Modelling software failures using Bayesian nets

Authors: Markus Porthin, Jan-Erik Holmberg

Confidentiality: Public
Report's title
Modelling software failures using Bayesian nets

Customer, contact person, address
VYR
European Commission

Order reference
34/2012SAF
269851

Project name
Safety evaluation and reliability analysis of nuclear automation
Harmonised Assessment of Reliability of Modern Nuclear I&C Software

Project number/Short
73874/SARANA
42925/HARMONICS

Author(s)
Markus Porthin, Jan-Erik Holmberg

Pages
24/4

Keywords
Nuclear power, software reliability, Bayesian belief net

Report identification code
VTT-R-08279-12

Summary
This report presents an approach to model independent post-release evaluation of safety-related software using Bayesian Nets (BN), with the specific aim to assess the probability of failure on demand (pfd) of the software. The models presented here differ from the usual ones presented in literature by assuming very little information, if any, to be available from the development process of the software, as it is usually confidential information not disclosed by the system provider. The advantage of BNs over other modelling approaches is that they enable combining of disparate types of evidence into the same model and can be graphically presented in an easily understandable manner. The models presented in this report include evidence on safety integrity level classification and design process quality, software complexity and operational usage or experience.

Confidentiality
Public

Espoo 7.2.2013
Written by
Jan-Erik Holmberg
Team Leader

Reviewed by
Janne Valkonen
Team Leader

Accepted by
Jari Hämäläinen
Technology Manager

VTT's contact address
firstname.lastname@vtt.fi

Distribution (customer and VTT)
SAFIR TR2 and TR8, HARMONICS project partners, VTT archive

The use of the name of the VTT Technical Research Centre of Finland (VTT) in advertising or publication in part of this report is only permissible with written authorisation from the VTT Technical Research Centre of Finland.
Preface

This report has been jointly prepared within the EU FP7 project HARMONICS (Harmonised Assessment of Reliability of Modern Nuclear I&C Software) and SAFIR2014 programme’s project SARANA (Safety evaluation and reliability analysis of nuclear automation).

Espoo, February 2013

Authors
List of contents

Preface ........................................................................................................................................... 2
List of contents ............................................................................................................................... 3
Abbreviations ................................................................................................................................. 4
1. Introduction ............................................................................................................................... 5
2. Safety integrity levels ............................................................................................................... 7
3. Bayesian nets ............................................................................................................................ 8
   3.1 Elements of Bayesian inference .......................................................................................... 8
   3.2 A basic BN example: Train strike ..................................................................................... 9
   3.3 Evidence and basic connections ....................................................................................... 10
   3.4 BN software ....................................................................................................................... 11
4. BN models for evaluating software reliability ........................................................................ 12
   4.1 Assumptions, definitions and limitations ........................................................................ 12
   4.2 Model 1: SIL and observation ......................................................................................... 13
   4.3 Model 2: SIL, observation and complexity ..................................................................... 14
5. Results ...................................................................................................................................... 14
   5.1 Model 1 ............................................................................................................................ 14
   5.2 Model 2 ............................................................................................................................ 17
6. Model validation ...................................................................................................................... 21
7. Conclusions ............................................................................................................................. 22
8. References ................................................................................................................................ 22
Appendix 1: Probability tables of the software reliability estimation examples ...................... 25
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BN</td>
<td>Bayesian net</td>
</tr>
<tr>
<td>SRGM</td>
<td>Software Reliability Growth Model</td>
</tr>
<tr>
<td>SIL</td>
<td>Safety integrity level</td>
</tr>
<tr>
<td>pfd</td>
<td>Probability of failure on demand</td>
</tr>
<tr>
<td>E/E/EP</td>
<td>Electrical, electronic, or programmable electronic</td>
</tr>
<tr>
<td>PSA</td>
<td>Probabilistic Safety Assessment</td>
</tr>
<tr>
<td>NPP</td>
<td>Nuclear power plants</td>
</tr>
<tr>
<td>CCF</td>
<td>Common cause failure</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical user interface</td>
</tr>
</tbody>
</table>

Dissemination level: PU
Date of issue of this report: 07/02/13
1. Introduction

Safety-related systems are used across industry sectors to reduce the risk associated with equipment that perform a function posing a risk to people, property or the environment. The risk is often reduced using electrical, electronic, or programmable electronic (E/E/EP) systems consisting of hardware and software. While there are well established methods to assess the reliability of hardware, analysis methods of software reliability are still evolving.

It is obvious that the failure mechanisms of software differ fundamentally from those of hardware. Software performs the tasks it has been programmed to do in a consistent manner, in principal always responding in the same way to identical input sequences. Because software failures are fundamentally systematic, there are different opinions whether software failures can be treated with probabilistic models or not. Nevertheless, probabilities can be used to represent our epistemic uncertainty of the performance of the software. Furthermore, the combinations of input sequences forming the exact state of the software are numerous and occur irregularly making the software failures to show up in a seemingly random manner.

Although being far less developed than hardware reliability modelling, different approaches striving to model software reliability can be found in literature. Shanthikumar (1983) reviews several models and groups them into empirical and analytical models, the latter further subdivided into static and dynamic models. In the analytical models some assumptions are made about the failure behaviour, e.g. independently distributed inter-failure times with similar distribution functions, but different parameters, or identical and independent probabilistic behaviour for each error. Generally the models require substantial data on software failures and produce estimates on e.g. number of remaining errors, mean time to next failure or reliability. The empirical models simply strive to find a suitable mathematical formula that fits the relationship between the software metrics and reliability measures without making any further assumptions.

Zhang (2004) divides software reliability models in white box models, which consider the structure of the software in estimating reliability, and black box models, which only consider failure data, or metrics that are gathered if test data is not available. Black box models generally assume that faults are perfectly fixed upon observation without introducing any new faults. Software reliability is thus increasing over time. Such models are known as Software Reliability Growth Models (SRGMs). Many of these have predictive power only over the short term, but long term models have also been developed (Bishop and Bloomfield, 1996). Zhang identifies four types of black box models: failure rate model, error seeding model, curve fitting model and Bayesian model. In white box, also called architecture based, models the software is decomposed into logically independent components that perform well-defined functions. Failures may occur during execution of a component or during the control transfer between two components. The system reliability is established by combining the component failure behaviours.
In the context of Probabilistic Safety Assessment (PSA) for nuclear power plants (NPP), there is an on-going discussion on how to treat software reliability in the quantification of reliability of systems important to safety. It is mostly agreed that software could and should be treated probabilistically (Dahll et al. 2007, Chu et al. 2009) but the question is to agree on a feasible approach. Chu et al. (2010) provides a review of possible methods. Software reliability estimation methods described in academic literature, shortly discussed above, are not applied in real industrial PSAs for NPPs. Software failures are either omitted in PSA or modelled in a very simple way as common cause failure (CCF) related to the application software of operating system (platform). It is difficult to find any basis for the numbers used except the reference to a standard statement that $10^{-4}$ per demand is a limit to reliability claims, which limit is then categorically used as a screening value for software CCF.

Bayesian belief nets, or shorter Bayesian nets (BN), have been suggested for software reliability estimation. Gran’s (2002) model estimates the failure probability by means of a soft-evidence part, with quality of the producer and complexity of the problem as the top nodes, and a testing part. The soft evidence part examines how well different objectives of the development and validation process defined in the standard DO-178B\(^1\) are met. Fenton et al. (2008) estimate the number of defects in a released software product and the number of defects found in operation by examining the design process quality, the complexity of the problem, the testing quality and the operational usage. Haapanen et al. (2004) developed a BN based approach for the reliability estimation of a computer-based motor protection relay, in which the evidence on the reliability was provided by the SW development team. Chu et al. (2012) suggest a BN model to represent fault insertion-and-removal process during the SW development. Bouissou et al. (1999) built a BN to assess safety-critical I&C systems developed by contracted suppliers using various sources of evidence related to both the development process and the final product. A more thorough discussion on the same topic, including idioms of BN fragments commonly occurring in safety assessments and instructions on how to construct safety arguments with BN is found in (SERENE, 1999).

This report presents an approach to model independent post-release evaluation of safety-related software using BN, with the specific aim to assess the probability of failure on demand (pfd) of the software. The models presented here differ from the ones described above by assuming very little information, if any, to be available from the development process of the software, as it is usually confidential information not disclosed by the system provider. The advantage of BNs over other modelling approaches is that they enable combining of disparate types of evidence into the same model (Littlewood, 2000) and can be graphically presented in an easily understandable manner. The models presented in this report include evidence on safety integrity level (SIL) classification and design process quality, software complexity and operational usage or experience.

\(^1\) RTCA/DO-178B "Software Considerations in Airborne Systems and Equipment Certification" (standard for software development in aircraft systems).
2. Safety integrity levels

The IEC 61508 Standard on functional safety of electrical/electronic/programmable electronic safety-related systems defines safety integrity as the likelihood of a safety-related system satisfactorily performing the required safety functions under all the stated conditions, within a stated period of time and a safety integrity level (SIL) as a discrete level (one of four) for specifying the safety integrity requirements of safety functions (IEC, 2010). A system or function is assigned a SIL based on risk assessment: The greater the required risk reduction provided by the safety-related system, the more reliable it needs to be, so the higher its SIL (Redmill, 1998). The target probabilities of critical failures of the system are presented in Table 1.

Table 1. Safety integrity levels and their target reliability.

<table>
<thead>
<tr>
<th>Safety integrity level (SIL)</th>
<th>Probability of dangerous failure on demand</th>
<th>Frequency of dangerous failure [h⁻¹] in continuous/high demand mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10⁻⁵ to 10⁻⁴</td>
<td>10⁻⁸ to 10⁻⁶</td>
</tr>
<tr>
<td>3</td>
<td>10⁻⁴ to 10⁻³</td>
<td>10⁻⁷ to 10⁻⁶</td>
</tr>
<tr>
<td>2</td>
<td>10⁻³ to 10⁻²</td>
<td>10⁻⁷ to 10⁻⁶</td>
</tr>
<tr>
<td>1</td>
<td>10⁻² to 10⁻¹</td>
<td>10⁻⁸ to 10⁻⁶</td>
</tr>
</tbody>
</table>

Software has no SIL in its own right, but IEC 61508-4 states that it is still convenient to talk about “SIL N software” meaning software in which confidence is justified (on a scale 1–4) that the element will not fail. The standard does not specify any reliability targets for the software itself, but it is evident that in order for the safety-related system to meet its SIL reliability target, the probability of critical failures of the software cannot be worse than the specified system level target.

Can a specified SIL be used in order to estimate the reliability of software? Basically this would be a circular deduction: Because it is required that the system (and its software) meets a specified reliability target, it will also do so. However, the idea is not that flawed as one at first might think. IEC 61508 gives guidance to the rigour appropriate to the various SILs and defines a number of methods and techniques to be appropriate for each SIL, still leaving room for informed management decisions, with the intention to ensure higher reliability for higher SILs. In other words, because of different levels of rigour required for software of different SIL, it can also be assumed that software with higher SIL will probably be better built and have higher reliability.

Holmberg and Bäckström (2012) suggest that IEC 61508 and the SIL concept could provide a framework for software probability estimates in the context of PSA for NPPs. There are several open issues that need to be addressed so that a common practice can be applied. The issues identified are mainly:

- Given a SIL: Which probability limit is justifiable, e.g., the upper or the lower limit of each SIL?
• How to consider different failure modes appropriately? Can the probability be split in different failure modes? If yes, what is a justifiable method for that?
• Failure modes for initiating event need to be addressed further.
• The diversification analysis of a system, and estimation of CCF probabilities – how shall that be achieved?

This report aims to answer some of the questions by means of BN based approach. A conceptual BN model of factors affecting the reliability of safety-related software is shown in Figure 1. As the information on the SIL of a system is usually available to the evaluator, it seems as a potentially relevant input for the reliability assessment of the software.

Figure 1. Conceptual model of factors affecting the reliability of safety-related software.

3. Bayesian nets

A Bayesian net (BN) is an intuitive graphical method of modelling probabilistic interrelations between variables. A BN consists of nodes representing variables which are interlinked with arches representing their causal or influential relationships. The variables can be either discrete, such as false/true or high/medium/low, or continuous. The causal relationships between variables are defined by conditional probability distributions, commonly referred to as node probability tables (NPT) or conditional probability tables (CPT). It is thus possible to calculate the marginal (or unconditional) probability distributions of the variables. Moreover, if evidence on some of the variables is observed, the other probabilities in the model can be updated using probability calculus and the Bayes theorem. This is referred to as propagation. (Korb and Nicholson, 2010; Fenton et al., 2008)

3.1 Elements of Bayesian inference

Bayesian inference is a process of fitting a probability model to a set of data and summarizing the result by a probability distribution on the parameters of the model and on unobserved quantities such as predictions of new observations (Gelman et al. 1997). Bayesian data analysis has the following main steps:
1. Construction of the full probability model, i.e., the joint probability model of all observable or unobservable quantities of the problem. BN is a way of doing it.
2. Conditioning on observed data, which a process of deriving the posterior probability distribution of the unobserved quantities given the observed data.
3. Evaluation of the fit of the model and the implications of the resulting posterior probability distribution. If needed, the steps 1–3 can be repeated.

The random variables of the probabilistic inference can be categorized into

- Observed quantities.
  - Data, evidence.
- Unobserved quantities.
  - Potentially observable: future data, censored data.
  - Non-observable: parameters governing the hypothetical process leading to the observed data.

Another way of distinguishing the random variables is into 1) parameters (unobserved), 2) data (observed) and 3) predictions (unobserved, but potentially observable). In the reliability analysis context, parameters are typically quantities such as failure rates and failure probabilities. Data include evidence to estimate the parameters such as operating experience and test results.

3.2 A basic BN example: Train strike

![Figure 2. BN for the train strike example.](image)

Let us consider a basic example introduced by (Fenton, 2012), see Figure 2. The two nodes in the lower part of the network represent the possibility of Martin and Norman arriving late to work. A train strike may influence their probabilities of being late, although not directly implying either of them being late, since they may go to work by another mode of transportation too. The probability of train strike is given in Table 2 and the conditional probabilities of the men being late given either train strike or no train strike are given in Table 3 and Table 4.

Table 2. Probability table for train strike.

<table>
<thead>
<tr>
<th>Train strike</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>0.1</td>
</tr>
<tr>
<td>False</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Using this information, it is possible to calculate the (unconditional) marginal probability of e.g. Martin arriving late to work:

\[
P(Martin\ late) = P(Martin\ late|Train\ strike) \cdot P(Train\ strike) + P(Martin\ late|No\ train\ strike) \cdot P(No\ train\ strike) \\
= 0.6 \cdot 0.1 + 0.5 \cdot 0.9 = 0.51
\]

Similarly, it can be calculated that the probability of Norman arriving late is 0.17. If the state of any of the nodes is observed, the probabilities of the others can be updated. E.g. if we know Norman is late, we can update our belief about a train strike using the Bayes theorem:

\[
P(Train\ strike|Norman\ late) = \frac{P(Norman\ late|Train\ strike) \cdot P(Train\ strike)}{P(Norman\ late)}
\]

\[
= \frac{0.8 \cdot 0.1}{0.17} = 0.47
\]

Using the updated probability of train strike, a new probability for Martin arriving late can also be calculated:

\[
P(Martin\ late|Norman\ late) = P(Martin\ late|Train\ strike,\ Norman\ late) \cdot \\
P(Train\ strike|Norman\ late) + P(Martin\ late|No\ train\ strike,\ Norman\ late) \cdot \\
P(No\ train\ strike|Norman\ late)
\]

\[
= 0.6 \cdot 0.47 + 0.5 \cdot 0.53 = 0.55
\]

### 3.3 Evidence and basic connections

Any BN can be constructed by combining three types of basic node connections: serial, diverging and converging, see Figure 3. In the serial connection in the figure, B has an
influence on A, which has an influence on C. In the diverging connection A has an influence on both B and C (as also in the example above) and in the converging connection both B and C have influence on A.

When evidence is observed on some node, the probabilities of the other ones can be updated. The evidence can be either a complete observation, e.g. node B is known to be in State1, called hard evidence, or incomplete, e.g. node B is in State1 with probability $p_1$ and in State 2 with probability $1 - p_1$, i.e. soft evidence.

A priori all nodes in the basic connections are d-connected except B and C in the converging connection. This means that receiving new evidence, be it either hard or soft, on any of the nodes affects our beliefs on the other nodes. However, hard evidence for A in the serial or diverging connection blocks evidence from B to C making them d-separated. Furthermore, receiving either hard or soft evidence for A in the converging connection makes B and C d-connected, i.e. knowledge on B will influence our beliefs on C.

The direction of the arrow expresses the direction of the stochastic dependence. It is a common convention to define that the parameter nodes are parents to the observable quantities, e.g., failure rate is a parent of the node for number of failures in testing.

3.4 BN software

Solving the unconditional node probabilities of a BN model is in general a mathematically demanding task, which in practice requires the usage of dedicated software. There has been a rapid development of BN software packages during the last 25 years and nowadays versatile tools are available. Commonly used BN software include e.g. Hugin Expert (www.hugin.com), Netica (www.norsys.com), Analytica (www.lumina.com), GeNNe and SMILE (http://genie.sis.pitt.edu), BayesiaLab (www.bayesia.com) and AgenaRisk (www.agenarisk.com). Thorough lists and evaluations of BN software packages can be found from (Murphy, 2012; Korb and Nicholson, 2010). In general, all packages include GUIs with user-friendly features. Many packages can handle only discrete variables. In such cases
continuous variables have to be discretised into a finite set of states. Choosing the number of intervals requires a compromise between model simplicity and accuracy. In many cases three or five well-chosen states is sufficient. When learning probabilities from data, also larger number of states can be used, e.g. using the dynamic discretisation algorithm in AgenaRisk (Fenton et al., 2008). Some software packages support continuous nodes without discretising e.g. by using conditional Gaussian models or mixtures of truncated exponentials models (Chen and Pollino, 2012). The calculations in this report have been done using GeNIe.

4. **BN models for evaluating software reliability**

This section presents two basic models for an independent post-release evaluation of the reliability of safety-related software. The target measure of reliability is the pfd of safety critical failures. The viewpoint is that of the user of the software or of an independent evaluator that has to make his conclusions based on the information provided by the user. Thus, in contrast to Gran’s model (2002), detailed information on the development process of the software is assumed not to be available, only the documentation that the supplier of the software has provided to the customer. In contrast to the model by Fenton et al. (2008), which evaluates the number of defects present and found in the software, the attempt is to estimate the pfd, which is needed in order to incorporate software failures in system level reliability models.

4.1 **Assumptions, definitions and limitations**

- The target measure of reliability is the pfd of safety critical failures.
- pfd is associated with a specific failure mode of a processor module performing data acquisition, data processing, voting functions, etc., typical to I&C units of a reactor protection system of a nuclear power plant.
- Failure mode is defined in functional way, e.g., failure to actuate when demanded.
- Hardware and software faults can be distinguished. Here we only consider software faults.
- Whether or not a software fault can appear in more than one processor module, causing simultaneous failure of several I&C units, is out of the scope of discussion of this report. Propagation of software fault to several I&C units is also out of the scope of discussion of this report. Software CCF will be discussed in another report.
- The BN model represents a class of similar kind of SW fault induced system failures. For instance, we may have a common BN model for application SW failures of a certain platform. Same model can be used to estimate a number of pfd:s, there may be some variation in the evidence for different pfd:s. For another platform or for operating system SW failures or for elementary function failures, different BN model may be required. In this report, application SW failures are mainly considered.
- The evidence (observable or potentially observable quantities) on SW reliability can partitioned into
  - those affecting pfd (kind of “performance shaping factors” as in human reliability analysis context),
those affected by pfd, such as test results and operating experience. Given pfd, the first and the second set of quantities are independent on each other. There is no direct link from the set 1 to set 2. This limitation could naturally be relaxed, but it would complicate considerably the mathematics. Therefore, the aim is to search for and define the pieces of evidence (set 1 and 2) in such a way that the conditional independence can be justified.

Figure 4. The basic structure of BN model for estimating pfd.

4.2 Model 1: SIL and observation
A basic model for assessing the pfd of software takes into account the SIL of the software and operational experience of its use, see Figure 5. Because the SIL of the software is seen to indicate its design process quality, it is assumed to affect the reliability and specifically the pfd of the software. Furthermore, the observed failures or other failure indications depend on the quality of the software, i.e. pfd, but also on the representativeness and extent of the usage or tests. Note that the observations do not affect the real reliability of the software (here neglecting repair of possible revealed bugs), but does affect our beliefs about the reliability.

Figure 5. Simple BN model for probability of failure on demand for software.

The possible values of the nodes can be seen in the conditional probability tables in the Appendix 1 (Model 1 Table 5 - Table 8). For the sake of demonstration, the tables have been populated with fictitious probability values. It is assumed that software classed to a certain SIL will a priori have a pfd falling within its target value (Table 1) with a probability of 0.7. The other probabilities are filled in using rational assumptions, e.g. higher pfd and higher test / usage.
representativeness gives higher probability of observing a failure. However, it should be noted that the probabilities are not based on any real data and no actual conclusion can be drawn from the strengths of influence between the nodes. The example shows that given these numbers and model structure, these are the results.

4.3 Model 2: SIL, observation and complexity

A slightly more versatile model than the one presented above takes into account also the complexity of the software. The more complex software, the more probable it is to fail. Complexity of software is not easy to define and measure accurately, so one may have to rely on some indicative complexity metrics or expert judgements. Still, receiving even indirect evidence on the complexity of the software influences our beliefs on its reliability, see Figure 6.

![Figure 6: BN model for probability of failure taking into account the complexity of the software.](image)

The possible values of the nodes as well as the fictitious probability numbers can be seen in the conditional probability tables in Appendix 1 (Model 2 Table 9 - Table 14). The probabilities follow the same rational as described for model 1 above. In addition it is assumed that more complex software will have higher probability to fail.

5. Results

5.1 Model 1

Figure 7 shows the marginal distributions of model 1 before any evidence has been entered. Thus, it represents the uncertainty before any specific information has been observed about the examined software, the nodes without parents having uniform distributions. An extra node called “pfd exp value” has been added to the model to calculate the expected value of the pfd using the mid-values of each interval in the states of the pfd node. 0.055 (i.e. the mid-value between $10^{-2}$ and $10^{-1}$) was used to represent the interval “$>10^{-2}$” and $5.5 \times 10^{-6}$ (i.e. the mid-value between $10^{-6}$ and $10^{-5}$) was used to represent the interval “$<10^{-5}$.” With
no further evidence on the examined software, there is a wide distribution for the pfd value, its expected value being approximately 0.0138.

Adding the information that the software has a SIL 3 classification narrows the distribution for pfd substantially, resulting in an expected value of 0.0027, see Figure 8. If no faults or suspicious behaviour have been observed, it decreases the expected value of pfd further to 0.0016 (Figure 9). Its effect on the pfd is still rather small compared to only knowing the SIL value. However, observing a fault in the software changes our beliefs quite strongly, resulting in an expected value of pfd of 0.0124 (Figure 10). The model offers an explanation, that the observed fault in the software is likely to be a consequence of the test or usage representativeness being high. If the SIL of the software is not known, but there is substantial usage experience available with no faults detected, it supports the belief in the high reliability of the software, but the effect is not as strong as if the software was known of having e.g. a SIL 3 (Figure 11).

Figure 7. BN model with marginal distributions for variables superimposed on nodes.
HARMONICS
(N°: WP2.06_VTT) – Modelling software failures using Bayesian Nets
Dissemination level: PU
Date of issue of this report: 07/02/13
5.2 Model 2
In the same manner as for model 1, Figure 12 shows the marginal distributions of model 2 before any evidence has been entered. Entering the same evidence to model 2 as for model 1 above results in nearly identical results (Figure 13 - Figure 16). In addition, it can be seen from Figure 15 that observing a fault in SIL 3 software is likely to be caused by high complexity of the model. With the current fictitious conditional probabilities used in the BN model, the complexity analysis has an effect on the belief on the pfd, i.e. with low complexity the pfd is believed to be lower than with high complexity (see Figure 17 and Figure 18). However, the effect is quite small.

Figure 10. SIL 3 and fault detected.

Figure 11. OK observation and high test / usage representativeness.
Figure 12. BN model incorporating software complexity with marginal distributions for variables superimposed on nodes.

Figure 13. SIL 3 observed.
Figure 14. SIL 3 and Ok observation.

Figure 15. SIL 3 and fault detected.
Figure 16. OK observation and high test / usage representativeness.

Figure 17. SIL 3 and complexity analysis with result "low".
6. Model validation

The hard question to any BN model is how the model can be validated. The validity of BN models is generally tested through one of two procedures: by comparing the model predictions to empirical data or by asking the experts involved in the model development to review on it. Pitchforth and Mengersen (2013) argue that these tests are limited and propose a general validation framework for expert elicited BNs.

BN model can be considered to consist of four elements:

- model structure,
- node discretisation (taking continuous factors and assigning them intervals, ordinal states or categories),
- parameterisation (adding the values elicited from experts to the BN),
- model behaviour (the joint likelihood of the entire network as well as its sub-networks and relationships).

Each of these elements can be raised as a source of uncertainty in BN modelling. The structure, discretisation and parameterisation should be tested for validity before any model behaviour tests can be run.

Pitchforth and Mengersen (2013) provide a broad range of conceptual tests that can be applied to validate BNs. These validity tests incorporate standard model-data fit comparisons, but expand the construct of validity to the broader definition of whether or not a model describes the system it is intended to describe, and produces output it is
intended to produce. The issue of model validation will be further studied in the next phase of the project.

7. **Conclusions**

This report shows examples on how BN can be used to model independent post-release evaluation of the reliability of safety-related software using pfd as the target metrics. SIL, operational experience or testing and software complexity are used as evidence when evaluating the pfd of the software. Although the model uses fictitious data, the results still show sensible features. This suggests that BN is a potential approach for modelling these types of problems. An advantage of BNs is that they enable combining different types of evidence in the same model.

In order to achieve a credible model for the pfd, the structure of the model should first be refined and further elaborated using the field experience of companies using safety-related software. It depends on what evidence is really available in practice for conducting an independent post-release evaluation. Further, the model parameters should be carefully examined. In an ideal world these would be populated using an extensive set of data fit for the purpose. While such a data set seems hard to be found, expert elicitations are probably a more feasible approach.

8. **References**


Chu, T.L., Yue, M., Varuttamaseni, A., Kim, M.C., Eom, H.S., Son, H.S., and Azarm, A. (2012) Applying Bayesian belief network method to quantifying software failure probability of a protection system, Proc. of the 8th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies, NPIC & HMIT 2012,


Zhang, Yi (2004). Reliability Quantification of Nuclear Safety-Related Software. DPh Thesis in Nuclear Engineering, Massachusetts Institute of Technology.
Appendix 1: Probability tables of the software reliability estimation examples

Model 1

Table 5. Probability table for SIL of the software.

<table>
<thead>
<tr>
<th>SIL</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIL 1</td>
<td>0.25</td>
</tr>
<tr>
<td>SIL 2</td>
<td>0.25</td>
</tr>
<tr>
<td>SIL 3</td>
<td>0.25</td>
</tr>
<tr>
<td>SIL 4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 6. Probability table for the test / usage representativeness.

<table>
<thead>
<tr>
<th>Test / usage representativeness</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>0.333</td>
</tr>
<tr>
<td>Medium</td>
<td>0.333</td>
</tr>
<tr>
<td>Low</td>
<td>0.333</td>
</tr>
</tbody>
</table>

Table 7. Conditional probability table for the pfd of the software given its SIL.

<table>
<thead>
<tr>
<th>SIL</th>
<th>SIL 1</th>
<th>SIL 2</th>
<th>SIL 3</th>
<th>SIL 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt; 10^{-2}</td>
<td>0.7</td>
<td>0.15</td>
<td>0.03</td>
<td>0.003</td>
</tr>
<tr>
<td>10^{-3} – 10^{-2}</td>
<td>0.27</td>
<td>0.7</td>
<td>0.12</td>
<td>0.027</td>
</tr>
</tbody>
</table>
Table 8. Conditional probability table for observation given pfd and test / usage representativeness.

<table>
<thead>
<tr>
<th>pfd</th>
<th>&gt; 10⁻²</th>
<th>10⁻³ – 10⁻²</th>
<th>10⁻⁴ – 10⁻³</th>
<th>10⁻⁵ – 10⁻⁴</th>
<th>&lt; 10⁻⁵</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test / usage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>representativeness</td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>Observation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td>0.333</td>
<td>0.38</td>
<td>0.42</td>
<td>0.42</td>
<td>0.57</td>
</tr>
<tr>
<td>Suspicious</td>
<td>0.333</td>
<td>0.34</td>
<td>0.34</td>
<td>0.33</td>
<td>0.2</td>
</tr>
<tr>
<td>Fault detected</td>
<td>0.333</td>
<td>0.28</td>
<td>0.24</td>
<td>0.24</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Model 2

Table 9. Probability table for SIL of the software.

<table>
<thead>
<tr>
<th>SIL</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIL 1</td>
<td>0.25</td>
</tr>
<tr>
<td>SIL 2</td>
<td>0.25</td>
</tr>
<tr>
<td>SIL 3</td>
<td>0.25</td>
</tr>
<tr>
<td>SIL 4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 10. Probability table for the test / usage representativeness.

<table>
<thead>
<tr>
<th>Test / usage representativeness</th>
<th>Probability</th>
</tr>
</thead>
</table>

[HARMONICS]
(N°: WP2.06_VTT) – Modelling software failures using Bayesian Nets
Dissemination level: PU
Date of issue of this report: 07/02/13
Table 11. Probability table for the complexity of the software.

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>0.333</td>
</tr>
<tr>
<td>Medium</td>
<td>0.333</td>
</tr>
<tr>
<td>Low</td>
<td>0.333</td>
</tr>
</tbody>
</table>

Table 12 Conditional probability table for the result of the complexity analysis of the software given its true underlying complexity.

<table>
<thead>
<tr>
<th>Complexity</th>
<th>High</th>
<th>Medium</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High assessment</td>
<td>0.7</td>
<td>0.15</td>
<td>0.1</td>
</tr>
<tr>
<td>Medium assessment</td>
<td>0.2</td>
<td>0.7</td>
<td>0.2</td>
</tr>
<tr>
<td>Low assessment</td>
<td>0.1</td>
<td>0.15</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 13. Conditional probability table for the pfd of the software given its SIL and complexity.

<table>
<thead>
<tr>
<th>SIL</th>
<th>SIL 1</th>
<th>SIL 2</th>
<th>SIL 3</th>
<th>SIL 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>pfd</td>
<td></td>
<td></td>
<td></td>
<td>&gt; 10^{-2}</td>
</tr>
<tr>
<td></td>
<td>10^{-3} – 10^{-2}</td>
<td>0.18</td>
<td>0.27</td>
<td>0.36</td>
</tr>
<tr>
<td>pfd</td>
<td>&gt; 10^{-2}</td>
<td>10^{-3} – 10^{-2}</td>
<td>10^{-4} – 10^{-3}</td>
<td>10^{-5} – 10^{-4}</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Test / usage representativeness</td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>Observation</td>
<td>OK</td>
<td>0.333 0.38 0.42 0.42 0.57 0.75</td>
<td>0.75 0.85 0.95</td>
<td>0.95 0.985 0.999</td>
</tr>
<tr>
<td>Suspicious</td>
<td>0.333 0.34 0.34 0.34 0.33 0.2</td>
<td>0.2 0.12 0.04</td>
<td>0.04 0.01 0.0009</td>
<td>0.0009 9e-5 9e-6</td>
</tr>
<tr>
<td>Fault detected</td>
<td>0.333 0.28 0.24 0.24 0.01 0.05</td>
<td>0.05 0.03 0.01</td>
<td>0.01 0.005 0.0001</td>
<td>0.0001 1e-5 1e-6</td>
</tr>
</tbody>
</table>

Table 14. Conditional probability table for observation given pfd and test / usage representativeness.