

# Enhancing Software Reliability Estimation Using Bayesian Networks and Fault Trees

Ganesh J Pai, Joanne Bechta Dugan  
Department of ECE, The University of Virginia, VA, 22903  
{gjp5j, jbd}@ee.virginia.edu

## 1 Introduction

A majority of mission-critical or safety-critical systems are complex computer-controlled systems, which are increasingly relying on software to provide added functionality. Although improved design techniques and development methodologies have reduced the software engineering cycle time, the complexity of software design and analysis has increased. The probabilistic assessment of software safety and reliability is important due to the pervasive use of software in such systems and since more system failures are being attributed to software failures. Moreover, software reliability is an important metric in determining overall system reliability. However, this is a difficult task involving uncertain predictions. We describe a novel approach to estimate software reliability by using bayesian belief networks (BBNs) and fault trees. The goal is to provide a more realistic estimate of software reliability that accounts for the causal dependence between processes in the software engineering cycle and software reliability.

## 2 The Proposed Methodology

Although software standards and software engineering processes guide the development of reliable or safe software, mathematically sound conclusions that quantify reliability from conformity to standards are hard to draw. Often, designers use experience-based beliefs and engineering judgment to state that the deployed software is reliable enough. Other quantitative models, such as software reliability growth models or architecture-based reliability models do not consider the effect of software engineering processes on software reliability. We attempt to develop a technique to relate software engineering processes and existing analysis methods to provide a more realistic estimate of software reliability. Further, we propose to combine BBNs and fault trees to improve software reliability analysis in complex systems.

### 2.1 Combining BBNs and Fault Trees

Prevalent approaches to software reliability modeling consider software as a monolithic component and determine reliability based on its interactions with other components in the system [1]. Newer approaches include a bayesian approach [2] for estimating software reliability of component based systems at design time, and a qualitative measure of reliability as a belief in fault freeness [3]. BBNs are a mathematical formalism allowing reasoning under uncertainty, and provide a robust probabilistic framework to

evaluate the impact of evidence on uncertain outcomes. Graphically, they are causal directed acyclic graphs with nodes and edges. The nodes represent random variables with probability distributions, while edges represent weighted causal relationships between the nodes. Fault trees are versatile graphical and mathematical models that logically describe a combination of events leading to system failure. BBNs have root nodes with marginal prior probabilities, while fault trees have basic events with a probability of occurrence or a failure rate. We propose to combine BBNs and fault trees hierarchically, where fault trees are used to compute root node probabilities for BBNs and BBNs are used to compute the basic event probabilities in fault trees.

An immediate advantage is the ability of the BBN to analyze variables with multiple states. Another powerful advantage is the unification of expert opinions and qualitative analyses from BBNs with quantitative results obtained from fault trees. This is helpful when we analyze component based software reliability, where we rely on beliefs and prior knowledge about the operation of the component and the software engineering process to deduce reliability. Besides this, BBNs assist in the diagnostic process because of the ability to compute posterior probabilities given some evidence.

### 2.2 Relating Software Engineering Processes with Software Reliability

The visionary objectives of this work are to incorporate software standards and software development procedures into a BBN to qualify software engineering processes. Thereafter, overall software reliability is estimated as a function of the qualitative results of the BBN and the reliability obtained from current approaches.

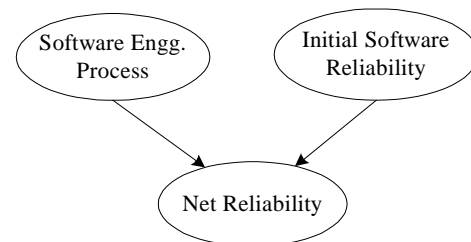


Figure 1. BBN relating Software Engineering Processes with Software Reliability

Fig. 1 shows one of the three proposed models that causally relate processes in the software engineering cycle with the initially computed reliability. This generates a new

overall reliability that accounts for the dependence between software engineering processes and software reliability. Besides this, we develop two more models: one where there are only two causally related nodes representing the process and the reliability; the second combines the former model and the model from Fig. 1 *i.e.* the initial software reliability is causally dependent on the process as well. Suppose we assume no dependency, the overall reliability should be the initial value. Also, given that the process is excellent, the reliability cannot be any more than this value and therefore this is the upper limit on the reliability. However if there is any dependency, then the reliability should be reduced for any other level of the process. Intuitively, this is what we expect, and in the subsequent sections, we show that the results of our analysis confirm our expectations.

### 3 Reliability Analysis of an NVP System

As a preliminary attempt, software quality is causally related to software reliability in a system employing the NVP ( $N = 3$ ) style of software fault tolerance. Fig. 2 shows a belief bar representation of our BBN model for the NVP system. We assumed a component-based architecture for the software modules. The node *ArchitecturalReliability* represents the reliability of the software computed using an architecture-based approach. The node *QualityProcess* represents the software quality process. The quality process is a BBN in itself, representing experience gained through the licensing process of a programmable system in a nuclear power plant. This is appropriate because nuclear power plants frequently use NVP for software fault tolerance. Both the process and its BBN were adopted from [3]. Briefly, this BBN related subprocesses such as quality assurance standards, quality assurance policy and quality control processes, among others. *NetReliability* represents the target node of the BBN, which gives us the new estimate of software reliability for the module. Fig. 2 shows the different states that the nodes can assume. We assumed weights for the edges of the BBNs for what we “believed” would be the influence of each node on the net reliability.

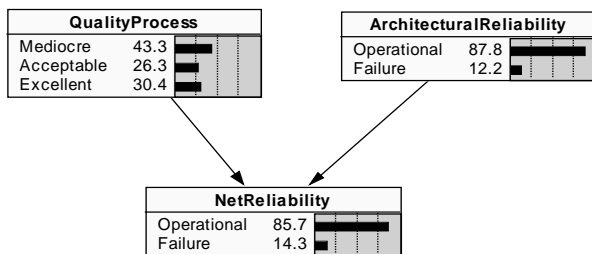


Figure 2. BBN for NVP system relating the quality process with initial reliability

The overall system reliability is computed using the failure state value of *NetReliability*, as the basic event probability of software failure for the NVP fault tree. Our analysis methodology was as follows: initially, we used the computed values for the quality process and evaluated the

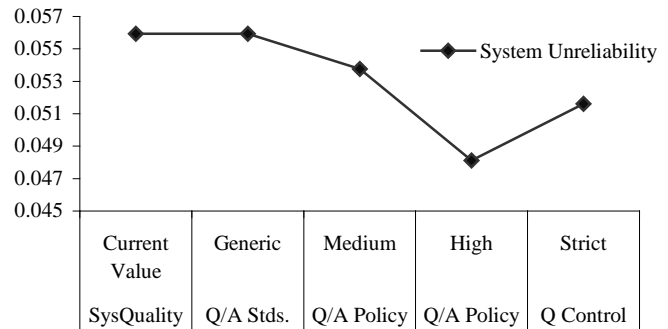


Figure 3. Effect of evidence of BBN nodes on unreliability

net reliability for the three models. Thereafter, we gave evidence for each of the states of the quality process and observed its influence on net reliability. Our final experiment was to vary the evidence on the nodes of the quality process BBN. Fig. 3 shows the results of this experiment.

### 4 Results and Concluding Remarks

All the models overestimate unreliability as compared to the initial value, however, for the BBN of Figure 2, the estimated reliability is closer to the initial value, as compared to the other two models that overestimate by a larger ratio. This is because the degree of dependence between *QualityProcess* and *ArchitecturalReliability* is difficult to determine. The second experiment yielded similar results reaffirming our initial assumption that the model of Fig. 2 is more correct than the other two models. Finally, from Fig. 3, we see that giving evidence to different nodes in the quality process influences system unreliability.

Intuitively, we anticipate that a high quality assurance policy will provide greater reliability as compared to a medium level quality assurance policy, and the results reflect this. We believe from our preliminary analysis that combining BBNs and fault trees appears to benefit software reliability estimation, by capturing causal dependencies between software engineering processes and software reliability. The results also indicate that the designs resulting from a combined analysis are likely to be more conservative in nature as compared to results of traditional methods. However, this is preferred over a design that has uncertain reliability estimates, especially for mission-critical or safety-critical systems that depend on software.

### References

- [1] K. S. Trivedi, *et al.*, “An analytical approach to architecture-based software reliability prediction”, In *Proc. of IPDS '98*.
- [2] B. Cukic *et al.*, “A bayesian approach to reliability prediction and assessment of component based systems”, To Appear In *Proc. of ISSRE 2001*, Nov. 2001
- [3] G. Dahll, “Combining disparate sources of information in the safety assessment of software-based systems”, *Nuclear Eng. and Design*, Vol. 195, No.3, 2000, pp. 307 – 319