STUK-YTO-TR 198 / JUNE 2003

RELIABILITY ASSESSMENT USING BAYESIAN NETWORKS

Case study on quantative reliability estimation of a software-based motor protection relay

Atte Helminen, Urho Pulkkinen

VTT Industrial Systems

In STUK this study was supervised by Marja-Leerna Järvinen

**STUK • SÄTEILYTURVAKESKUS
STRÅLSÄKERHETSCENTRALEN
RADIATION AND NUCLEAR SAFETY AUTHORITY**

Osoite/Address • Laippatie 4, 00880 Helsinki
Postiosoite / Postal address • PL / P.O.Box 14, FIN-00881 Helsinki, FINLAND
Puh./Tel. (09) 759 881, +358 9 759 881 • Fax (09) 759 88 500, +358 9 759 88 500 • www.stuk.fi

ISBN 951-712-690-5 (print)
ISBN 951-712-691-3 (pdf)
ISSN 0785-9325

Dark Oy, Vantaa/Finland 2003

HELMINEN Atte, PULKKINEN Urho (VTT Industrial Systems). Reliability assessment using Bayesian networks.. Case study on quantative reliability estimation of a software-based motor protection relay. STUK-YTO-TR 198. Helsinki 2003. 28 pp. + Appendices 3 pp.

Keywords: reliability assessment, software-based system, Bayesian network

Abstract

In this report a quantitative reliability assessment of motor protection relay SPAM 150 C has been carried out. The assessment focuses to the methodological analysis of the quantitative reliability assessment using the software-based motor protection relay as a case study. The assessment method is based on Bayesian networks and tries to take the full advantage of the previous work done in a project called Programmable Automation System Safety Integrity assessment (PASSI).

From the results and experiences achieved during the work it is justified to claim that the assessment method presented in the work enables a flexible use of qualitative and quantitative elements of reliability related evidence in a single reliability assessment. At the same time the assessment method is a concurrent way of reasoning one's beliefs and references about the reliability of the system.

Full advantage of the assessment method is taken when using the method as a way to cultivate the information related to the reliability of software-based systems. The method can also be used as a communicational instrument in a licensing process of software-based systems.

Foreword

This report is a description of a pilot case study in which the ideas developed earlier in a project called Programmable Automation System Safety Integrity assessment (PASSI) are tested first time in practice. The practical implementation and further developed of the reliability assessment method has been laborious but also very interesting and encouraging. In the case study the theory and the assessment method are efficiently demonstrated. However, this has only been the first step and similar case studies need to be analysed in the future research. Through case studies like the one carried out in the work it is possible to obtain a maximum learning from the relation between software development process and software reliability. At the same time the case studies guide toward the best ways of implementing reliability estimation methods to software development processes.

The assessment was carried out in VTT Industrial Automation and the research team consists of Professor Urho Pulkkinen and research scientist Atte Helminen. The assessment was carried out in co-operation with ABB Substation Automation. The authors would like to thank ABB Substation Automation and especially Tapio Hakola, Grels Linqvist, Tapio Niemi and Henrik Sundell for their time and expertise in this reliability assessment process.

Contents

1 Introduction

The need for shifting from analog instrumentation and control (I&C) systems to corresponding digital I&C systems is becoming current also with the I&C systems of nuclear power plants. To have a control over this transition and to have basis for the licensing of digital I&C systems there needs to be solid methods for assessing the reliability of the new digital I&C systems. However, to give the maximum support for the licensing process of digital I&C systems the reliability assessment method should not only produce exact values related to reliability, but with the assessment method the reasons for the produced values should be clarified.

With the analog systems there is the nice feature that systems can be tested thoroughly. For digital systems this is usually not the case. Digital systems mainly fail because of their inherent design faults, which are triggered at appropriate conditions with certain inputs. Since the number of possible inputs for even relatively simple systems becomes unreasonably large, the system can only be tested to a certain extent. As the system is not tested with all the possible inputs under all different conditions i.e. under all operational profiles, there is always some uncertainty left to the reliability of the system. The most convenient way to handle such uncertainty is through the use of probabilistic calculus.

For digital systems that can't be tested exhaustively there rises a question if such systems can ever be used in applications for which high reliability is required. One way to tackle this dilemma is to compensate the lack of reliability related testing evidence with evidence from other sources. Such sources are for example the system development process, design features and operational experience. In the reliability assessment of digital systems this compensation of evidence is not always a straightforward operation. Different evidence sources involve many qualitative characteristics that are difficult to translate to unambiguous quantitative measures. These characteristics contain a lot of information relevant to the reliability of the system and should not be overlooked and thrown away. Therefore, the methods used for the reliability estimation should have enough flexibility to overcome the problem.

The approach used in the work tries to take the points mentioned above into consideration. The approach is based on Bayesian statistics and in particular to its technical solution called Bayesian networks. Bayesian statistics enables the implementation of qualitative and quantitative information flexibly together as well as updating the estimation while new information is obtained about the system and introduced to the estimation. The reliability assessment carried out in the report is a practical continuum to the work done previously in PASSI-project. The theory and ideas on which the assessment is mainly based on can be found more explicitly explained in report by Helminen [1]. The purpose of the report is to test the developed assessment method with a practical case study. The emphasis has been on the methodological analysis of the quantitative reliability assessment. The details related to the technical features of the case study system and to the evidence used have only been reviewed on a level necessary to test the functionality of the assessment method.

In Chapter 2 a general description of the system under case study and some initial assumptions of the assessment are given. Chapter 3 contains a representation of the assessment process and overview on the evidence involved in the assessment. Closer review on the mathematical formalism and the Bayesian network model used in the work for the reliability estimation is given in Chapter 4. Chapter 4 includes more detailed information, which is not needed for all readers but gives valuable information for advanced readers interested on the mathematical viewpoint of the topic. In Chapter 5 a detailed explanation of the assessment process and the evidence is given. The numerical results of the assessment are presented in Chapter 6. The assessment process and the results are discussed in Chapter 7 and the last chapter, Chapter 8, is left for short conclusions about the work.

2 Description of the assessment target and assumptions

2.1 Target of the case study

The system under assessment is SPAM 150 C – motor protection relay produced by ABB Substation Automation. A picture of the relay is shown in Figure 1. The numerical motor protection relay SPAM 150 C is an integrated design current measuring multifunction relay for the complete protection of alternating current motors. The main area of application covers large and medium-sized three-phase motors in all types of conventional contactor or circuit-breaker controlled motor drives.

The relay continuously measures the three phase currents and the residual current of the protected object. When a fault occurs, the relay starts and operates, if the fault persists long enough to exceed the set or calculated operate time of the relay. Depending on the relay setting and on the fault occured the relay either gives an alarm or a launch signal for the protection operation. The multi-function relay module comprises seven units: a three-phase overcurrent unit, a thermal overload unit, start-up supervision unit, a phase unbalance unit, an incorrect phase sequence unit, an undercurrent unit and a non-directional earth-fault unit. The descriptions of the protection functions for different units are following:

Figure 1. Motor protection relay SPAM 150 C. [2]

- The overcurrent unit holds a low-set stage *Is* and a high-set stage *I>>*. The high-set stage provides short-circuit protection. The low-set stage can be used for start-up supervision or for time overcurrent protection.
- The thermal unit supervises the thermal stress of the protected object during various load conditions. The unit provides thermal prior alarm and thermal tripping and it prevents the protected object from being re-energised, if the protected object is too hot to be successfully re-energised.
- The start-up supervision can be realised according to several principles. It can be based on measuring the start time, measuring the thermal stress at start-up or the use of an external speed switch.
- The phase unbalance unit protects, e.g. motors, from the stress caused by an unbalanced network. The unit operates with inverse time characteristic. The operate time of the unit at full unbalance, i.e. at loss of one phase is 1 second.
- The incorrect phase sequence protection has a factory-set operate time of 0.6 seconds.
- The undercurrent unit is used for the protection of motors at sudden loss of load in certain applications, such as conveyors and submersible pumps. The unit features definite time characteristic.
- The earth-fault unit provides a sensitive earth-fault protection with definite time characteristic.

The relay incorporates a sophisticated self-supervision system with auto-diagnosis, which increases the availability of the relay and the reliability of the system. The self-supervision system continuously monitors the hardware and the software of the relay. The system also supervises the operation of the auxiliary supply module and the voltage generated by the module. [2]

2.2 Assumptions in the assessment

The reliability assessment involves evidence of both quantitative and qualitative nature. To make the assessment possible some assumptions on the evidence was made. For example the operational experience data is based directly on the information obtained from the target system manufacturer, and no additional evaluation was carried out. The following evidence was accepted in the assessment as such:

- classification of software faults
- number of software faults for different software versions
- classification of software changes
- number of software changes made between different software versions
- estimated amount of operational experience for different software versions.

In principle, the evidence listed above could have been interpreted as uncertain, and therefore described as random variables in the assessment model. In the assessment more evidence was collected and certain assumptions was made using expert judgements and the given evidence. To carry out the assessment following assumptions was made:

- Prior estimation on the reliability of the first software version given by the system developers
- Prior estimation on the reliability changes between software versions given by the assessment executives
- Different software versions function in a single and similar operational profile

In addition, to enable the combination of disparate evidence together and on the other hand to maintain the workload in a reasonable scale certain simplifications to the given evidence were made. These simplifications are described in detail in the following chapters.

3 Description of assessment process

3.1 Overview

The idea in the assessment process is to combine all available evidence to form a reliability estimation of the target system. Characteristics of different kind of evidence and the main sources of reliability related evidence in the case of software-based safety critical system is explained more explicitly in report by Helminen [1]. In the assessment the evidence was mainly obtained from the system development process and from the operational experience of the system. The evidence of system testing involved in the assessment was integrated as a part of the system development process evidence.

In the assessment process the whole life cycle of the motor protection relay was taken into consideration. A diagram representing the assessment process is depicted in Figure 2. First, a prior estimate for the reliability of system's first software version was built. The prior estimate is mainly based on the quality evidence of system development process. This estimate was then updated using the data, i.e. operational experience, obtained for the first software version. Later on when the software was modified the effect of the modifications to the system reliability was estimated and operational experience for the second version was introduced to the estimation. This procedure was repeated, as many times as there were new software versions produced in the life cycle of the system.

3.2 Prior estimation

The analysis began from building a prior estimation for the first software version of the system. The prior estimation was built using the expert judgements given by the designers and the ones responsible for the implementation of the system. The expert judgements were collected in an interview, which was mainly based on guidebooks, recommendations and standards for developing high quality software-based systems. An important aspect in the interview was to compare the validity of the given expert judgements to the documentation produced from the system during the life cycle of the system. The questions asked in the interview is presented in Appendix 1. In the interview the questions concerning the software development process were divided in five separate groups: project control and quality, requirement specification phase, design phase, implementation phase and testing phase.

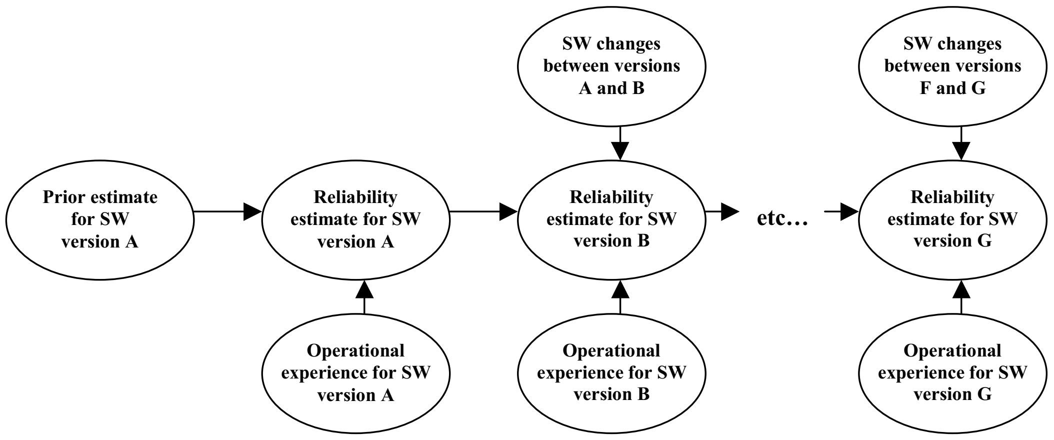

SW changes between versions A and B
SW changes between versions F and G
Prior estimate for SW version A
Reliability estimate for SW version A
Reliability estimate for SW version B
etc...
Reliability estimate for SW version G
Operational experience for SW version A
Operational experience for SW version B
Operational experience for SW version G

Figure 2. Diagram representing the software (SW) reliability assessment process.

Before the interview the experts received a short period of training on the assessment process and how to give probability estimations based on expert judgement overall. The training session was given to the experts at the same time, but the interviews were made individually. The questions in the interview formed a basis for a discussion with a expert on matters related to different software development phases. After the query of each software development phase an expert was asked to quantify how well he thought the production team managed in the execution of the phase in a numerical scale of one to ten. The numerical value given by the expert was called a score value. The expert was also asked to give a weight on the importance of the software development phase to the system reliability. In the training session the numerical scale used in the interview and the meaning of different score values were discussed and this way each expert was able to build an impression of his own about the meaning of the scaled values in the aspect of system reliability.

The last step of the interview, after giving the score values and the weights, the expert was asked to give a system failure frequency distribution for two or three different score values between one and ten. The values are used for the calibration of the scale of score values. The values didn't need to be among the score values given earlier in the interview, but the system failure frequency distributions should represent expert's impression of the score values as well as possible. By giving failure frequency distributions for arbitrary score values it was possible to extrapolate a failure frequency distribution for other score values in the scale.

The score values of each expert were combined using an additive value function, which then provided a single score value representing the expert's reliability estimation for the first software version of the system. In the final reliability assessment the reliability estimates or failure frequency distributions of all the experts were combined together to form the prior failure estimate for the first software version of the target system.

Table I. Categories of software faults.

Category	Software fault criticality:
1	Causes no trouble to customer
2	Causes inconvenience to small fraction of customers
3	Causes inconvenience to over 80% of customers or critical fault to small fraction of customers
4	Critical fault, which requires announcement to all customers and updates if requested by the customer
5	Critical fault, which requires an update to all customers

3.3 Operational experience

The operational experience data included to the estimation was statistical operational experience data estimated for each software version of the target system. The operational experience considered in this work is the amount of working years estimated for each software version and the amount of software faults detected for the software version during these working years. The software faults encountered for each software version were reported either by the developer or the customers using the system. The faults were analysed and depending on the nature of the faults they were corrected in the next software version.

In the assessment the software faults were classified into five categories depending on their severity from the customer's point of view. The categories were numbered from one to five, one being the least critical and five being the most critical fault to the customer. The five categories and their descriptions are given in Table I. The label software fault may be somewhat a harsh and misleading description for all the categories described in Table I. Some of the faults encountered for the device from the customer's point of view might be caused for example from using the device for a purpose it wasn't even originally designed for. Among other things this aspect is taken into consideration in a new categorisation carried out later in Chapter 5.

3.4 Version management

In case there are several software versions produced during the life cycle of a software-based system it is important to be able to use the evidence from previous software versions in the reliability estimation of the last software version. This is especially important if the operational experience available from the latest software version is small.

If the reliability assessment is carried out for a software-based system involving two or more software versions, the effects of the modifications between the software versions and their criticality to the system reliability need to be evaluated. In the assessment the evaluation was carried out by classifying the changes in software to five different categories depending on the criticality the changes have to the software. The categories were numbered from one to five. Making a change from category one is expected to have the least critical significance while making a change from category five is expected to have the most critical significance to the software and therefore to the reliability of the system. The five different categories and their descriptions are given in Table II.

Table II. Categories of software changes.

Category	Software change criticality:
1	Limit value change, having no strain to CPU
2	Change to a module, which can be well tested using automatic tests scripts
3	Change to a protection function
4	Major change having an influence to the interruption vectors
5	Major change having an influence to many places in the software

Based on the number of faults and changes made for a software version and depending on the categories the faults and changes fall into a suitable increment or reduction to the failure frequency distribution is introduced in the assessment. More of this procedure is explained in Chapter 5.

4 Bayesian network model

4.1 Overview

The approach used in the assessment is based on Bayesian statistics and in particular to its technical solution called Bayesian networks. More detailed description of the Bayesian statistics and the Bayesian network theory can be found for example in books by Gelman et al. [3] and Box & Tiao [4]. An application of a Bayesian framework in combining expert judgements is explained in detail for example in a report by Pulkkinen & Holmberg [5]. More explicit description on the theory of using Bayesian networks to the reliability assessment of software-based systems can be found for example in reports by Helminen [1] and Korhonen et al. [6]. The Bayesian network modelling and simulations in this report are carried out using WinBUGS program, and so all representations of Bayesian networks shown below are in WinBUGS format. For a closer review about WinBUGS program see Spiegelhalter et al. [7].

The basic principle of the Bayesian network model used in the work follows the general description of the assessment process given in Figure 2. More detailed description of the evidence used in the assessment and how to quantify it for the use of the model will be given later in the text. In the model the prior distribution for the reliability of the first software version is determined using expert judgements on the software development process. The prior distribution is then updated by using the operation experience of the first software version. The corrections and changes made to develop the next software version were taken into account by assuming that the reliability parameters are subjected to a random disturbance. This leads to a prior distribution of the reliability of the second software version, which is updated by using the operational experience of the second software version. The procedure is repeated for all software versions.

A Bayesian network utilising the evidence related to the assessment process is depicted in Appendix 2. The Bayesian network presented in Appendix 2 is the network used to obtain a quantitative reliability estimation of the system under study. The WinBUGS-code giving an unambiguous description of the network is given in Appendix 3. In this chapter parts of the Bayesian network shown in Appendix 2 are explained in more detail.

4.2 Prior for first software version

Implementation of the prior estimation for the first software version given by the four experts is taken care by the upper left part of the network. The part of network under interest is pictured also in Figure 3. Figure 3 describes the determination of prior distribution for the reliability parameter of the first software version. Mathematically, the reliability parameter, *ThetaA*, is assumed to be a mixture of normal distributions, determined by the judgements of the four experts. In the network a discrete value for parameter *Z* is sampled in each sampling round. *Z* determines the parameter values of *Mu* and *InvVar*. The values possible for *Mu* and *InvVar* are listed in parameters *OpMu[]* and *OpInvVar[]*. In *OpMu[]* and *OpInvVar[]* are entered the prior parameters of the first software version given by the four experts. *Mu* and *InvVar* are the parameters of the normal distribution *ThetaA*. Therefore, the distri-

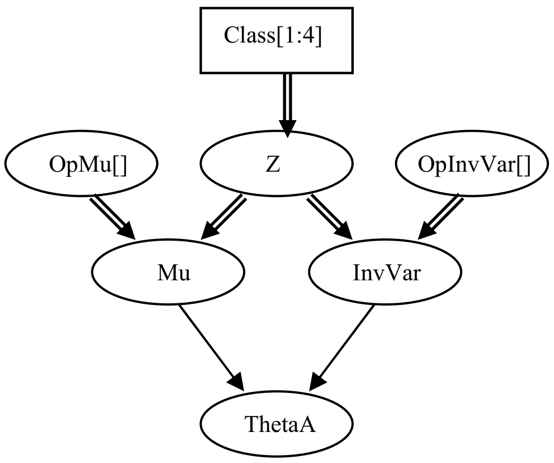


Figure 3. Network to build prior reliability estimation for the first software version.

bution of *ThetaA*, correspond to mixture:

$$p(ThetaA) = \sum_{i=1}^{4} p_i\,(ThetaA)\,,$$

where $p_i(ThetaA) = N(OpMu[i], 1 / OpInvVar[i])$ are the normal distributions based on the opinions of the four experts.

4.3 Implementing operational experience of software version

Implementation of operational experience is carried out separately for all software versions and can be seen as the repetitive lower part of the network shown in Appendix 2. The repetitive part for the first software version is shown in Figure 4. Since the operational experience obtained in the assessment for each software version is in a form of failures found in a certain time interval it is reasonable to assume that failures X1 are Poisson distributed with mean parameter t1LambdaA as following:

$X1 \sim Poisson(t1LambdaA)$,

where parameter *t1LambdaA* is a product of time interval *t1* and the failure frequency of the software version named as parameter *LambdaA*.

Normal distributed failure parameter *ThetaA* is integrated to the network to enable more flexible combination of different prior estimations and software versions using a single Bayesian network. The idea is analogous with the combination of different operational environments in a single network as explained in report by Helminen [1]. Normal distribution is defined in the interval (-∞,∞), and since the failure frequency parameter *LambdaA* can only receive positive values a log-transformation is needed and, therefore, *LambdaA* is the log-transformation of parameter *ThetaA* defined as following:

$LambdaA = exp(ThetaA)$.

The underlying model in the network depicted in Figure 4 is so called lognormal-Poisson model. Closer review on using lognormal-Poisson model in failure frequency rate estimation is found for example in report by Pulkkinen & Simola [8].

4.4 Implementing influence of software changes

Implementation of the influence of software changes between software versions is carried out with a repetitive upper part of the network shown in Appendix 2. The repetitive part for the first two software versions is shown in Figure 5. Idea in combining the evidence from successive software versions is that the failure probability of the latter software version, *ThetaB*, is the same as the failure probability of the preceding software version, *ThetaA*, added with a normal distributed random accretion or reduction term named *OmegaAB*. The relation of the parameters is defined as following:

$ThetaB = ThetaA + OmegaAB$.

Variables *MuAB* and *InvVarAB* determine the prior parameters for the normal distributed random variable *OmegaAB* as following:

$$OmegaAB \sim N\left(MuAB, {}^{1}\!/\!_{InvVarAB}\right).$$

The values for parameters *MuAB* and *InvVarAB* used in the assessment are given for two different approaches in the last two columns of Table IX.

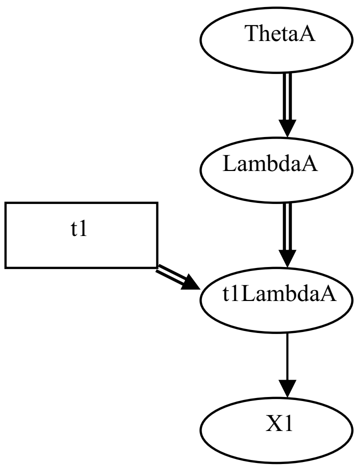


Figure 4. Network to implement operational experience to software version.

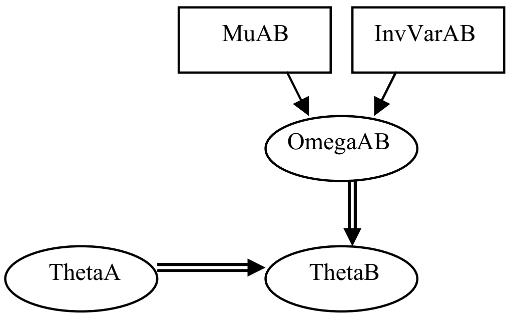


Figure 5. Network to implement influence of software changes.

5 Description of evidence

5.1 Overview

There were four experts taking part in the assessment process. The amount of experts was approximately half the size of the whole production team of motor protection relay SPAM 150 C. All experts involved in the assessment process were current employees of ABB Substation Automation and had a major role in the development process and execution of the relay. Two of the experts were mainly involved with the tasks related to the system requirement specification and project management. The other two were mainly involved with the system design, implementation and testing.

The software in the motor protection relay was strongly based on the software made for its predecessors and the relay was a part of so called SPACOM-product family. The legacy from the previous products had probably a strong influence to the reliability estimations given by the experts, but this matter was not explicitly considered in the work.

The first version of SPAM 150 C was produced during years 1989 and 1990 in which the approximately amount of work was two man-years. The total amount of man-years used for producing the predecessors from year 1980 to 1989 was about twelve. The software includes approximately 40 000 lines of code about half of the lines being for commentary.

5.2 Expert judgements

A diagram representing the actual steps of the expert judgement process for building up the prior estimation for the first software version is presented in Figure 6. Steps 1, 2 and 4 were carried out during the interview described in general in Chapter 3. The fourth step turned out to be very troublesome for the experts. Reason for the difficulty lay in the approach used in the interview for the elaboration of the failure frequency distributions of the arbitrary score values. The question setting was such that the experts were supposed to give the failure frequency distribution of software faults causing the system to fail in a protection function on a scale of system failures per protection demands given to the system. Taking into consideration the continuous protection characteristic of the relay a better approach would have been to scale the failure frequency distributions on a scale of system failures per time the system is operating. This impracticality was corrected by renewing the fourth step of the expert judgement process in a written form within a week from the original interview.

5.2.1 Steps 1 to 3

In the first two steps of expert judgement process the five different phases of the software development process were discussed in an interview. Based on the previous experience and the conclusions made in the interview the experts gave score values and weights for the different software development phases. The score values and weights for the different experts are shown in Table III.

Using the additive value function presented in the step 3 of Figure 6 total score values were calculated from the score values and weights given by the experts. The total score values for the experts are listed in the last column of Table III.

5.2.2 Step 4

To convert the total score value of the expert to corresponding failure frequency distribution each expert was asked to give failure frequency distributions for two or three different score values. To be more specific, an expert was asked to give failure frequency fractiles of given score values. Failure frequency distributions for the score values were then solved by fitting lognormal distributions for the failure frequency fractiles given by

Table III. Score and weight values given by the experts.

Phase:	Project control and quality		Requirement specification		Design		Implementation		Testing		Total
	weight	score	weight	score	weight	score	weight	score	weight	score	score
Expert 1	5	5,5	8	8	7	7,5	6	7,5	7	8,75	**7,58**
Expert 2	7	7	8	8	9	6,5	7	8,5	10	9	**7,83**
Expert 3	9	8,5	10	9	8	8,5	5	8	10	9	**8,68**
Expert 4	8	9	10	9,5	7	8,5	9	9,25	8	9,5	**9,18**

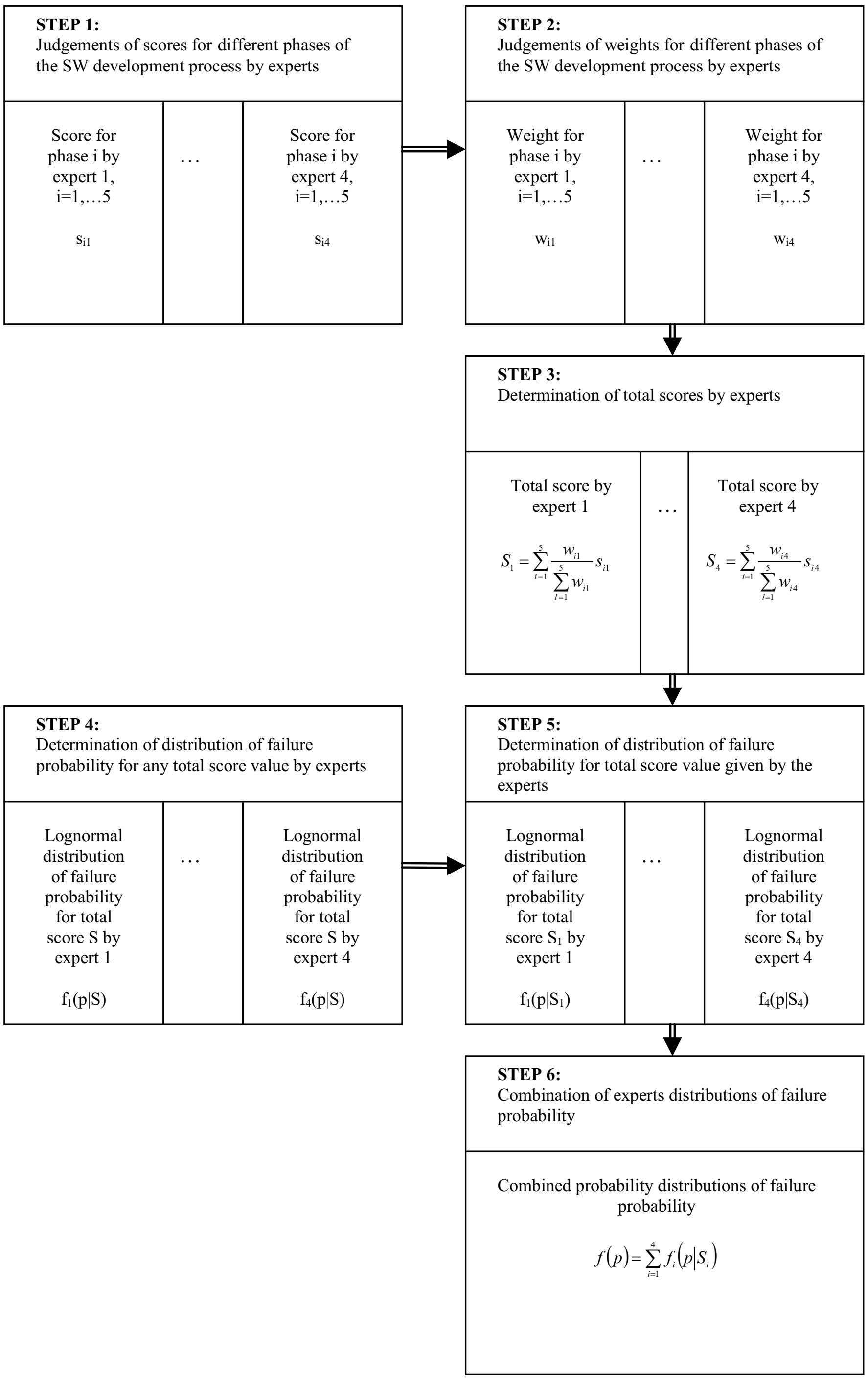

STEP 1:
Judgements of scores for different phases of the SW development process by experts
Score for phase i by expert 1, i=1,…5
s_{i1}
…
Score for phase i by expert 4, i=1,…5
s_{i4}
STEP 2:
Judgements of weights for different phases of the SW development process by experts
Weight for phase i by expert 1, i=1,…5
w_{i1}
…
Weight for phase i by expert 4, i=1,…5
w_{i4}
STEP 3:
Determination of total scores by experts
Total score by expert 1
$S_1 = \sum_{i=1}^{5} \frac{w_{i1}}{\sum_{i=1}^{5} w_{i1}} s_{i1}$
…
Total score by expert 4
$S_4 = \sum_{i=1}^{5} \frac{w_{i4}}{\sum_{i=1}^{5} w_{i4}} s_{i4}$
STEP 4:
Determination of distribution of failure probability for any total score value by experts
Lognormal distribution of failure probability for total score S by expert 1
$f_1(p|S)$
…
Lognormal distribution of failure probability for total score S by expert 4
$f_4(p|S)$
STEP 5:
Determination of distribution of failure probability for total score value given by the experts
Lognormal distribution of failure probability for total score S_1 by expert 1
$f_1(p|S_1)$
…
Lognormal distribution of failure probability for total score S_4 by expert 4
$f_4(p|S_4)$
STEP 6:
Combination of experts distributions of failure probability
Combined probability distributions of failure probability
$f(p) = \sum_{i=1}^{4} f_i(p|S_i)$

Figure 6. Diagram presenting the six steps of the expert judgement process.

Table IV. Fractiles for different score values given by the experts and μ and σ^2 parameter values of the most suitable lognormal distributions.

Score:	**0**				**5**				**10**			
Expert	p	X(p)	μ	σ^2	p	X(p)	μ	σ^2	p	X(p)	μ	σ^2
Expert 1	0,1	5·10e–5	–7,6	3,2	0,1	1,5·10e–5	–8,8	3,2	0,1	1·10e–6	–11,5	3,2
	0,5	5·10e–5			0,5	1,5·10e–4			0,5	1·10e–5		
	0,9	5·10e–3			0,9	1,5·10e–3			0,9	1·10e–4		
Expert 2	0,05	1·10e–5	–9,1	2,8	0,05	5·10e–6	–9,8	2,8	0,05	1·10e–6	–11,4	2,8
	0,33	5·10e–5			0,33	2,5·10e–5			0,33	5·10e–6		
	0,5	1·10e–4			0,5	5·10e–5			0,5	1·10e–5		
	0,67	3·10e–4			0,67	1,5·10e–4			0,67	3·10e–5		
	0,95	1·10e–3			0,95	5·10e–4			0,95	1·10e–4		
Scores:	**0**				**8**				**10**			
Experts:	p	X(p)	μ	σ^2	p	X(p)	μ	σ^2	p	X(p)	μ	σ^2
Expert 3	0,05	1·10e–3	–3,0	0,2	0,05	5·10e–5	–8,5	0,8	0,05	2·10e–5	–9,2	1,0
	0,5	5·10e–2			0,5	2·10e–4			0,5	1·10e–4		
	0,95	1·10e–1			0,95	1·10e–3			0,95	5·10e–4		
Expert 4	0,1	2·10e–1	–0,8	0,4	0,1	2·10e–4	–6,9	1,2	0,1	1·10e–4	–7,7	1,4
	0,25	3·10e–1			0,25	5·10e–4			0,25	2·10e–4		
	0,5	5·10e–1			0,5	1·10e–3			0,5	5·10e–4		
	0,75	7·10e–1			0,75	2·10e–3			0,75	1·10e–3		
	0,9	1·10e0			0,9	4·10e–3			0,9	2·10e–3		

the expert. The failure frequency fractiles and the mean and variance parameters of the most suitable lognormal distributions are shown in Table IV.

5.2.3 Step 5

Based on the total score value in the last column of Table III and the failure frequency distributions of different score values in Table IV a failure frequency distribution of the first system software version was determined for each expert by using the method of least square. The mean and variance parameters of the lognormal distributions obtained for the experts are shown in Table V.

5.2.4 Step 6

The last step was to combine the four failure frequency distributions given by the experts to one joint prior distribution reflecting the reliability of the first software version of the target system. The combination was carried out in the Bayesian network model and by the calculation algorithm of WinBUGS-program.

5.3 Operational experience

So far there had been seven software versions of SPAM 150 C. The operational experience of different software versions available in the assessment was the approximated amount of working years and the amount and types of software faults encountered with the different software versions. The approximated number of working years for each software version is given in Table VI. Based on the numbers of devices sold on each year the amount of working years for each software versions was estimated. The number of working years was calculated so that on the delivery year the devices were in operation only half the year i.e. the devices were delivered uniformly during the

Table V. Lognormal parameters of the failure frequency distributions of the experts.

	Prior:	
Expert:	μ	σ^2
Expert 1	–10,3	3,2
Expert 2	–10,8	2,8
Expert 3	–8,6	0,8
Expert 4	–7,4	1,3

Table VI. The approximated number of working years for software versions.

SW version:	**Working years per SW version:**
A	68
B	345
C	115
D	369
E	9805
F	2386
G	36747

Table VII. The number of software defects found for different software versions.

SW version:	Category 1	Category 2	Category 3	Category 4	Category 5
A	3	0	1	0	1
B	5	1	0	0	0
C	0	0	1	1	0
D	1	0	0	0	0
E	4	1	3	0	0
F	1	1	0	1	0
G	—	—	—	—	—

year. On the following years the devices sold before are in operation for the full year until a new software version was introduced. It was assumed that after commissioning the device was in operation full time around the clock.

In Table I the categories for software fault criticality were given. The number of software faults found in different fault criticality categories for different software versions is presented in Table VII. However, to give a better overall description about the nature of software fault and to simplify the calculation process of the assessment the categories of the faults encountered with different software versions were reduced from the original five categories to two groups. The software faults were put in one of the two groups depending on which of the five original categories the faults found fall into. Two groups were named as software inconveniences and software faults. Faults in categories 1 and 2 in Table I were considered as software inconveniences whereas faults in category 3, 4 and 5 were considered as software faults in the assessment. The quantity of software inconveniences and faults encountered for the different software versions are listed in Table VIII.

Table VIII. Software faults and software inconveniences encountered for different software versions.

SW version:	Number of SW inconveniences (categories 1–2):	Number of SW faults (categories 3–5):
A	3	2
B	6	0
C	0	2
D	1	0
E	5	3
F	2	1
G	—	—

5.4 Changes between software versions

When considering the independent changes made between successive software versions and their influence on the system reliability two different policies were taken in the assessment. In the first approach neutral attitude towards the changes was taken, which means that the prior mean value of change in failure probability between successive software versions was assumed zero. In the second approach a rather conservative attitude was taken and it was assumed that as a prior assumption a change in software has always a negative influence to the reliability of the system. What this means is that as a fault is removed from a software new faults are always introduced to the software. The magnitude of reduction in the reliability depends on the amount and criticality of the changes made to the software.

As in the previous section also the criticality of software changes was divided to two groups depending on which category in Table II the changes fall into. The groups were named as minor and major change made to the software. In the assessment a change from category 1 or 2 was considered as a minor change whereas a change from category 3, 4 or 5 was considered as a major change. The number of minor and major changes for each software version is listed in Table IX. Depending on the amount of minor or major changes made for a new software version a suitable prior estimation was built to reflect the change of system reliability the changes were believed to have. Parameters of the normal distributed priors used in the assessment to reflect the change between software versions are given in the last two columns of Table IX. The prior distributions are based on the expert judgement of the assessment executives.

Table IX. Amount of software changes made to each software version and prior estimations on the influence of software changes. μ and σ^2 are the parameters of normal distributions.

SW version:	Number of minor changes (categories 1–2):	Number of major changes (categories 3–5):	Prior (neutral approach):		Prior (conservative approach):	
			μ	σ^2	μ	σ^2
A	—	—	—	—	—	—
B	11	0	0	1	0,1	1
C	14	10	0	1	1	1
D	3	1	0	1	0,25	1
E	1	1	0	1	0,25	1
F	16	6	0	1	0,5	1
G	10	5	0	1	0,5	1

6 Results

6.1 Failure frequency

The parameters we are most interested in the network presented in Appendix 2 are the values of parameters *Lambda*, especially *LambdaG*, which reflects the failure frequency distribution of the last software version. Also the values of parameters *Omega* are some of interest, since they reflect the change in the software reliability between successive software versions. The more negative the distribution of parameter *Omega* is the greater enhancement has been achieved with a new software version. However, the results considering parameters *Omega* haven't been contemplated in more detail in the report.

Calculations with the Bayesian network were carried out for four different scenarios using all the evidence available at the time being. The scenarios differ by the approach used on the influence of software changes and by the data regarding the software faults encountered for the software versions. Different data sets were used for the conservative and neutral approach on the influence of software changes. In the first data set only the number of software faults in Table VIII encountered for different software versions were implemented to the network as fault data. In the second data set both the software faults and the software inconveniences in Table VIII were taken into consideration.

Using all evidence in the assessment the posterior failure frequency distributions of different software versions for the conservative approach is illustrated in Figure 7. In the figure are the posterior failure frequency distributions ranging from 2,5 percentile, the lower bar, to 97,5 percentile, the upper bar, and median marked as a dot somewhere in between. Corresponding graph for the neutral approach is given in Figure 8.

An approximation of the expectation value for the number of devices encountering a software fault during a year of operation can be obtained from the median values of distributions shown in Figures 7 and 8. The reader should, however, be reminded that median is not the same thing as expectation value even though in symmetrical distributions they concur. The posterior distributions in this case are not symmetrical and therefore median value gives only an approximation for the expectation value. For example from Figure 7 it can be estimated that for software version A it is expected that two devises out of thousand will encounter a software fault during a year of operation. For software version G the expected number is approximately four devices out of one hundred thousand during a year of operation. Here all the devices are assumed to function in a similar operational profile.

6.2 Predictive estimation

An interesting addition to the failure frequency estimation using the full evidence is how well the model predicts the failure frequency of a new software version when only part of the evidence is used in the assessment. Predictive estimation can be seen as a chronological estimation. The failure frequency distribution for a new software version is estimated only using the evidence available at that moment back in the history. Estimates below were calculated for all software versions using the same operational experience times as in the previous section.

Calculations for the first software version was carried out using only the first part of the network shown in Appendix 2, i.e. using the parts of network shown in Figures 3 and 4. The prediction calculation was performed by marking the data representing the amount of software failures of the first software version as missing. The realised calculation was carried out likewise but having the data representing the amount of software failures of the first software version implemented to the network. For the second software version only the first repetitive part of the network was attached to the previous network and same procedure was carried out for the second software version. Similar operations were repeated until for the last software version the network was the same as shown in Appendix 2.

The estimates were carried out for the conservative and neutral approach using the number of software faiults as data. The results of the calculations for the predictive failure frequencies are illustrated in Figures 9 and 10.

In Figures 9 and 10 the meaning of the approaches used for the influence of software changes can be seen in more detail. For neutral approach the median of the predictive failure frequency distribution of the new software version is the same as the median of the failure frequency distribution of the previous software version. For conservative approach the median is shifted upward by the amount determined in Table VIII.

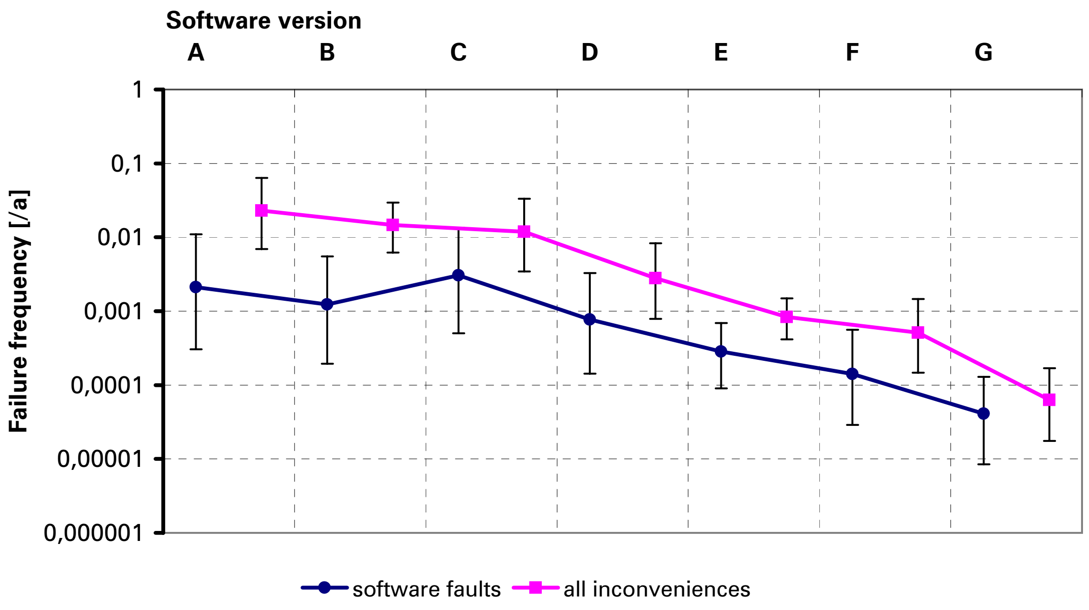


Figure 7. Posterior 0,025–0,5–0,975 percentiles for failure frequency distributions of different software versions for the conservative approach.

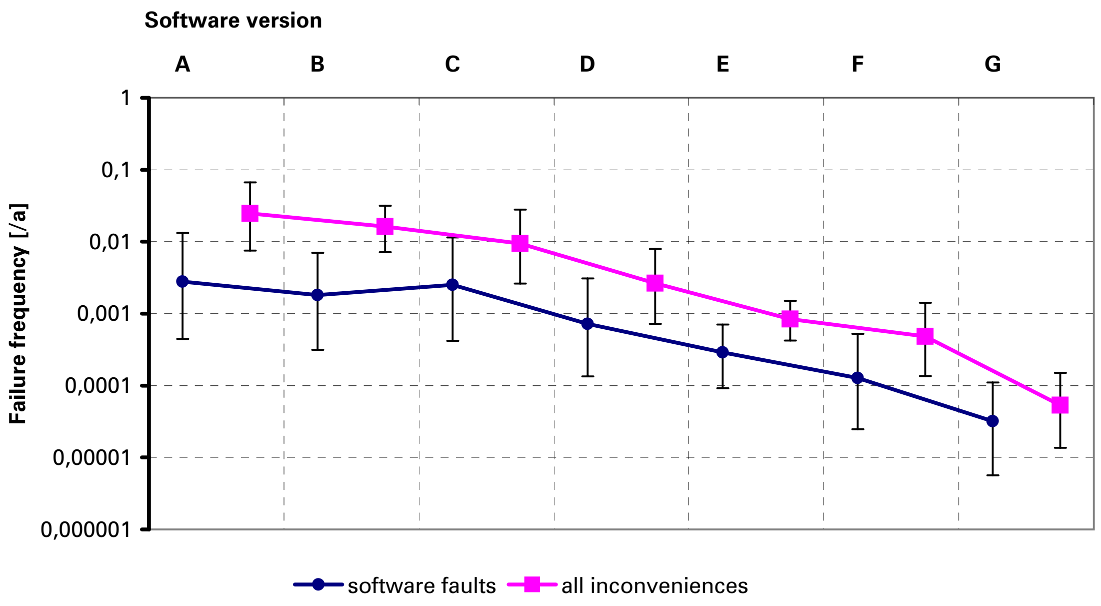


Figure 8. Posterior 0,025–0,5–0,975 percentiles for failure frequency distributions of different software versions for the neutral approach.

The differences in the variances of distributions are caused by the different amounts of data for different software versions.

6.3 Analysis of results

Significant differences between the posterior failure frequency distributions of the two approaches used for the influence of software changes can't be detected. With the conservative approach the failure frequency distributions of different software versions seems to be more monotonous, i.e. the estimates for the early software versions are better than in the neutral approach and vice versa for the later software versions. However, the difference between the two approaches for the failure frequency of the last and crucial software version is negligible as can be verified in Figures 7 and 8. Explanation to the small difference in the last software version can most probably be found from the amount, and therefore, the dominant role the data has reached in the assessment.

From the predictive estimates it can be seen

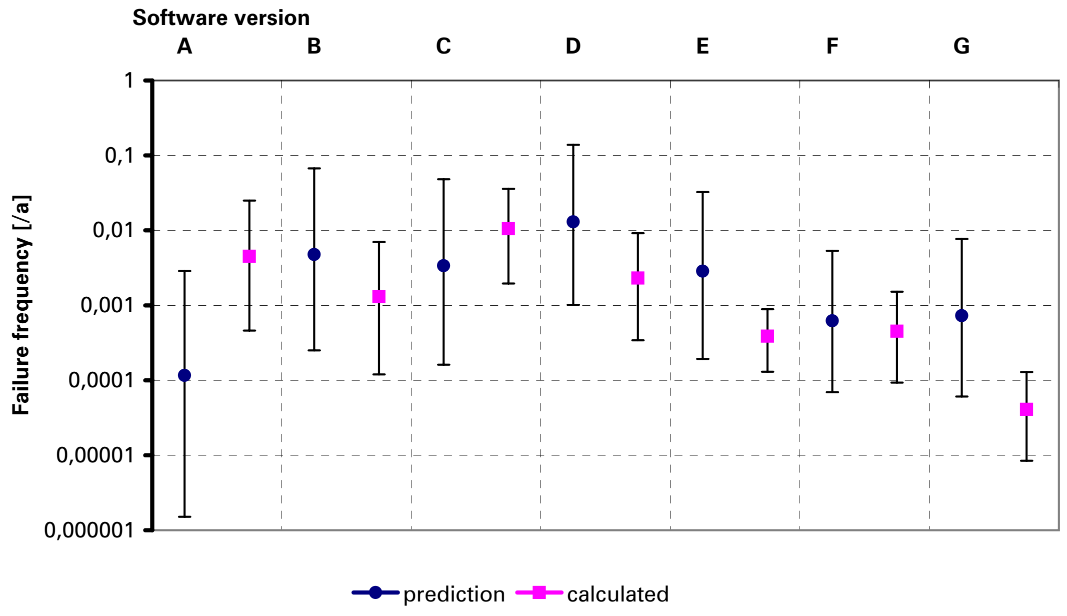


Figure 9. Predictive and calculated posterior 0,025–0,5–0,975 percentiles for failure frequency distributions of different software versions for the conservative approach.

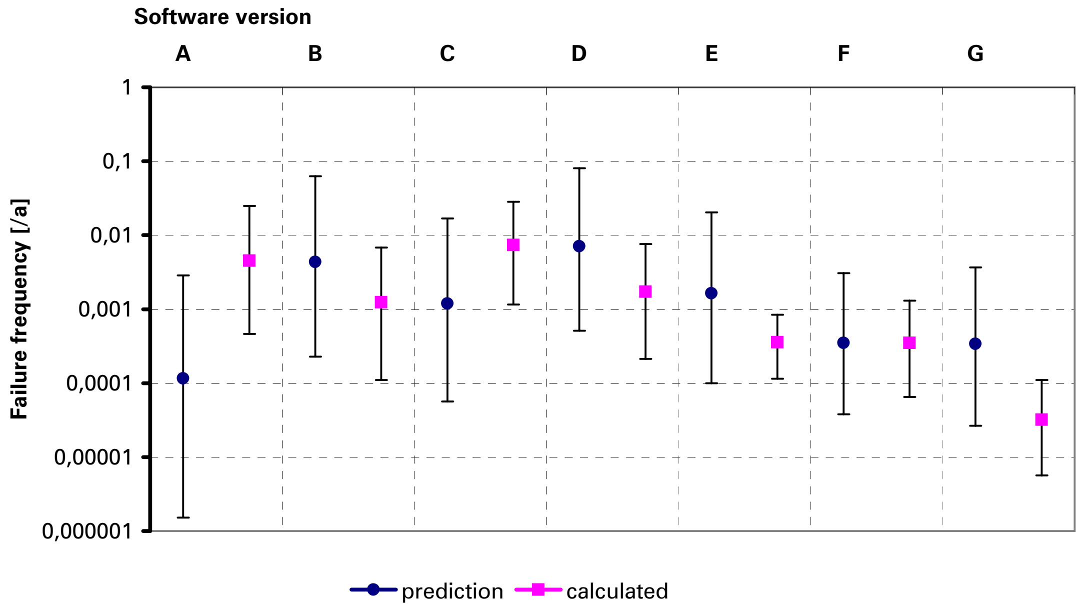


Figure 10. Predictive and calculated posterior 0,025–0,5–0,975 percentiles for failure frequency distributions of different software versions for the neutral approach.

that the prior estimates given by the experts are rather optimistic. Reasons for this can be found from the facts that the system under assessment is based on its predecessors and that the assessment was made long after the development of the system and information about the operational experience of the system was available to the experts. It is evident that these facts have strong influence to the estimates given by the experts, even though it was emphasised before the interview process to the experts that these matters should be excluded from the expert judgements. Interesting detail consolidating this belief is that the posterior failure frequency distribution of the last software version matches pretty nicely with the prior estimation of the first software version.

Another observation that can be made from the predictive estimations is that in spite of the optimistic prior for the first software version the conservative approach on the influence of software changes seems over pessimistic. There is clear statistical evidence that improvement between most of the software versions have been achieved.

7 Discussion

Before the actual interview process several conversations about the system under assessment were carried out between the assessment developers and the experts giving judgements about the system. Purpose of these conversations was to modify the assessment process suitable for the particular target system. One of the main goals in the interviews of the expert judgement process was to ground the questions asked and discussed during the interview to current guidebooks, recommendations and standards of developing high quality software-based systems. This wasn't an easy task since there weren't many practical examples on how to implement the valid recommendations and standards to concrete questions. A guidebook turning out to be useful in forming the interview questions was a guidebook published by the Finnish Society of Automation [9]. Especially the example documents at the end of the guidebook provided good basis for the questions in the interview. There is no doubt that the questions considered in the interview and presented in Appendix 1 would cover all the aspects of developing a high quality software-based system. Instead, the questions should be seen as a structured way of recalling the history and building an overview about the quality of the software-based system under estimation.

The comparison of the current standards for safety-critical software-based systems and the system under assessment was rather difficult matter. One main reason for this was that suitable standards for safety-critical software-based systems have mainly been drawn up after the development of the target system and therefore explicit considerations of such standards in the development phase of the system was not possible. Another main reason was that the motor protection relay SPAM 150 C wasn't explicitly considered to be a safety-critical application but only an application for which high reliability was required. Therefore, an important phase of explicit clarification of safety requirements for the system wasn't carried out in a sense recommended for example in safety standard IEC 61508. To ensure the correct operation and high reliability of the relay developer's own quality assurance methods were followed in the development process. Also the documentation involved with the development process was carried out in a way specified by the developer and not by a specific safety standard. All of the factors mentioned above had an influenced to the fact that the comparison of the answers and the documentation of the system development process to the requirements of current standards of safety-critical software-based systems was a rather troublesome matter.

The decision analytic method used in the assessment to build the experts prior estimation for the software-based system was applied in a rather informal way. Extra effort wasn't spent to the consideration of things like heuristics, biases and dependability of preferences in the questions and given answers. More analytical interview process would surely have been beneficial, but on the other hand, as mentioned already in the beginning the main emphasis of the work was on the methodological analysis of the reliability assessment and therefore an overview kind of interview process was sufficient for the assessment.

As mentioned in section 5.2 there were some trouble in converting the score values given by the experts to failure distributions of the system. Main reason for the inconvenience was the improper approach taken by the developers of the assessment method, which was corrected later in the assessment process. On the other hand it could be noted that more practice on giving the reliability estimation of a system in a probabilistic way would be appropriate. One way of achieving this goal would be by taking an assessment method as shown here to the regular activity in the software development process. In subsection 6.3 it was suspected that the expert weren't actually considering the reliability of the first software version, as they should have, but merely the latest software versions of the system. If this is really the case the results indicate that the experts have rather good intuition about the reliability of the system and therefore it is justified to use expert judgement in the reliability estimation of a software-based system. However, if the operational experience evidence is as strong as it is assumed in the case study the evidence from the expert judgements used for the building of the prior distribution has only a small influence to the final reliability estimation.

The most significant shortage in the assessment presented in the report is that the operational experience data used hasn't gone through a specific analysis. The reliability of software-based system is a property of the operational environment as well as the system itself. Although there may be faults in the software, these faults can cause a loss of safety function only when certain inputs occurring with very low probability are introduced to the system. In other words, the reliability of software-based system depends on the operational profile, which as the probability distribution of input sequences varies from one environment to another. To make the assessment more complete the operational experience data should be analysed in more specifics to be able to justify the results for a certain operational environment.

Other area of improvement in the network model presented is in the influence of software changes. The results shown in the previous chapter indicate that there was clear statistical evidence on improvement between most of the software versions and therefore the prior approaches taken, especially in the conservative case, was too pessimistic. Through finding a balance between the prior approaches and the actions made in changing the software it is possible to find the best practices for the software version management. The process for finding the best practices of software version management is a long learning process. The method described in the report can provide a good statistical approach for supporting the learning process of software version management and therefore supporting the development of high quality software.

8 Conclusions

The approach used in the report for the reliability estimation of software-based system seems extremely promising. Despite the slight inconveniences confronted in the assessment process with converting the score values given by the experts to corresponding failure probability distributions the experience on the assessment method is very encouraging. The experience and results obtained from the assessment process support our prior belief on the fact that one of the most suitable ways to estimate the reliability of software-based system involving diverse evidence from various kinds of sources is based on the use of Bayesian statistics. Especially for large systems where the system and the software contained by the system extends to the limits where the system can't be modelled or tested presumably completely it is reasonable to rely on such assessment methods as presented in this report. The example case study on a quantitative reliability estimation of a software-based system described in the report utilises mainly the evidence from the system development process and the operational experience of the system. Due to the limitation of the evidence sources the assessment presented gives a reliability estimation of the target system from a certain angle only. To obtain a full comprehension on the reliability of a software-based system a variety of different analyses like the one shown in the report should be practised.

The assessment procedure presented enable a flexible use of qualitative and quantitative elements of reliability related evidence. At the same time the assessment method is a concurrent way of reasoning one's beliefs and references about the reliability of the system. It is therefore justified to claim that a reliability estimation method shown here can provide a strong support for the licensing process of digital I&C systems. Most of all the reliability estimation method establishes a solid ground for the communication between different participants of a licensing process.

The assessment method shown in the report is everything but complete. Further research needs to be carried out particularly in developing the interview process of the assessment so that the questions cover all the relevant areas of developing a high quality software-based system. The questions and interview process should be formulated so that if necessary the grounds for the answers given by an expert can be traced back to the documentation of the system. The operational experience and the operational environments where the operational experience have been collected from is also a significant area of additional research effort.

References

[1] Helminen A. Reliability estimation of safety-critical software-based systems using Bayesian networks. STUK-YTO-TR 178. Radiation and Nuclear Safety Authority, Helsinki 2001: 1–23.

[2] Motor protection relay SPAM 150 C -product description. ABB Substation Automation Oy. http://fisub.abb.fi/products/bghtml/spam150.htm.

[3] Gelman A, Carlin JB, Stern HS, Rubin DB. Bayesian data analysis. Chapman & Hall, London 1995: 1–526.

[4] Box G, Tiao G. Bayesian inference in statistical analysis. Addison-Wesley Publishing Company, Reading 1972: 1–588.

[5] Pulkkinen U, Holmberg J. A method for using expert judgement in PSA. STUK-YTO-TR 129. Radiation and Nuclear Safety Authority, Helsinki 1997: 1–32.

[6] Korhonen J, Pulkkinen U, Haapanen P. Statistical reliability assessment of software-based systems. STUK-YTO-TR 119. Radiation and Nuclear Safety Authority, Helsinki 1997: 1–31.

[7] Spiegelhalter D, Thomas A, Best N, Gilks W. BUGS 0.5 Bayesian inference using Gibbs sampling manual (version ii). MRC Biostatistic Unit, Cambridge 1996: 1–59.

[8] Pulkkinen U. & Simola K., Bayesian Ageing Models and Indicators for Repairable Components, Helsinki University of Technology, Espoo 1999: 1–23.

[9] Ajo R, Hakonen S, Harju H, Järvi J, Kaskes K, Lenardic E, Niukkanen E, Nurminen T, Ritala P, Tolppanen M, Tommila T. Quality in automation—Best practices. Finnish Society of Automation, Helsinki 2001: 1–245. (in Finnish)

APPENDIX 1 QUESTIONS OF THE INTERVIEW

Project control and quality

- Did project have a quality plan?
- How would you describe the quality plan? What were the good/bad things in it?
- What kind of quality control methods did the project group have during the project?
- How well/badly were the quality control methods carried out in the project?
- Did the project have a project plan?
- How would you describe the project plan? Were all the main elements included in the project plan and how well did the project plan considered the needs of the system under development?
- How well/badly was the project plan carried out?

Requirement specification phase

- Was all the functionality of the system described in a functional specification?
- How would you describe the functional specification? What made it good/bad?
- Was the criticality of the system functions evaluated anyhow? How comprehensive was the evaluation? How were the results of the evaluation used? Was the possibility of the system functioning in safety critical applications considered?
- How well/badly would you describe the functional specification covered all the requirements set for the system in the user specifications and system requirements?

Design phase

- Was the system described in design specifications?
- How would you describe the design specifications? What made them good/bad? How comprehensive were the design specifications?
- What kinds of tools were used during the design phase? How would you describe the tools used? What made them good/bad?
- How well/badly would you describe the design specifications cover all the requirements set for the system in the functional specification?

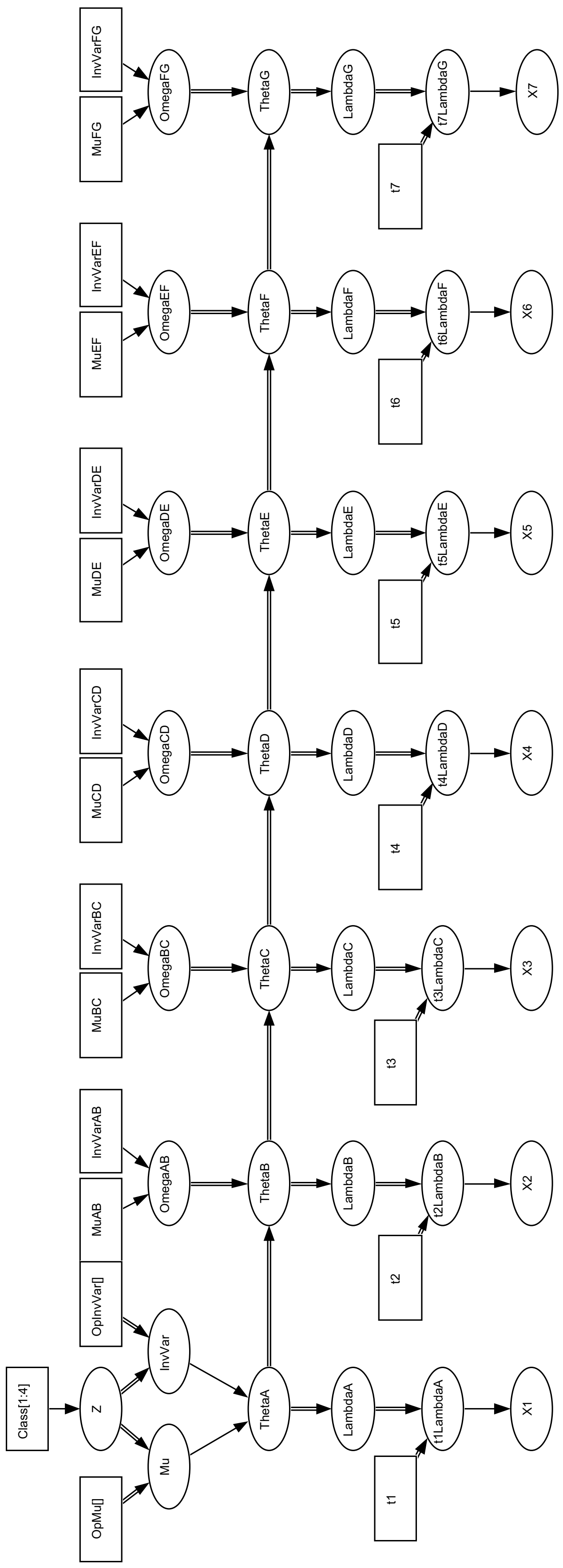
Implementation phase

- How well/badly was the system implementation carried out? What made the implementation good/bad?
- What kinds of tools were used in the system implementation? How would you describe the tools used? What made the tool good/bad?
- How well/badly did the implementation follow the design specifications?

Testing phase

- Did the system development process have a test plan?
- How well/badly would you describe the test plan covered the system implementation? What made the test plan good/bad?
- How well/badly was the test plan followed in the development process?
- When was the test plan generated? Were the test plan and the test cases in accordance with quality plan, functional specifications and design specifications?

APPENDIX 2 The Bayesian network in the assessment



APPENDIX 3 WinBUGS-code of the Bayesian network

```
model;
{
   Z ~ dcat(Class[1:4])
   Mu <- OpMu[Z]
   InvVar <- OpInvVar[Z]
   ThetaA ~ dnorm(Mu,InvVar)
   log(LambdaA) <- ThetaA
   t1LambdaA <- t1 * LambdaA
   X1 ~ dpois(t1LambdaA)
   ThetaB <- ThetaA + OmegaAB
   OmegaAB ~ dnorm(MuAB,InvVarAB)
   log(LambdaB) <- ThetaB
   t2LambdaB <- t2 * LambdaB
   X2 ~ dpois(t2LambdaB)
   log(LambdaC) <- ThetaC
   t3LambdaC <- t3 * LambdaC
   X3 ~ dpois(t3LambdaC)
   ThetaC <- ThetaB + OmegaBC
   OmegaBC ~ dnorm(MuBC,InvVarBC)
   log(LambdaD) <- ThetaD
   t4LambdaD <- t4 * LambdaD
   X4 ~ dpois(t4LambdaD)
   ThetaD <- ThetaC + OmegaCD
   OmegaCD ~ dnorm(MuCD,InvVarCD)
   log(LambdaE) <- ThetaE
   t5LambdaE <- t5 * LambdaE
   X5 ~ dpois(t5LambdaE)
   ThetaE <- ThetaD + OmegaDE
   OmegaDE ~ dnorm(MuDE,InvVarDE)
   t6LambdaF <- t6 * LambdaF
   log(LambdaF) <- ThetaF
   ThetaF <- ThetaE + OmegaEF
   OmegaEF ~ dnorm(MuEF,InvVarEF)
   X6 ~ dpois(t6LambdaF)
   log(LambdaG) <- ThetaG
   t7LambdaG <- t7 * LambdaG
   X7 ~ dpois(t7LambdaG)
   ThetaG <- ThetaF + OmegaFG
   OmegaFG ~ dnorm(MuFG,InvVarFG)
}
```