability. Some of these factors have a knock on effect. Frequent maintenance can cause high downtime that is likely to make for dissatisfied users. Of course there may be other explanations for some of these consequences. Firstly there may be other sources of failure that are causing high downtime and work-around procedures. For example the successful execution of application software is dependent on operating systems software. If this is unreliable then downtime of application and operating systems will result. Poor user training can result in dissatisfaction, high downtime and the production and use of work-around procedures. After all, a system may be very reliable but if it is not being used correctly problems are bound to occur.

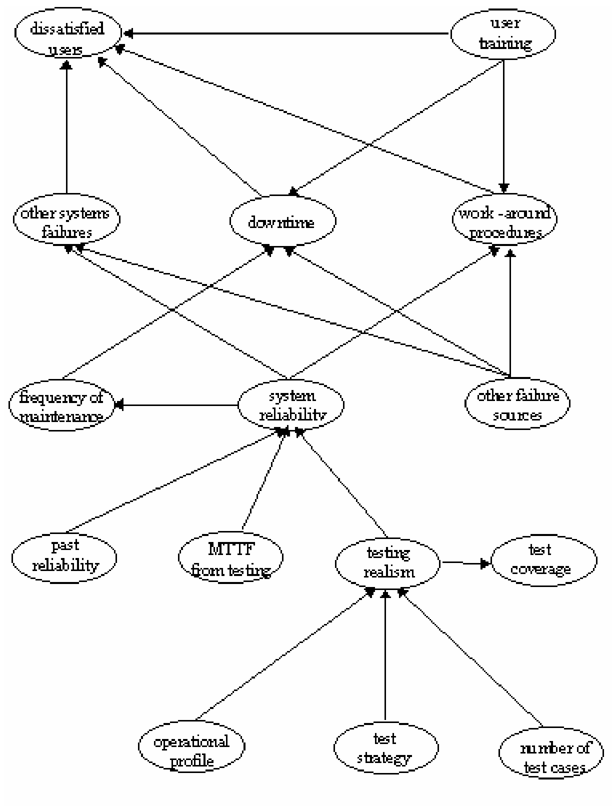

Figure 5: Reliability BBN Template

We can predict system reliability during testing using formal techniques, by the use of reliability models, or by informal judgement. We concentrate on a judgemental approach here. Figure 5 shows the causal and influential factors for system reliability from the testing processes, past reliability of duplicates, MTTF during testing and realistic usage scenario. There is one factor deserves particular attention - the reliability data available from use of duplicates copies the software. For instance if a developer has a historical system that achieved 10^{-7} per hour then we might be willing to consider the argument that the new system will achieve 10^{-7} per hour. Of course such reasoning would be very weak when making claims about a novel system, changed version, or use in a new operational environment. However we could consider making trade-offs along the following lines. If a system has demonstrated high reliability in its original operating environment and is being reused in a new working environment then full testing may not need to be performed. We must however be careful: such reasoning would be valid only if the new working environment was sufficiently similar to the old one and the software was indeed a duplicate. With some provisos we can use historical reliability knowledge to compensate for full testing information.

6 Conclusion

We have argued that, despite the considerable success in the use of current assessment methods, dependability assessment of computer systems is still a very uncertain and often ad-hoc procedure. We have provided an overview of current assessment approaches, including standards, quality models and software safety analysis. Examining each of these has led to the conclusion that these methods do not allow quantitative expressions of dependability in the rigorous way we desire. Where quantification is done, it relies on either a frequentist interpretation that disallows expression of expert judgement or is expressed as imprecise rankings. The frequentist requirement contrasts with the predominant role of expertise in systems assessments.

From an analysis of current practice a number of challenges are evident. We have highlighted the need to express expert opinion, employ proper empirical methods, apply costs/benefit analysis and develop an integrating framework as major challenges to the research community.

In an attempt to meet these challenges a move from ad-hoc assessment to argumentation is advocated. Such a move would exploit the more robust Bayesian interpretation of probability, thus reconciling judgements about single events with statistical data. Finally, Bayesian Belief Networks (BBNs) are proposed as the most promising technology to support this kind of dependability argumentation.

We need to solve a number of outstanding issues to apply in practice dependability argumentation, through the use of BBNs. Firstly, the development of a template-based toolset is needed to support large scale assessments. Secondly, the number of probabilities needed for BBNs, even in small scale argumentation exercises, can be extremely large. Work is currently underway to resolve this problem by automatically generating conditional probability tables using Beta distributions.

Acknowledgements

This work was supported by the DATUM (Dependability Assessment Through the Unification of Measurable evidence) and PROMISE (PROduct Measurement for Integrity and Safety Enhancement) projects. DATUM and PROMISE are funded by the Safety-Critical Research Programme, supported by the DTI and EPSRC (DATUM grant number: GR/H89944, PROMISE grant number: J18880). The DATUM consortium is composed of the Centre for Software Reliability and the Centre for Human Computer Interface Design at City University, the Department of Computer Science at Royal Holloway and Bedford New College, and Lloyd's Register as an industrial "uncle". The PROMISE consortium consists of the Centre for Systems and Software Engineering, South Bank University, and Philips Medical Systems - Radiotherapy as an industrial "uncle".

The authors thank Bob Malcolm for his encouragement and Jennie Rogers for reviewing the paper.

References

[Apostolakis 90] Apostolakis G. The Concept of Probability in Safety Assessments of Technological Systems. Science, vol. 250, December 1990.

[Ayton 94] Ayton P. On the Competence and Incompetence of Experts. In Expertise and Decision Support (Ed. G. Wright and F. Bolger), pp. 77-105, Plenum Press, 1994.

[Basili and Rombach 88] Basili V. and Rombach D. The TAME project: Towards Improvement-Orientated Software Environments. IEEE Transactions in Software Engineering, Vol. 14, No 6 January, pp. 758-773, 1988.

[Blockley 80] Blockley D. I. The Nature of Structural Design and Safety. Ellis Horwood Ltd, 1980.

[Boehm 78] Boehm B.W. Characteristics of Software Quality. TRW Series of Software Technology 1. North-Holland Publishing Company, 1978.

[Burnell and Horovitz 95] Burnell L. and Horvitz E. Structure and Chance: Melding Logic and Probability for Software Debugging. Communications of the ACM, vol. 38, no.3, 1995.

[de Finetti 74] de Finetti B. Theory of Probability, Volume 1. John Wiley & Sons, 1974.

[DEF00-55 91] The Procurement of Safety Critical Software in Defence Equipment Part 1: Requirements Part 2: Guidance Interim Defence Standard, no 00-55 Issue 1. Ministry of Defence, Directorate of Standardisation, Kentigern House, 65 Brown Street, Glasgow, G2 8EX, UK, 1991.

[DEF00-56 91] Hazard Analysis and Safety Classification of the Computer and Programmable Electronic System Elements of Defence Equipment. Interim Defence Standard no 00-56 Issue 1. Ministry of Defence, Directorate of Standardisation Kentigern House, 65 Brown Street, Glasgow, G2 8EX, UK, 1991.

[DEF00-58 95] A Guideline for HAZOP Studies on Systems Which Include A Programmable Electronic System. Interim Defence Standard no 00-58 Issue 1. Ministry of Defence, Directorate of Standardisation, Kentigern House, 65 Brown Street, Glasgow, G2 8EX, UK, 1995.

[Devine et al. 93] Devine C. Fenton N and Page S. Deficiencies in existing software engineering standards as exposed by SMARTIE. In Safety Critical Systems. (Ed. Redmill F and Anderson T.) Chapman and Hall, pp 255-272, 1993.

[DIN-0801 89] Preliminary Standard, DIN0801 DIN-V/VDE 0801. Principles for Computers in Safety-Related Systems, 1989.

[Fenton et al. 94] Fenton N.E. Pfleeger L and Glass R. Science and Substance: A Challenge to Software Engineers. IEEE Software, pp. 86-95, July 1994.

[Frimtzis et al. 78] Frimtzis A. Lipow M. and Reifer D.J. Software Failure Modes and Effects Analysis. Proceedings of Industry/Space and Missile Systems Organisation Conference and Workshop on Mission Assurance. Los Angeles, California, April 1978.

[HUGIN 94] HUGIN Expert A/S. P.O. Box 8201 DK-9220 Aalborg, Denmark.

[IEC65 92] Functional Safety of Programmable Electronic Systems: Generic Aspects. International Electrotechnical Commission. Technical Committee no. 65, Working Group 10 (WG10), type IEC no 65A (Secretariat), February, 1992.

[IEC65A 91] Software for Computers in the Application of Industrial Safety Related Systems. International Electrotechnical Commission Technical Committee no. 65 Working Group 9 (WG9) IEC no 65A (Secretariat), Version 1.0 August 1991.

[IEEE-1061 91] IEEE Standard 1061: Software Quality Metrics Methodology, 1991.

[ISO9126 91] ISO (International Organisation for Standardisation). Information Technology - Software Product Evaluation - Quality characteristics and guidelines for their use - ISO9126. 1991.

[Laprie 92] Laprie, J.C. (Ed.) Dependability: Basic Concepts and Terminology. IFIP WG 10.4 Dependable Computing and Fault Tolerance. Springer-Verlag, Vienna, 1992.

[Lauritzen and Spiegelhalter 88] Lauritzen S. L. and Spiegelhalter D.J. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion). J. R. Statis. Soc. B, 50, No 2, pp 157-224, 1988.

[Leveson 95] Leveson N.G. Safeware: System Safety and Computers, a guide to preventing accidents and losses caused by technology. Addison-Wesley Publishing company, 1995.

[Lindley 87] Lindley D.V. The Probability Approach to the Treatment of Uncertainty in Artificial Intelligence and Expert Systems. Statistical Science, Vol 2, No 1, pp17-24, 1987.

[Littlewood 93] Littlewood B. The need for evidence from disparate sources to evaluate safety. In directions in safety critical systems - proceedings of the first safety-critical systems symposium (eds. F. Redmill and T. Anderson). Springer-Verlag, London, 1993.

[Littlewood and Strigini 1993] Validation of Ultra-High Dependability for software-based Systems. Communications of the ACM, 36, 11, pp.69-80, 1993.

[Littlewood et al. 95a] Littlewood B. Neil M. and Ostrolenk G. Uncertainty in Software-Intensive Systems. Accepted for publication in High-Integrity Systems Journal, 1995.

[Littlewood et al. 95b] Littlewood B. Neil M. and Ostrolenk G. The Role of Models in Managing Uncertainty of Software-Intensive Systems. Accepted for publication by Reliability Engineering and System Safety Journal, 1995.

[McCabe 76] McCabe T.J. A Complexity Measure, IEEE Transactions In Software Engineering, vol. 2, no 4, p 308 - 320, 1976.

[Mellor 94] Mellor, P. CAD: Computer Aided Disasters. High Integrity Systems. Volume 1, number 2, pp. 101 -156. 1994.

[Pearl 88] Pearl J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman, 1988.

[Perrow 84] Perrow C. Normal Accidents: Living with High-Risk Technologies, Basic Books, 1984.

[Rae et al. 95] Rae A. Robert P. and Hausen H. (Eds.) Software Evaluation for Certification: Principles, Practice and Legal Liability. McGraw Hill, International Software Quality Assurance Series, London, 1995.

[Smith and Lloyd 93] Smith D. and Lloyd. D. Wither Objectivity: Technocracy and the Social Construction of Risk. In proceedings of the safety and reliability society symposium on engineers and risk issues (Ed. Cox R.F. and Watson I.A.), Altrincham, October 1993.

[TickIT 92] Guide to Software Quality Management, System Construction and Certification using ISO 9001/EN 29001/BS 5750, Issue 2.0. DTI, available from TickIT Project Office, 68 Newman Street, London W1A 4SE, 1992.

[Walters and McCall 78] Walters G.F. and McCall J.A. Development of Metrics for Reliability and Maintainability. Proceedings Annual Reliability and Maintainability Symposium, IEEE, 1978.

[Watson 94] Watson S. R. The meaning of Probability in Probabilistic Safety Analysis. Reliability Engineering and System Safety, 45, pp.261-269, Elsevier Science Ltd, 1994.