Bjørn Axel Gran

The use of Bayesian Belief Networks for combining disparate sources of information in the safety assessment of software based systems

Dr.Ing. Thesis

Department of Mathematical Sciences
Norwegian University of Science and Technology
2002

Preface

This thesis is submitted in partial fulfilment of the requirements for the degree "Doktor Ingeniør" (Dr.Ing.). The research has been carried out as a part of the long-term research within the OECD Halden Reactor Project, hosted by the Institutt for energiteknikk.

I want to acknowledge the different persons that have taken part in this research. First of all, I thank Gustav Dahll, which has been in-house supervisor, motivator, and taken active part in all the discussions behind this research. I also thank Harald Thumen and my other former and present colleagues in Halden, who have shown grate interest in the research.

I thank my supervisor at the Department of Mathematical Science Stian Lydersen for his support and guidance, and I thank my co-supervisor Tor Stålhane for both critical feedback as well as backing arguments. I enjoyed the year as guest in Trondheim. I am grateful to the rest of the project team that performed the "M-ADS project": Siegfried Eisinger from Det Norske Veritas, and Eivind J. Lund, Jan Gerhard Norstrøm, Peter Strocka, and Britt J. Ystanes from Kongsberg Defence & Aerospace AS (KDA). I also want to thank KDA for allowing me to further work applying their observations. I also thank Atte Helminen and his colleagues at VTT Automation for bringing in new ideas and co-operative work. Finally, acknowledge to Hugin Expert A/S for allowing me to use the HUGIN tool for my Ph.D.

At last, I thank Frauke for her encouragement, love and care.

Thesis Outline

The thesis consist of the following articles:

I. The Use of Bayesian Belief Nets in Safety Assessment of Software Based Systems.

   (with Gustav Dahll) In Special Issues of International Journal on Intelligent Information Systems at FLINS'98, *Int. J. General System*s, **24** (2), pp 205-229, 2000.

II. Assessment of programmable systems using Bayesian belief nets.

   Submitted and accepted for *Journal Safety Science*, Special Issue on Safecomp-2000. To be published 2002. (Extended version of the paper: Gran, B.A., Dahll, G., Eisinger, S., Lund, E., Norstrøm, J., Strocka, P., and Ystanes, B.: Estimating Dependability of Programmable Systems Using BBNs. *Computer Safety, Reliability and Security, Proceedings from Safecomp 2000,* (*LNCS 1943*), Koornneef F. and van der Meulen, M. (Eds), Springer, Berlin , pp. 309-320, 2000.)

III. The use of Bayesian belief networks for combining disparate sources of information in the safety assessment of software based systems.

   Submitted and accepted for *International Journal of Systems Science,* Special Issue on Intelligent Product Support Systems. To be published 2002.

IV. Applying Bayesian belief net in software safety assessment on a real, safety related programmable system.

   In *Safety & Reliability, Towards a safer world*. Zio, E., Demichela, M., and Piccinini, N. (Eds), Politecnico di Torino, Torino, pp. 1045-1052, 2001.

Appendix: EISTRAM - Experimental Investigation of the PIE-technique.

   (with Harald Thunem) In *Safety and Reliability.* Lydersen, S., Hansen, G., and Sandtorv, H., (Eds), Balkema, Rotterdam, pp 409-416, 1998.

Paper I presents the motivation for applying Bayesian Belief Networks for combining disparate sources of information in the safety assessment of software based systems. A part of this motivation is the experiences from the project presented in the appendix. Paper I also presents a first proposal for a BBN for "System Quality". Paper II has its basis in a project, in which the BBN-method was applied for evaluation of a real, safety related programmable system ("M-ADS"), developed according to the avionic standard RTCA/DO-178B. The results presented in the paper can be divided into two:

- the possibility to transfer the requirements of a software safety standard into BBNs; and
- the experiences with respect to the use of the BBN-method.

Paper III discuss some more on how to combine the Bayesian Belief Net method with the software safety standard for safety assessment of software-based systems. It also presents how the BBNs can be merged with a network, developed by VTT (Helminen 2000), representing evidence from disparate operational environments. This provides additional experiences on the applicability of the BBN methodology. Finally, paper IV describes some of the findings from the "M-ADS project", and discusses some of the results that were pinpointed as interesting, strange or counter-intuitive. It is natural to read paper I before the other papers, although they can be read independently of each other.

The following chapters give an overview of the work covered by the papers, and can be read on its own. All the sub-BBNs constructed with respect to how to combine the Bayesian Belief Net method with the standard RTCA/DO-178B are also presented. These have been left out in the papers due to the large number of BBNs.

Table of Content

1 Background

With the use of programmable equipment in safety critical systems a new aspect was introduced, to produce safe software. Therefore there is in many application areas necessary with a thorough safety assessment of the system, including intelligent product support systems, for a final acceptance or licensing of the system. In many application areas, including the field of nuclear power, law regulates this, and a safety case must be put forward for the licensing authorities for each safety critical application. A part of this safety case is the assessment of the reliability of the system.

1.1 Reliability assessment of software

For a hardware component, even in a safety critical system, it is accepted to assume that a failure can occur during the lifetime of the system, given that the expected frequency and/or consequence of the failure is sufficiently low. The reliability of a hardware system is thereby based on failure statistics, i.e. one measures the failure frequency in standard components and computes the system reliability on the basis of this, although that this practise may ignore the inherent faults in the hardware.

The characteristics of software make it difficult to carry out such a reliability assessment. Software is not subject to ageing, and any failure that occurs during operation is due to faults that are inherent in the software from the beginning. Any randomness in software failure is due to randomness in the input data. It is also a fact that environments, such as hardware, operating system and user needs, change over time. Furthermore, the software behaviour may change over time due to maintenance activities. As a consequence, there is a problem with the assessment and licensing of systems, both hardware and software, with inherent faults.

Various reliability growth models (Xie, 1991) have been suggested, but they are mainly applicable to large commercial systems, and not to safety critical software. The main reason is that a computer program implemented in a safety critical system presumably contains no known faults, since any revealed fault would be corrected. There is a possibility that it contains unknown faults.

1.2 Confidence in fault freeness

An alternative reliability measure is the confidence in fault freeness of the program, or more generally the upper limit of the “bug-size” (Voas et al. 1993, Gran and Thunem 1998). The PIE-technique (Voas 1992) is a dynamic failure-based technique for performing program sensitivity and testability analysis. The acronym stands for Propagation, Infection and Execution, which during the analysis are performed in reverse order, i.e. execution of a location, infection of the data state, and propagation of a fault to a discernible output. The PIE-technique is related to mutation testing and fault-based testing (Ramamoorthy and Bastani 1982, DeMillo and Offutt 1991), but while the purpose of most mutation testing techniques is to prove the absence of certain classes of faults, the purpose of applying the PIE-technique is to identify locations in a program, where faults, if they exist, are more likely to remain undetected during testing.

By applying the PIE-technique to the two larger test cases (Gran and Thunem 1998), we observed that the number of locations, which were likely to hide a fault during random

testing, was very high. Using several input distributions to test the mutants reduced the number of locations. However, the number of locations was still high, for the Power Range Monitoring of a nuclear reactor (PRM) program 66 out of 122 tested locations, and for the NEW_VTT program, a program that was developed in the Project on Diverse Software - PODS (Barnes et al. 1985, Bishop et al. 1986), 142 out of 300 locations.

The high number of locations that would be likely to hide a fault during testing means that one has a large number of pinpointed locations that are candidates for other testing methods or testing techniques. In this view one could conclude that the PIE-technique was not very efficient. On the other hand, the large number of "insensitive" locations could be an indication of fault tolerant programs, e.g. it can later be proven that the simulated faults will have no effect on the program.

We also wanted to compare the results from the PIE-analysis with the results from testing the PODS programs (Barnes et al. 1985). The back-to-back testing of the PODS-programs and the error seeding in one of the PODS-programs gave an indication of fault freeness. However, there was no guarantee of absence of hidden faults. Furthermore, the results depended upon the selected test input distributions and the number of tests. The PIE-analysis ended up with a high number of locations that would be likely to hide a fault during testing. This indicated that there might be hidden faults, and that one should decrease the confidence in fault freeness of the program. On the other hand, the large number of "insensitive" locations could also be an indication of fault tolerant programs, e.g. it can later be proven that the simulated faults will have no effect on the program. If this is the case, one has reasons to increase the confidence in fault freeness of the program.

Another problem with measuring the confidence in fault freeness based on statistical testing is that the validity of this measurement is highly dependent on a proper choice of test data (Leveson 1995). For the PRM program two different input distributions were applied, and for the NEW_VTT program five different input distributions were applied. In both cases it was observed that the effectiveness of the PIE-technique was improved in the sense that more locations became sensitive. However, it is uncertain to which extent the choice of input parameters and input distributions was representative with respect to the actual usage profile for the programs.

A final remark from this experiment is that it demonstrates the need of a useful way to combine different sources of information to produce a reliability figure. It should be able to make use of more information than traditional software testing techniques.

1.3 Rule based, risk based and judgement based safety assessment

There are various principles for how system safety assessment is performed. One can, however, classify these into some main types: *rule based*, *risk based (probabilistic)*, and *judgement based (expert judgement)* (Dahll and Gran 2000).

Rule based safety (also somewhat misleading called deterministic) assessment implies that an assessor checks that a system fulfils a set of criteria given in a safety standard. The rule based safety assessment approach is for nuclear safety based on two principles: "leak tight barriers" and the concept of "defence-in-depth". The principle of "leak tight barriers" is a basic strategy to prevent releases of radioactive materials. The "defence-in-depth" consist of taking into account all potential equipment failures and human errors, and it is applied in both the design and the operation phase. In the safety assessment it is assumed that accidents

may still occur. The systems are therefore designed and installed to ensure that the consequences of such accidents are acceptable for both the public and the environment.

This type of safety assessment has some advantages. The rules are easy to follow for the developer and easy to check for the assessor. On the other side, this method easily gets rigid and inadequate to handle new technology. The rules for safe software are normally based on consensus among experts of what is required for safety critical software. This is expressed through standards and guidelines.

In a *risk based* safety assessment the objective is rather to base the licensing on assessing the probability of potential risks associated with the system. The authorities, at least in the nuclear power area, often require probabilistic safety assessment (PSA). The objective is to check whether the probability of a major hazard is below a required limit. The first, and probably the most well known, probabilistic safety assessment was carried out in 1974 and is known as the Rasmussen report (Rasmussen, 1974). It provides the assessment of the potential risk of core damage for two power reactors.

One can in this respect distinguish between the frequentist's and the subjectivist's (or Bayesian) interpretation of the probability concept (Welsh, 1996). The frequentist's view on probability is best suited to measure properties of mass-produced components, of parameters where one has large statistical material, or with results from controlled experiments. This interpretation can be applied on the hardware components of a system, and basic rules for probability computation can be used to compute the probability of a hazard on the system as a whole (Leveson, 1995). The former interprets the probability as the measured frequency that a variable is in a specific state. The subjectivist, on the other hand, interprets probability as a (subjective) belief in the same. This belief can be supported or refuted by existing evidence.

The assessment of safety critical software is often faced with the problem of approving systems for which there are no clear rules, and for which it is difficult to apply probabilistic methods. The rules given in standards and guidelines are often imprecise, or they are not directly applicable for an actual system.

Licensing authorities are in many cases faced with the problem of approving systems for which there are no clear rules, as e.g. for safety assessment of computer based systems. One possibility for assessors and licensing authorities is to make their *judgement based* on the opinion of experts in various fields, including process knowledge, reliability engineering, human factors etc. The combined judgement of the different evidences about the system and its environment constitutes the basis for approval or not. Methods has been proposed to make reliability estimates based on expert judgements about information from different evidences, see e.g. research by Cooke (1991) and Pulkkinen (1994).

2 Safety assessment based on multiple evidences

The problem with the reliability measures is that they do not take into account that there are several factors that are important to software reliability (Dahll 1997), even if they cannot be put directly into a reliability formula. Some of these are of qualitative nature, like the producer’s reputation, the development quality etc. Others are measurable, but not directly connected to reliability estimation, like program size, program complexity etc. The connection between these quantities and software reliability is also of qualitative nature. It is suggested to apply traditional methods in probabilistic safety assessment (PSA) to software (Leveson 1995, Dahll, 1997, Cudleigh and Catmur 1992). As reasons for this choice it is

argued that these methods are well tried, standardised, documented and familiar to the customers (Stålhane 1997). Furthermore, it allows the customer to contribute with their knowledge about the system.

2.1 Evidences which influences a safety assessment

A combination of disparate evidences which influences a safety assessment is illustrated in Figure 1 in the form of an "influence net", i.e. a directed graph where each node represents an aspect in the total assessment process (Dahll and Gran 2000).

The top nodes in the graph represent the basic information sources that are used in the acceptance process. This information is penetrated through the net down to the bottom node. This represents the safety assessment, which is the main basis for a final acceptance of the system.

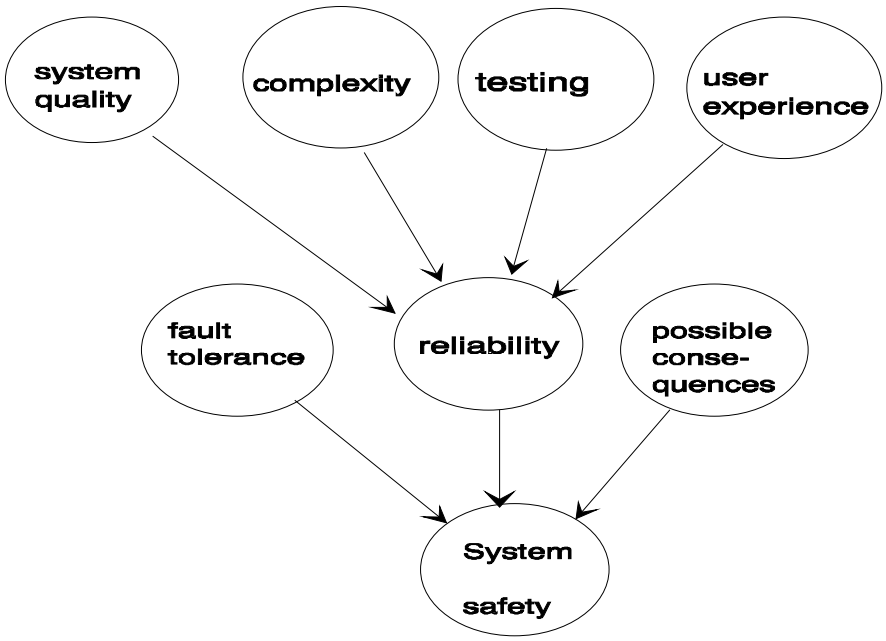

Figure 1: Influence graph of a safety acceptance and acceptance process.

The safety assessment is influenced by a reliability assessment of the system, as well as by an evaluation of whether a failure in the system will jeopardise safety. This evaluation can be achieved through a hazard analysis of potential risks to plant and environment. Safety defences (both against hardware and software failures) may be implemented as additional barriers against consequences of failures. A commonly used principle in this respect is diversity, i.e. the same functional goal is obtained through different means. The highest degree of diversity can be obtained if the same functional goal can be reached with completely different functions. This is often required to reach the safety goals of a safety critical system in a nuclear power plant.

The following sections discuss the basic information sources in more detail. In particular it is referred to the particular problem concerning assessment of commercial-off-the-shelf software (COTS). An important challenge in evaluating safety and reliability when reusing software is that the information available to the analyst usually will be different from what is normal. Typically, there will be more information from actual usage, while there might be less information regarding the software development.

2.2 Information about producer and development process

The avoidance of faults in the program is clearly related to the quality of the development process of the software system. A lousy made program can of course be correct, but a well-documented production procedure, in accordance with accepted standards, enhances the assessor's confidence in the reliability of the product. This confidence is also enhanced if the producer can document a history of producing high quality products.

To obtain a sufficiently high confidence in the quality, one should require that all parties involved in the development follow a quality assurance policy based on well-known standards for safety related systems (e.g. IEC publication 880). This may, however, be difficult when COTS software modules are concerned, since they are often delivered without appropriate information on the development process as well as on the final product itself. It may thus be difficult to assess whether the system has been developed according to the standards required for safety critical software.

2.3 Information about the programs

Detailed information about the software is needed to assess the reliability of its application. One aspect is to identify structural properties of the program that makes it vulnerable to programming errors. Complexity is obviously one of these, i.e. the more complex a module is, the more likely it is that it contains coding faults. Information about the complexity can be gained through an analysis of the program listings. However, for COTS software such listings are in general not available. It may be difficult to assess the complexity without this, but an indication on the complexity of the module can be seen from the complexity of the specification. A well-structured and comprehensible explanation of the use of the module is also an indication of a well-structured program.

A third aspect is the inherent complexity of the actual function itself. It is intuitively obvious that an adaptive controller is more complex to make, and therefore more error prone, than an AND gate, to take two extremes. One way to measure the inherent complexity of a module, where one does not have access to the source code, is to write it in a formal way, either as a program in high-level language, or as a formal specification, and then define a metric to measure the complexity.

2.4 Information about V&V and testing

A thorough verification and validation (V&V) and testing activity, at the module level as well as on the program as a whole, will increase the confidence in the program, and thereby its reliability. Information about the V&V activities can be obtained from various sources, as debugging reports, factory acceptance tests, site acceptance tests etc. An important information source, in particular for COTS software systems, is test data compiled during the development of the system, and during modifications before new releases.

To measure statement and path coverage for a test, one needs to know the program code in the software module. For a COTS module, however, the code is in general not available. An alternative is to make a coverage measure based on the specification, e.g. to measure the number of properties, or combinations of properties, which are checked by a certain test. If an oracle program has been made, an alternative is to instrument this with counters, and perform the coverage measurement on this.

2.5 Information about usage

The producers of COTS systems often use "proven design" as an argument for high reliability. This means that a wide range of users has used the system over a long period, with no, or few, reported faults. The idea behind this claim is that long user experience should reveal all inherent faults, if they exist. So if no faults have been reported over a long period, this should be a strong indication on error freeness.

The number of versions of a COTS system that is released is also relevant information. A new version implies changes in the system, and changes may have influence on its reliability. It is therefore relevant to know which changes have been made, or at least where the changes were made. In an actual application one should know whether any changes have been made in the software modules that are used in this application.

3 The BBN methodology

A more qualitative type of reliability measure is expressed as a subjective judgement, as a "belief" in fault freeness. The methodology proposed is to use the Bayesian Belief Network (BBN) methodology to combine the evidences from different information sources for a quantitative assessment of this belief. The objective of using BBNs in software safety assessment is to show the link between basic information and the confidence one can have in a system.

3.1 Applying BBN methodology for safety assessment

A literature survey on the BBN methodology (Chrisman, 1996) gives the impression that the main activities in this area up to then have been rather theoretical, and related to the AI area. However, there were also references to real applications medical diagnosis, geological exploration. The survey contained no references to the use of BBNs in safety assessment of programmable systems.

The SHIP (Safety of Hazardous Industrial Processes) project discusses, the possibility of applying BBNs in software safety cases and how to use formalised probabilistic safety arguments via BBNs (Delic, Mazzanti and Strigini, 1995, 1997).

More recently, it has also been applied to software safety assessment. Work in this area has been performed in the European projects SERENE (1999), IMPRESS (2000) and DeVa, in particular through previous research at the Centre for Software Reliability at City University, and present research at Queen Mary, University in London. The research has resulted in various papers, e.g. by Bertolino and Strigini (1996a, b, 1998), Neil et. al. (1996a, b, 1998, 2000, 2001), Fenton and Neil (1999) and Littlewood and Wright (1995, 1997). Ongoing work on this topic is also performed at VTT in Finland (Korhonen, 1997, Helminen 2000). This has also been the topic for research at the OECD Halden Reactor Project (HRP) (Dahll and Gran 2000).

3.2 Background

The Bayesian Belief Networks methodology was introduced in the 1980s, and is in particular described in the book by Pearl (1988) and the paper by Lauritzen and Spiegelhalter (1988). In 1993 the tool HUGIN (Aldenryd, Jensen and Nielsen 1993, Jensen 1996) was introduced, which made BBNs feasible. The theory, however, is based on the Bayes rule, discovered by

Sir Thomas Bayes (1744-1809) which says for two variables X and Y that P(X|Y)= P(Y|X)*P(X)/P(Y). By allowing $\{X_i\}$ be a complete set of mutually exclusive instances of X, this formula can be extended. A description of Bayesian interference, Bayesian network methodology and theory for calculations on BBNs can also be found in the books by Gelman et al. (1995), Welch (1996), Cowell et al. (1999), the report by Pulkkinen and Holmberg (1997), and older references such as Whittaker (1990), and Speigelhalter et al. (1993).

A BBN is a connected and directed graph, consisting of a set of nodes and a set of directed arcs (or links) between them. Uncertain variables, both events and singular propositions, are associated to each node where the uncertainty is expressed by a probability density. The probability density expresses our confidence to the various variable outcomes, and depends conditionally on the status of the "parent" nodes at the incoming edges. The nodes and associated variables can be classified into three groups:

- Target node(s) - the node(s) about which the objective of the network is to make an assessment. A typical example of such nodes is "No faults in a program".
- Intermediate nodes - nodes for which one have limited information, or only "beliefs". The associated variables are the hidden variables. Typical hidden variables represent quality aspects such as "development quality", "producer's reputation", or "quality at a certain stage of the development" without discussing "quality" in detail.
- Observable nodes - nodes that can be directly observed. Some examples are nodes representing observable properties about the system for evaluation: "no failures during testing" and "all quality requirements are fulfilled".

Application of the BBN method consists of three tasks:

- construction of BBN topology;
- elicitation of probabilities to nodes and edges; and
- making computations.

3.3 **The construction of BBN topology**

The literature on BBNs has mostly presented small "complete" BBNs (Neil et al. 2000). The construction of small BBNs can be made gradually. Information about the system is collected and expressed via the nodes. The nodes are connected to a directed graph that expresses the conditional relationship between the variables. The aim is to combine the information in the net. One way is to start from a target node and draw edges to influencing nodes. To decide the direction of an edge, one can follow the causal direction (Dahll and Gran 2000). However, this direction is not always obvious, in particular between nodes representing qualitative variables. In these cases the direction of the arrow often goes from higher abstraction to lower abstraction, or from the more general concept to the more detailed. A general interpretation of an arrow between two nodes A and B is that a "belief" in A implies expectations on B. The practical procedure is to start with constructing a BBN, containing nodes representing high-level information.

When building larger-scale BBNs this procedure is rather effort consuming. Neil, Fenton and Nielsen (2000) offer a solution based on building blocks (idioms), which serve solution patterns. These can then be combined into larger BBNs. This approach is applied in the SERENE project (1999), and has been applied to construct large-scale BBNs for predicting software safety. The use of idioms is also applied for the construction of the BBNs presented

in the next chapter. However, the BBNs are not of such large-scale, so it is also possible to argue through the "causal direction approach".

3.4 **The elicitation of probabilities**

The second step is the elicitation of probability distribution functions (pdfs) to the nodes and edges. To begin with, one gives prior pdfs for the top nodes, and conditional pdfs for the influences represented by the edges. These pdfs may be either continuous functions or they have a discrete form. The latter means that the ranges of the variables are divided into finite number of states.

The advantages of the pdfs in discrete form are that it becomes conceptually easier in an expert judgement to assign discrete values, and that it makes the computation simpler. The conditional probabilities for edges between discrete variables are given as conditional probability tables between the states of the variables associated with the start node and the end node of the edge respectively. However, since many of the aspects to be considered are of qualitative nature and not directly measurable estimation may be difficult. This was observed for the co-operative project between the Halden Project (HRP), Kongsberg Defence & Aerospace AS (KDA) and Det Norske Veritas (DNV), even if some of the project members can be considered as experts in their fields (Gran et. al. 2000). It is therefore highly recommendable to make use of some expert judgment tools or expert judgment expertise. Another observation was that the establishment of the BBNs and prior conditional pdfs was rather time consuming.

The problem of defining the node probability tables is also addressed by Neil, Fenton and Nielsen (2000). They apply a "divide and conquer" approach to build the BBNs. This manages the complexity of the BBNs, and thereby reduces the number of probability values to be addressed.

3.5 **Making computations**

Making computations with BBNs above a certain size and complexity is rather difficult by hand, but is easy by applying the latest computerised tools. At HRP the HUGIN tool (Aldenryd, Jensen and Nielsen 1993) has been used, and in the "M-ADS" project both the HUGIN tool and the SERENE methodology (1999) were applied.

The computation of our belief about a specific node (target node) is based on the rules for conditional probability calculations given by the Bayesian methodology. The procedure is to insert observations in the observable nodes, and then use the rules for probability calculation backward and forward along the edges, from the observable nodes, through the intermediate nodes to the target node, which again can be an intermediate node in a BBN at a higher level. Forward calculation is straight forward, while backward computation is more complicated (Spiegelhalter et. al. 1993). For details on computations see the references in the beginning of this section, and for good examples on making computations with BBNs see for example Pearl (1988) and Jensen (1996).

4 A BBN for System Quality

A first attempt to construct a BBN for safety assessment, (Dahll and Gran 2000), was based on the "influence net" given in Figure .1. An extended version of this influence graph can be found as the "safety acceptance and acceptance process of the software" (Dahll 2001). These

are not themselves BBNs, but quite similar, so it was fairly straightforward to construct a high level BBN for "system quality" based on this (Dahll and Gran 2000), see the BBN shown in Figure 2.

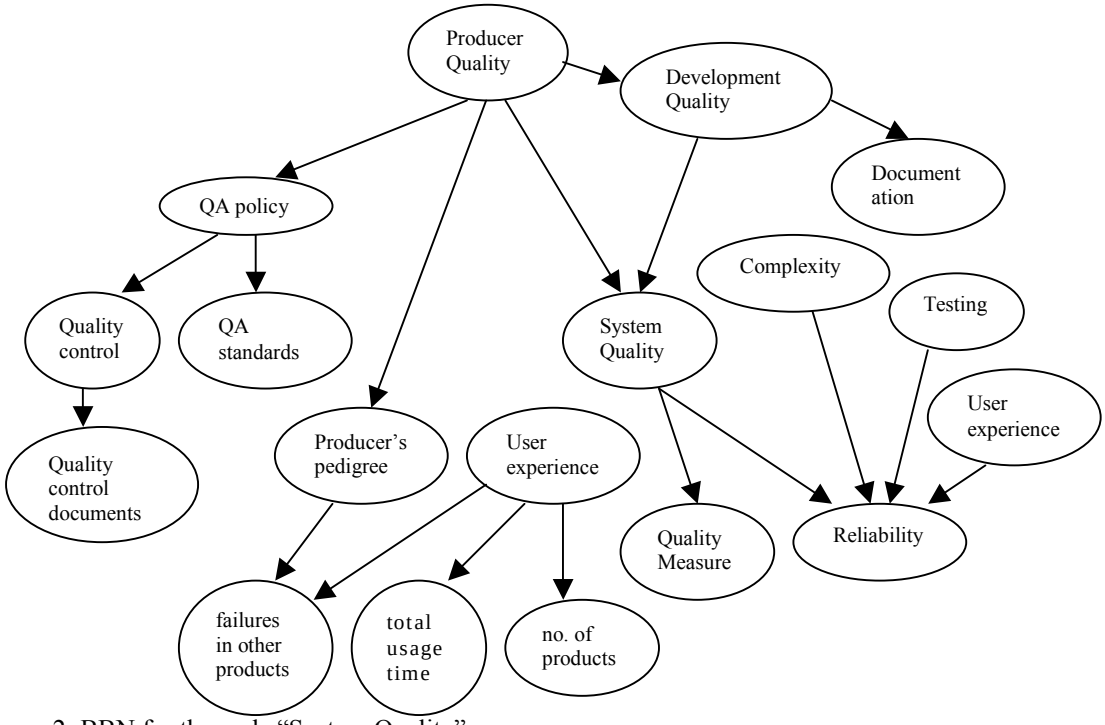

Figure 2: BBN for the node "System Quality".

The BBN was constructed gradually by applying the causal approach, combining the target node(s) with the observable and the intermediate nodes. The aim was to combine all available relevant information into the net. One problem, however, was to decide when to stop, i.e. how much details does one want to have in the BBN?

The highest node in this figure is the "producer quality". This is a hidden variable representing a fairly abstract quantity, which manifest itself through the variables it influences in adjacent nodes. The producer quality has a direct influence on the system quality, as indicated by an arrow. But this influence can also be seen indirectly, through the process by which the system is actually developed. This is shown by the edge to the node "development process", which again has an edge to the node "product quality". It has, however, also an edge leading to the node "documentation". This should be interpreted such that the quality of the development process influences the quality of the documentation from the development. The latter is an observable, and one may put some kind of measures on the documentation quality. Evidences about the system quality are quality attributes such as readability, structuredness etc. These are grouped into one node called "quality measures". This node could, however, be expanded further.

Another edge from "producer quality" leads to the node "QA policy". The arrow of the edge between them could be expressed as: "a producer of high quality is likely to have a good Quality Assurance (QA) policy and use recommended development methods". The further argument is that a good QA policy implies that accepted QA standards are followed, and this can be observed. It also implies that a strict QA control is followed, and this may be observed through the QA control documentation, which is also observable.

The producer quality also has impact on the reliability of other products from the same producer. This again will clearly influence the number of failures observed in these products, which can be directly measured. However, the number of failures found in these products are clearly depending on the amount of user experience with these products, i.e. the more these products are used, the higher is the likelihood that any inherent faults in the product will be revealed through an observed failure. This user experience can be observed through user reports, which is an observable node.

4.1 Computation on the BBN for system quality

To demonstrate the computation on a larger BBN the BBN for system quality was selected. Notice that this was intended as an illustration of the method, and not as a real attempt to compute the quality of a system. The computations were based on discrete variables and the use of the HUGIN tool.

The first step was to divide the variables associated with each node into discrete states. To limit the size of the problem, there was a maximum of three states for each variable. The states of the nodes were selected as given in table 2 in (Dahll and Gran 2000). The target node was selected to be the "System Quality". The observable nodes were: "Quality-control-documents", "QA-standards", "Failures-in-other-products", "Number-of-products", "Usage-time", "Documentation" and "Quality-measures". All assignments of values to the variables and relation matrices were chosen somewhat ad hoc, i.e. reasonable for an illustration, but not based on any deeper analysis. The prior values for all the nodes are given in the appendix in (Dahll and Gran 2000).

By placing findings on the "Number-of-products", the pdfs for the "User-experience", "Usage time" and "Failures-in-other-prod." changed, but the findings had no effect on the rest of the variables. Similar observation was made for findings at "Usage-time". This is in accordance with the conditional independences observed. By placing findings on the "QA-standards", the pdfs for the "System quality" changed somewhat, but not as much as when findings were placed on the "Documentation" and "Quality-measures". This is in accordance with what one should expect, and also in accordance with the independence graph where e.g. the "Quality-measures" is directly connected to the "System Quality". By placing findings on the "Failures-in-other-products", the pdfs for the "System Quality" changed in opposite way as described above. This observation is not obvious by only observing the influence graph.

The next step was to observe the BBN in the case of several findings at once. This was done in two cases, assuming all observable variables to be in their *worst* state and in their *best* state. The results did not show any unexpected results, except for the "Producer's pedigree" which had approximately similar results in the two cases.

4.2 Conclusions from computation on the BBN for system quality

The evaluation of the test case showed how a finding on one or more specific observable variables would change the belief in a hidden explanatory variable such as the target node "System Quality". The evaluation also showed the effect of conditional dependence and independence between variables. Further, the test case indicated that the HUGIN tool is suitable to be used in the calculations of a complete realistic test case. On the other hand, applying the BBN-methodology required that probability density functions were assigned for all variables, something that requires the use of expert judgement and collection of real data.

5 Standards and Guidelines for Safety Related Software

Recently much effort has been taken to make international standards and guidelines for the development of programmable systems for safety related applications. A generic standard is IEC 61508 “Functional safety of electrical/electronic/programmable electronic safety-related systems” (IEC 61508). This standard will constitute a framework for other, more specific standards. Examples of branch specific standards are IEC-880 (IEC 880), IAEA software safety guide (IAEA ID NS 264) in the nuclear industry, and RTCA/DO-178B (1999) for safety critical software in civil aviation.

A general impression from these standards is that they are built on the same basic framework, and follow the same principles, although they may differ in the aspects they put special emphasis on. The common framework is expressed in a software lifecycle model, where the different stages in the system development are placed. For each of these stages requirements or recommendations are given. The division into stages, and the starting and end stages of the lifecycle model, may differ between the standards. The standards also differ in the requirements they are particularly emphasising. Even if different standards vary in the degree of detail, a general characteristic of software standards is that the requirements and recommendations are of qualitative nature, in distinction from hardware standards where there in general are clear and objective requirements. Ideally a requirement in a standard should be objective in two ways: the requirement itself should be objective in an unambiguous way, and there should be an objective way to state whether the requirement is fulfilled or not. This problem is thoroughly discussed in (Neil and Fenton 1998).

A question in connection with software safety standards is whether the fulfilment of their requirements actually guarantees that the system is safe. A standard is in general developed, over a long time period, by a group of experts. Other experts around the world then review the draft international standards. Such a thorough preparation by internationally renowned experts should strongly indicate that a system made according to this standard is safe. There is, however, no objective evidence that guarantees that this is true. Even the views of experts are to a large degree based on judgement. These judgements also need to be calibrated, which is an activity that dependents upon that the experts receive feedback on his/her judgement. In addition, the experts in this field constitute a fairly limited society, so it is likely that they are strongly influenced by each other.

Of course, the safety assessment is not necessarily based on qualitative judgement only. There are analytical methods like e.g. fault tree analysis, reverse engineering, formal verification, etc., as well as statistical reliability evaluation based on operating experience or testing. Testing is essential for a safety assessment of the final product. A general impression is, however, that the standards are not very precise on required strategies for testing, but leave this to human judgement.

A conclusion from these considerations is that it is not straightforward to decide objectively whether a software-based system is sufficiently safe on the basis of the criteria given in a standard only. There is a need for a systematic decision support system associated with a standard, which can help the licensing authority or any safety assessor. It is suggested that Bayesian Belief Nets and associated tools can provide this help (Gran 2002 Safety Science).

6 BBNs based upon RTCA/DO-178B (the "M-ADS project")

The attempt to combine the Bayesian Belief Nets methodology with the rules of a standard for safety critical software, RTCA/DO-178B (1999), hereafter referred to as DO-178B, was done in an experimental project carried out by a consortium composed of Kongsberg Defence & Aerospace AS (KDA), Det Norske Veritas (DNV) and the Halden Project (HRP). The project goal was to evaluate the use of BBN for investigating the implementation of the DO-178B standard for software approval in the commercials world. To reach that objectives a computerized system for automated transmission of graphical position information from helicopters to land based control stations (M-ADS) was selected and studied (Gran et. al. 2000, Gran 2002a). Please note that references to the system developed by KDA and conclusions here represent by no means any official policy of KDA.

6.1 **RTCA/DO-178B**

The purpose of the DO-178B standard is to provide guidelines for the production of safety critical software for airborne systems. This guideline was chosen for the study since the M-ADS system is applied in civil aviation, and was previously qualified on the basis of this standard. DO-178B discusses aspects of airworthiness certification pertaining to the production of software for airborne systems and equipment used in aircraft. To aid in understanding the certification process the system life cycle is briefly discussed to show relationship to the software life cycle process. DO-178B does not provide guidelines concerning the structure of the applicant's organization, relations to suppliers and personnel qualification criteria.

DO-178B defines a set of five software levels (A to E), based on the contribution from software to potential failure conditions as determined by the system safety assessment process. The main recommendations in DO-178B are given in a set of 10 tables, see table 1. Each table relates to a certain stage in the development and validation process, and contains a set of objectives. A difference between the DO-178B and e.g. IEC61508 is that most of the requirements are mandatory in the latter, while the requirements are guidelines in DO-178B (Neil and Fenton 1998).

Table 1: The stages in the development and validation process given by DO-178B

	Stage in the development and validation process
A1	Software planning process.
A2	Software development process.
A3	Verification of outputs of software requirements process.
A4	Verification of outputs of software design process.
A5	Verification of outputs of software coding & integration process.
A6	Testing of outputs of integration process.
A7	Verification of verification process results.
A8	Software configuration management process.
A9	Software quality assurance process.
A10	Certification liaison process.

6.2 The M-ADS Airborne Equipment

The M-ADS airborne equipment was designed by KDA for installation in helicopter aircrafts (Gran et. al. 2000). The system provides air traffic services transmitting aircraft parameters upon request from the air traffic control where personnel will request positioning data. The M-ADS system is designed to automatically transmit flight information via data link to one or more requesting air control centres. M-ADS uses existing avionics on board the aircraft to provide aircraft position, speed and additional optional data. The most important data are the aircraft position, position accuracy, altitude and time stamp for the data validity. The main purpose of the M-ADS Airborne Equipment is to aid in a rescue operation if the helicopter has made an emergency landing on the sea. A correct localization is necessary for a successful rescue operation, the system is therefore safety critical, and the system had to be approved by the Norwegian Civil Aviation Authority. The software development process was performed according to the DO-178B standard.

6.3 The construction process

The basic philosophy of the proposed process is to relate the safety of the system to the fulfilment of the requirements in an internationally accepted safety standard. This philosophy can of course be questioned, but such standards are based on consensus among experts in the area relevant for an actual safety critical system. Even if conformance to a safety standard does not imply safety, it is a strong indication of the effort put into making the system safe. This indication can also be used as prior probability in a Bayesian model for a further safety assessment based on safety testing. Recall that one want to achieve a way of stating how well the development of a safety critical system conforms to the requirements of the standard. However, such standards do not contain any measures of conformity, but rather a large number of requirements of rather disparate nature, which should be fulfilled. The objective of the M-ADS project was to use BBN methodology to construct such a measure.

The first action in the construction is to identify the main characteristics that may influence the dependability of a system. One can distinguish between characteristics that are related to the system itself and characteristic that are related to the interaction between the system and its environment (usage of the system, potential hazards etc.). The former includes quality characteristics, which are divided into four types:

- *Quality of the producer*. (Qproducer) This includes the reputation and experience of the producer, quality assurance policy, quality of staff etc.
- *Quality of the production process*. (Qprocess) A high quality implies that the system is developed according to guidelines for good software engineering, that all phases are well documented, and that the documentation shows that the system at all development phases possesses desirable quality attributes as completeness, consistency, traceability etc.
- *Quality of the product*. (Qproduct) This includes quality attributes for the final product, as reliability, simplicity, verifiability etc.
- *Quality of the analysis*. (Qanalysis) This includes all activities performed to validate the correctness of the system during all stages of the system development. Such activities may include model checking of the specifications, inspections and walkthroughs of the documentation, static analysis of code and testing of the system.

The next step is to construct the BBN in two levels. The higher level shows how nodes representing the four types of characteristics listed above are combined with other nodes in

the net and lead to nodes representing the reliability and safety of the system. At the lower level there are four BBNs, where the four characteristics are represented as top nodes.

6.4 The higher-level BBN

The higher-level network consists of two parts: the *quality-part* (or soft-evidence part) and the *testing-part*, as presented in Figure 3, (Gran 2002a).

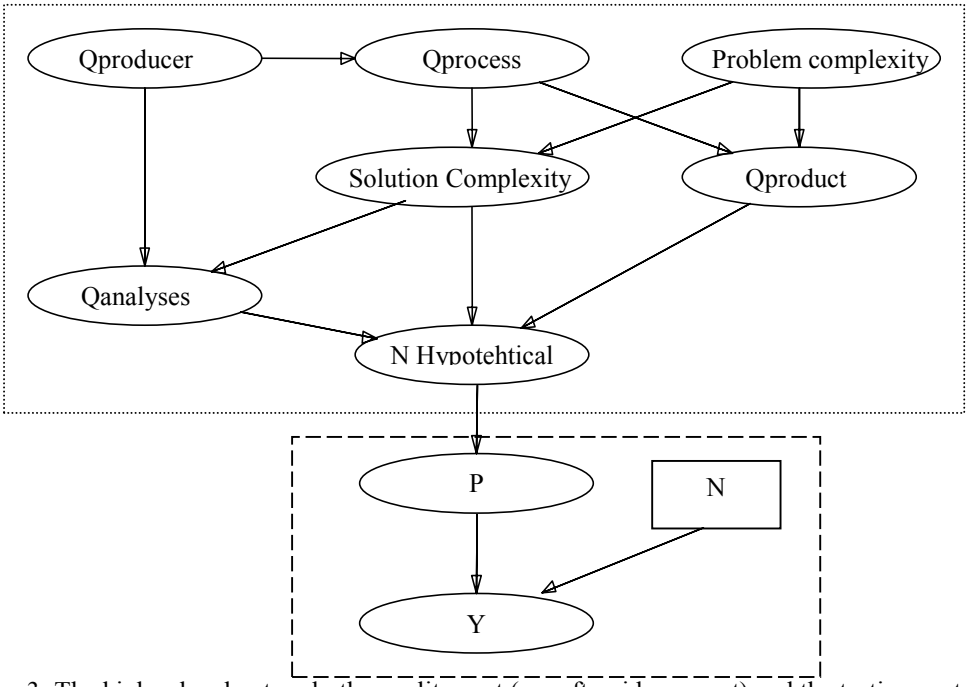

Figure 3: The higher-level network: the quality-part (or soft-evidence part) and the testing-part

The *quality-part* consists of the four quality nodes listed in the previous section. In addition it includes the nodes "*problem complexity"* and "*solution complexity"*. The initial nodes or top nodes are the quality node *Qproducer* and the *problem complexity*, where the latter is an attribute of the system to be developed, and can be measured. It is assumed that the *Qproducer* directly influences the *Qprocess*, and that the *solution complexity* is influenced by the *problem complexity* and the *Qprocess*. The same dependencies are assumed for the *Qproduct*. The product quality depends upon how difficult it is to fulfil the requirements (the complexity of the problem), and upon the ability of the development process to handle complex systems. The *Qanalysis* is assumed to be influenced by the *Qproducer*, how well prepared the organization is to perform an analysis, and the *solution complexity*, how difficult it is to analyse. All these assumptions are based on the BBNs presented in the SERENE project and in accordance with networks for system quality (see chapter 5).

The higher-level network leads to an end node *N-hypothetical*. The intention is to express that the information in the quality-part is equivalent to that the system is tested with *N* randomly chosen inputs without failure.

The *testing-part* represented by the node "P *Y*: failures in *N* new tests", describes the connection between hard evidences, *Y=0* failures in *N* tests, and the failure probability of the system (in the context, usage, environment, etc. the system is tested). The failure probability can be interpreted either as a number of failures on a defined number of demands, or as a

number of failures on a defined time period. For the defined number of demands *N* with the constant failure probability *P* the random number of failures *Y* has a binomial distribution.

The failure probability *P* can be linked to a node representing the system safety, which in addition is also depending on the usage of the system and the consequences of eventual failures. In the described project no modelling of the dependencies with respect to the system safety was made. Of this reason these nodes are not included in Figure 3, and no calculations related to this node were done.

The link between the *quality-part* and the *testing-part* is given by the edge between *N-Hypothetical* and *P*. The dependency associated with this edge, leading to the results presented, was given by "*P = 1/ N-Hypothetical"*. However, it was applied in the way that P(*P*∈[p,q]) = P(*N-Hypothetical* ∈[1/q,1/p]). The same dependency would have arisen by assuming direct dependencies between P and the nodes *Qanalysis*, *Solution Complexity* and *QProduct*. For the expert team it was, however, conceptual easier by this two-step procedure.

An alternative BBN for the *quality-par*t is to replace the node *N-hypothetical* with a node representing the "P(failed state)" directly, as presented in Figure 4 (Gran 2002b). This node is not to be viewed as a failure rate representing a specific usage or safety function, but rather as a deterministic property of the system expressing fault content. One interpretation is the size of the inherent faults in the software. Assuming that no failures are found or modifications are made during later testing of the system, this true failure rate is not changed; only the confidence in the reliability, or freeness of faults, of the program is enhanced. Thereby it also offers a support in the assessment of the software.

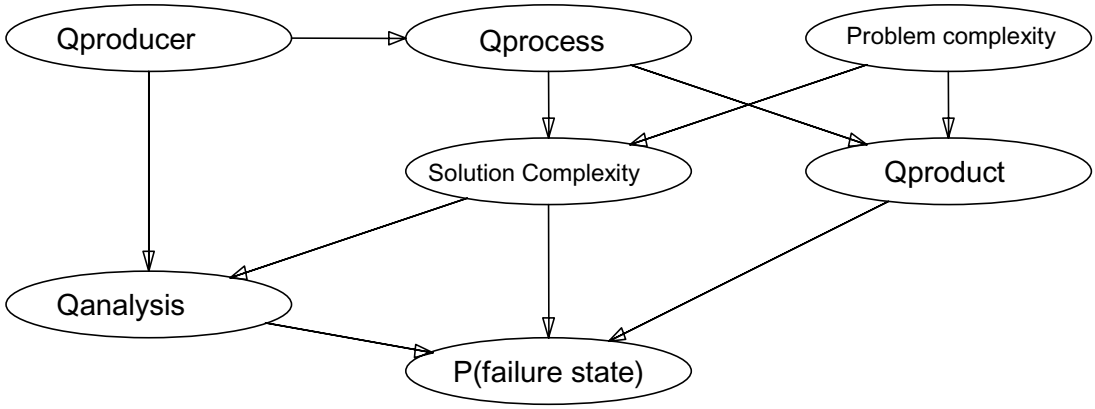

Figure 4: The upper network for DO-178B

6.5 **The construction of BBNs on the lower level**

At the lower level there are four BBNs, one for each of the four quality aspects, with the quality aspects as top-nodes in the BBNs. Each top node is then linked to intermediate nodes representing the 10 lifecycle stages represented by the tables A1 to A10 of DO-178B. Each of these nodes are again linked to other intermediate nodes, representing the objectives of the tables. The four BBNs are presented in Appendix A. (Remark that these figures are the ones generated by the HUGIN tool, and contain misprints that are not corrected in accordance with the text in this report. The text in the nodes is also amputated due to the selection of the node size, which has to be equal for all nodes.)

The associating of the different objectives to the different quality aspects can be done by a group of experts, consisting of experts related to the standard itself, development in accordance with the standard, and experts within safety assessment of critical systems. In the M-ADS project each objective was identified to belong to one or more of the quality aspects. In addition a stage "hmi-aspects" representing objectives related to human-machine interfaces was added.

The further proposed step is to identify a list of questions to each objective. In the M-ADS project these questions were based on the understanding of the text in the main part of DO-178B, and formulated so that the answer could be given by a "*yes*" or a "*no*". However, as the questions often are of a qualitative nature, it may be difficult to give a straight answer. It could therefore be possible to answer the question with a number between 0 and 1 as an expression of the strength in the belief that the answer is *yes* (1) or *no* (0). A list of the questions identified related to the "quality of product" for (A2) is presented in table 2. Figure 5 presents the same example as a BBN. A list of all the questions constructed is given in the Appendix B.

Table 2: The questions related to the lifecycle stage A2: software development process

Objective	Question:
sw req. data	Are all system functional requirements, safety requirements and auxiliary requirements specified in the software specification?
design descrip-tion	Are all tasks specified in the requirements also included in the design?
	Does the design adequately describe the information flow between components?
	Does the design address sequencing, concurrency and time related information?
	Does the design adequately describe the data structures and their properties?
	Is it a clear separation in the design between safety critical and not safety critical parts of the system?
	Are measures for fault tolerance, like diversity or redundancy designed into the system?
	Are control and data flow monitored when safety requirements dictate, e.g. through watchdog timers, reasonableness checks, input data checks etc.?
	Are the responses to failure conditions consistent with safety related requirements?

6.6 **The elicitation of probability tables**

The elicitation of conditional probability tables (cpts) to the nodes and edges can be done as a brainstorming exercise by the expert group. In general, this means that for each node, the expert group has to assess two conditional probabilities of the type P(good measurement | good quality) and P(bad measurement | bad quality).

The first probability is relatively easy to assess. Based on general knowledge and experience in software development and evaluation, it can be done by ranking the importance of the different sub nodes, giving them probabilities from a predefined set such as {0.5, 0.7. 0.9, 0.95, 0.99}. The latter, however, can be very difficult. Often, where the experts state that there is a dependency between good quality and a specific good measurement, they cannot state the opposite effect.

This was one of the lessons learned from the M-ADS project. Furthermore, also the approach of ranking the nodes had restricted success. Even if some of the project members can be considered as experts within their fields, it is, highly recommendable to make use of some

expert judgement tools or expert judgement expertise. For the lower-level network about 130 conditional probability tables were assessed. The establishment of the BBNs and prior pdfs was rather time consuming, and would be even more so for a system. On the other hand, the generation of the BBNs was related to DO-178B and on safety assessment in general, and not to the actual system. This implies that the BBNs have a general nature, and can be reused in many applications. They can also be gradually improved based on experience. Note also that on the lower level, as illustrated in Figure 5, all nodes have only one parent. This makes the complexity of the BBNs manageable. In the case of nodes with more incoming edges, it would be a good solution to apply the approach suggested by Neil et al. (2000)

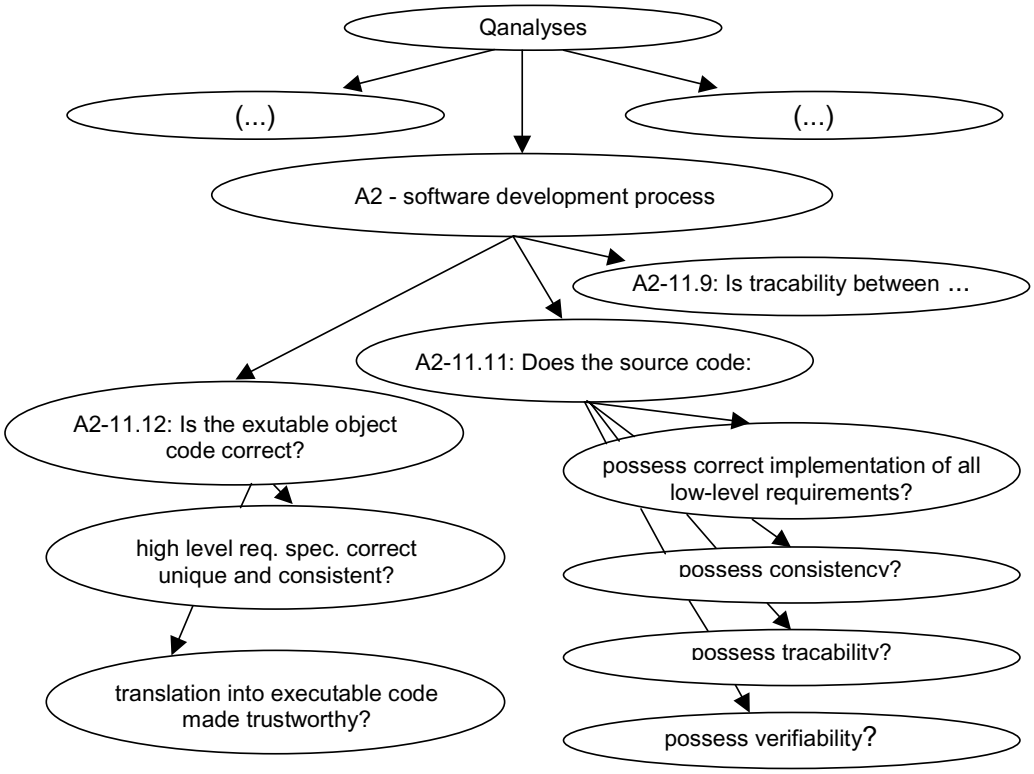

Figure 5: Example of a list of questions associated with two of the objectives for the software development process related to the quality of analyses

6.7 **Results from the M-ADS project**

All the BBNs were implemented, and all the conditional probability tables were fed into the HUGIN and SERENE tools. This made it possible to make a variety of computations (Gran et al. 2000), with the aim to investigate different aspects of the methodology, such as:

- What is the effect of observations during only one lifecycle process?
- How does the result change by subsequent inclusion of observations from the lifecycle processes?
- How sensitive is the result to changes in individual observations?

Since the number of possible scenarios is exploding when one wants to explore both different sets of observations and prior cpts, a limited number of computations were made. However, an interesting observation was that we rapidly found surprising results that

required further discussion and calculations. These results provided a list of topics for further research, both with respect to topological issues and with respect to different cases of observations.

The observations were done by KDA through several interview sessions with experts involved in the project. Totally, experts representing the software design and coding role, as well as project management role, were involved. In each session the questions associated with the end nodes in the network were used to assess the module in view of the scope defined by the node. The answers were, as discussed in section 6.5, given as weighted values on the scale from zero to one. In general the value zero (0) means objective achieved with poor quality, while the value one (1) means objective achieved at highest level of quality. There also were a few cases where a score, say 0.95, indicated objective achieved at highest level of quality for 95% of the modules. As an example refer to a question for the BBN for *Qanalysis*: "is the software quality assurance process properly performed and recorded?" The answer, 0.95, means that the expert board judged that software quality assurance process is properly performed and recorded for 95% of the modules.

6.7.1 *The partial scenarios results*

In addition to surprising results, this research demonstrated the importance of a good quality assurance of the observations entered into a BBN. The trigging event was the discovery of a wrong entered observation. Correcting this error demonstrated that one negative observations can have a significant effect on a partial results. By using the wrong observations it was concluded that there were effects with respect to the different quality aspects, and in particular with respect to the node "Qproducer" (Gran et al. 2000, Gran 2002a). After correcting this error, the effect of the observations during only one stage in the development and validation process showed that the effects, with respect to "QProducer", were approximately the same for all the processes (Gran 2001). The evaluation also showed that the one wrong (negative) observation, as well as a set of a few negative observations, is not enough to change the overall results.

6.7.2 *The incremental scenarios*

The observations could also be added subsequently, first during process A1, then A2 and so on (Gran et al. 2000, Gran 2002a). This illustrates how the posterior probability distributions change from the initial prior values towards a scenario given by all the KDA observations. For the *Qproducer* the expected value came up to a *top level* already after observations during processes A1 and A2 were made. This does not mean that the quality of the producer will remain on this level independent of other additional observations, but means that making additional "good" observations does not change our posterior results. With respect to the nodes *Qprocess*, *Qproduct*, and *Qanalysis* we had to make positive observations on all the processes A1 towards A8 before the posterior probability distributions achieved the top level. For the node *P*, the posterior distribution was at its top level after observations were made during process A1 up to A3. This is the similar effect as for the *Qproducer*. Note that, although there is no direct link between these two nodes, they behave in the same manner due to the propagation of positive measurements.

6.7.3 *Sensitivity cases*

A sensitivity analysis was performed for the node *P* given future observations on the node *N*

(new tests) (Gran et al. 2000, Gran 2002a). That is, with all the observations on the quality characteristics, represented in the node *N_hypothetical*, different measurements were made on the node *N*. Note that making a measurement equal to *m* assumes that a failure occurred after *m* failure free tests. The posterior probability distributions for *P* are shown in Figure 6. Compared to testing alone, these results show that observing *m* failure free tests, where *m* is higher than the hypothetical *N* failure free tests, will increase our belief in a shift left of the distribution for *P*. In the same way, observing *m* lower than *N* will shift it right, due to the situation that our prior belief is not in accordance with the real measurements.

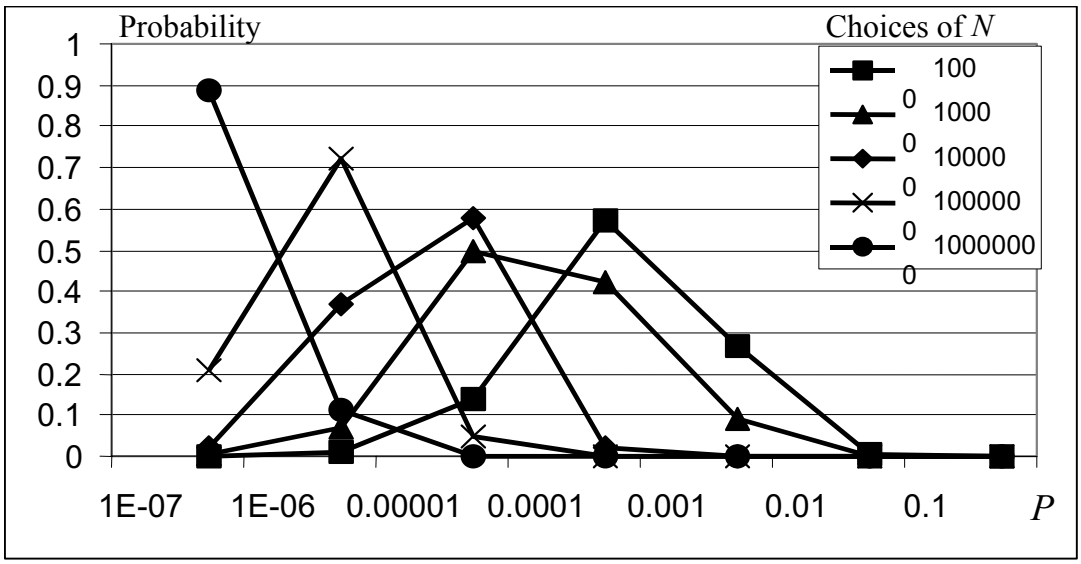

Figure 6. The posterior probability distribution for *P* for different number of new tests

6.7.4 *The effect of one negative and one "not positive" observation*

As stated for the partial scenario it was discovered that one observation, with respect to process A4, was given the value zero. This value corresponds to a negative answer to the question "Is the software partitioning integrity confirmed?" However, whether this answer was meant to be negative; i.e. that this question is of importance to the reliability of the product, or if this question was ranked as irrelevant, was not further discussed. In the latter case it would have been better not to give any value to this observable node at all. This is equivalent to cutting the edge to this node. A further walk-through of the observations also identified that 6 questions, which belong to two or more of the quality aspect networks, were given different observations in the different sub-networks (Gran 2001). Of these 6 questions, one belonged to the process A10, and was given a very low score for the "Qproduct". For the other divergences, the differences were smaller.

The result of correcting these faults was that the "surprising low effect for A4 and A10" disappeared. And, the processes with low effect were now observed to be A1 and A8. These were both also identified as contributors to low effect for the other quality aspects.

On the other hand, *if* one assumes that the questions should have been non-identical, and that the observations on these in fact were negative or low as entered, *then* we have identified a case where only two negative observations can lead to negative significant changes in the partial scenarios.

6.7.5 *The effect of some negative observations*

The latter result is related to the fact that the observations applied in the project were in general positive. An open question is therefore: what would be the result if more observations were negative? In particular, what are the overall results after entering observations in all phases? And, is it possible to find a set of "negative observations" that belong to all or more phases? The reason for the latter is that it is very little realistic to have good observation within 9 phases, and negative observations within the others. More realistic is that the negative observations are distributed over all phases.

An attempt to find such set of observations (out of a total of 71 observations) was to look into the set of observations (19 observations) that is related to two or three processes (Gran 2001). These 19 observations can be divided into 5 groups as shown in Table 3.

Table 3: The 5 groups of observations related to more quality aspects

Group	Related to quality aspect:	Processes:
1	"QProduct", "Qananlysis"	A4, A5, A7, other
2	"QProcess", "Qananlysis"	A1, A2, A6
3	"QProduct", "QProcess"	A5, A6
4	"QProduct", "QProcess", "QProducer"	A9, A10
5	"QProduct", "QProcess", "Qananlysis"	A3, A5

By entering negative observations to the questions related to the three groups we observed the effects as shown in Table 4 (Gran 2001). Remark that all the other observations are held positive, and the effect of change is observed related to "as observed by KDA", that is more or less all positive. As shown in the table, we see that there was only a significant effect on the "Qproducer". That means that we by entering negative observations on the two questions related to processes A9 and A10, we achieve a lower confidence in good quality of the producer.

Table 4: The effect of negative observations related to the questions from the groups 1-5.

Gr.	Observed Effect
1	Minor effect to "QProduct" and "Qananlysis"
2	Minor effect on "QProcess", no effect on "Qananlysis"
3	Minor effect on "QProcess", no effect on "QProduct"
4	No effect on "QProduct", minor effect on "QProcess", but significant effect on "QProducer"
5	No effect on "QProduct", minor effect on "QProcess", and no effect on "Qananlysis"

6.7.6 *The effect of 19 negative observations*

Based on the results presented above, the next scenario was to enter a negative observation on all the questions related to all the groups presented in Table 3. This had a significant effect on all the quality aspects, and also the node "P(failure state)"as shown in Table 5 and Table 6. An issue for further investigation is to look the combinations of these 19 to see how the results turn from positive towards negative.

Table 5: The effect of 19 negative observations on the node QProducer.

Scenario	good=5	4	3	2	bad=1
KDA original[1]	0.145	0.782	0.070	8E-6	6E-8
KDA corrected[2]	0.184	0.804	0.011	1E-6	1E-8
19 negative	0.018	0.359	0.621	0.01	1E-6

(1) with wrong observation, (2) after corrections

Table 6: The effect of 19 negative obs. on the nodes Qprocess, Qproduct and Qanalysis.

Scenario	Process		Product		Analysis	
	good	bad	good	bad	good	bad
KDA original[1]	1.0	4E-7	1.0	3E-9	1.0	1E-9
KDA corrected[2]	1.0	2E-7	1.0	1E-10	1.0	1E-9
19 negative	0.039	0.961	0.117	0.883	0.993	0.007

(1) with wrong observation, (2) after corrections

6.7.7 *Discussion of the results*

One other observation from the results from the incremental scenario is that they reached stable maxims very fast. This indicates that the activities in the later stages in the development and validation process have little effect. Similarly, the partial results are almost as good as complete results. These results are not expected.

One possible explanation is that a good score in one table is an indication of high quality during all phases, so that there also should be high scores in other tables. Another explanation is that 19 questions are repeated in two or three tables. However, these two explanations are not necessarily different. The latter can be a way to use the BBN topology to express the first explanation; i.e. that certain types of observations are relevant for several of the development phases associated with the tables.

6.7.8 *The difference in partial results for A4 and A5*

A third observation from the project was obtained by comparing the partial results from the lifecycle processes "verification of outputs of software design processes" (A4) and "verification of outputs of software coding and integration process" (A5). The "good" score for these are the same for the quality aspects "producer" and "analysis", but A5 scores better on "process" and in particular on "product". To explain this difference, the differences in BBN topology, in the cpts, and in the observations are investigated (Gran 2001).

The investigation showed related to the partial results for the processes A4 and A5, that we have the effects of both neutralizing, conformity, enlargement and the effect of the observations alone, see details in Table 7.

The comparison of the results for A4 and A5 can also explain the difficulty of finding a subset of observations that turns the results negative. Accordingly we shall expect problems with finding a subset of positive observations leading to stable maxims.

These results also indicate the problems of performing a verification of a Bayesian Belief Network. The reason is that two different groups of experts can come up with two different

BBNs. If one then enters somehow different observations into these networks, there is a good chance of observing the same results for the target nodes. On the other hand, these results also point in the direction that two different BBNs should be based up on the same observations. This again is an argument in favour of the BBN-construction process applied: each objective in the guideline associated to a list of questions.

Table 7: The partial results for the processes A4 and A5.

Quality aspect	Observed difference in A4 and A5	Difference in topology	Difference in observations	Effect of observations and topology (A4 vs. A5)
Producer	"A4 = A5"	differences in the cpts	different observations	neutralize each other
Analyses	"A4 = A5"	different number of questions	no large differences	conformity, i.e. "A4 = A5"
Process	"A4 ≠ A5"	differences in the cpts and topology	different observations	topology and observations pull in same direction (enlargement)
Product	"A4 ≠≠ A5	different number of questions	different observations	although a different number of questions, the observations alone give the difference

6.8 **Discussion of the M-ADS results**

The research conducted addresses some of the observations pinpointed as interesting, strange or counter-intuitive in the project on combining the Bayesian Belief Nets technology with the rules of a standard for safety critical software, DO-178B for a real, safety related, programmable system. One results is the importance of a good quality assurance of the observations entered into a BBN. One the other hand, it also demonstrates that one negative observations can have a significant effect on a partial results. The evaluation has also showed that one negative observation, or a set of a few negative observations is not enough to change the overall results.

The results also show that there can be a rapid change in the overall results, given a specific order of turning the observations. This work indicates that this change takes place somewhere in the "middle" of "negative observations on a few repeated questions" and "negative observations on all repeated questions". A further evaluation can give more specific results. However, the evaluation has also showed that there is an effect of the combination of topology, cpts and observations. A pinpointed set of observations could therefore change by a change in the topology or a conditional probability table.

Finally the evaluation points on some of the problems that one will be faced with wanting to perform a validation or verification of the BBN. One hypothesis is that the use of questionnaires can be a vital point.

7 Extending the BBNs based upon RTCA/DO-178B

Within the nuclear field there is an increased focus on risk-based regulation of nuclear power plants. This is in accordance with the new generic guideline for programmable safety related systems, IEC-61508 (2000), where probabilistic safety integrity levels are given as requirements for safe operation. Therefore, there is a need to establish methods to assess the

reliability of programmable systems, including the software. One approach in this research is an on-going long-term joint research activity between the Halden Project (HRP) and VTT Automation (VTT) in Finland (Gran and Helminen 2001, Gran 2002b). One objective of this co-operative project is to investigate how a network, representing the software safety guideline and different quality aspects, as described in the previous chapter, can be merged with a network, developed by VTT, representing evidence from disparate operational environments (Helminen 2000).

7.1 **The VTT approach**

The main sources of reliability evidence in the case of safety critical systems considered in the VTT approach are depicted in Figure 7 (Neil et. al 1996a). A similar version of this model has been presented by Stålhane et al. (1993). Part of the evidence, such as the evidence obtained through operational experience and testing, may be directly measurable statistical evidence. Part of the evidence, such as the design features and the development process of the system, may be qualitative characterization of the system.

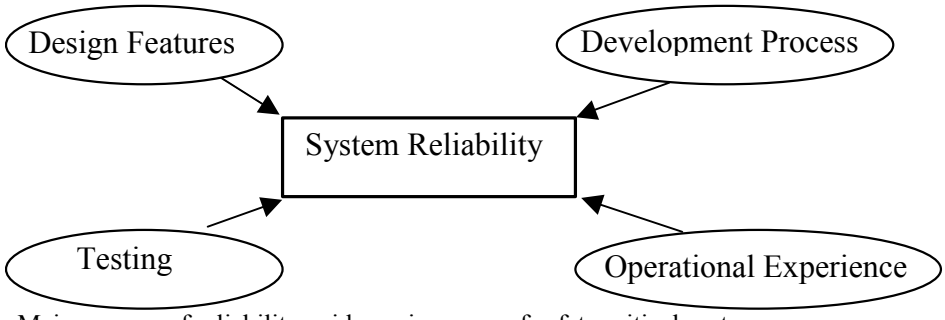

Figure 7: Main sources of reliability evidence in a case of safety critical system

The qualitative characterization of the design features and the development process follows certain quality assurance and quality control principles, which are based on applicable standards. Running a good development process alone does not guarantee a more reliable product. However, the more strict standards the characterizations fulfil, combined with good testing results, the more confidence one will become in having a reliable system. The evidence based on qualitative characterization can be considered as soft evidence, while evidence obtained from operational experience and testing can be considered as hard evidence. The exploitation of soft evidence in the reliability analysis of software-based system requires extensive use of expert judgment making it quite an unforeseeable matter and therefore the VTT approach is mainly focused to the utilization of hard evidence (Helminen 2000).

The reliability of a software-based system is modelled as a failure probability parameter, which reflects the probability that the automation system does not operate when demanded. Information for the estimation of the failure probability parameter can be obtained from the disparate sources of hard and soft evidence. To obtain the best possible estimate for the failure probability parameter of the target system all evidence should to be combined.

The principle idea of the estimation method is to build a priori estimate for the failure probability parameter of software-based system using the soft and hard evidence obtained from the system development process, pre-testing and evaluating system design features while system is produced, but before it is deployed. The prior estimation is then updated to a

posterior estimate using the hard evidence obtained from testing after the system is deployed and from operational experience while the system is operational. The difference between disparate evidence sources can be taken care in the structural modelling of the Bayesian Network model.

To analyse the applicability of Bayesian Networks to the reliability estimation of software-based systems Bayesian Network models for safety critical systems are built. The different models are distinguished by the evidence, which is collected from different systems and from different operational profiles. The Bayesian Network shown in the "left-low" part of Figure 8 describes a system, for which the observed number of failures *Y* is binomial distributed with parameters *N* and *P*. The parameter *N* describes the number of demands in the single test cycle and parameter *P* is the random failure probability parameter. To increase the flexibility of the model depicted in the left part, a logit-transformed *P* parameter Θ is included into the network, and the network becomes as shown to left in Figure 8.

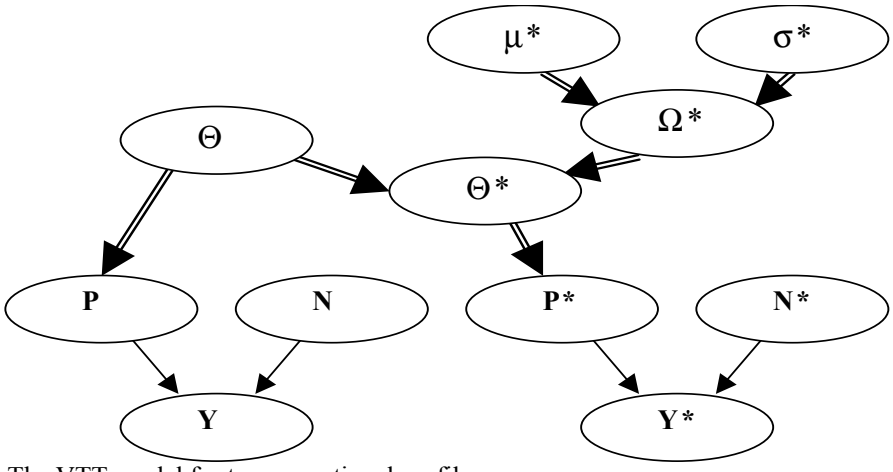

Figure 8: The VTT-model for two operational profiles

Often the system is tested with different operational profiles under different operational environments. The results from applying the different operational profiles provide different failure probabilities for the same system. However, the failure probability from testing gives us some information about the failure probability of the system functioning in a different operational profile than where the testing was made. This evidence provided by testing is valuable and one should make a good use of it by taking into account the difference in the operational profiles when building the model. Helminen (2000) solve the problem of different operational profiles by connecting the binomial distributed evidence from different operational profiles to separate failure probability parameters. The Bayesian Network representing the case is illustrated as the whole of Figure 8.

7.2 **Merging the HRP approach and the VTT approach**

The merging of the two approaches is based on the on the network presented in Figure 4 (the higher level) and the left part of the network shown in Figure 8. The merged network is displayed in Figure 9 (Gran and Helminen 2001). The merging is done by replacing the node "P(failure state)" by the node "Θ_{priori}".

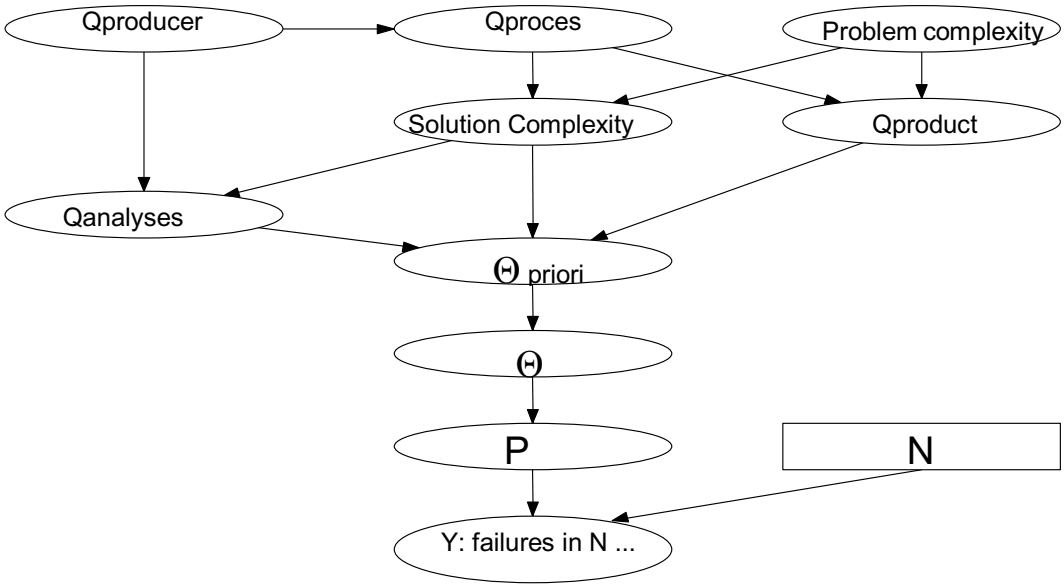

Figure 9: The merged network from the VTT and HRP approaches

7.3 **Results from applying the merged BBN**

Each of the quality aspect nodes was connected to quality aspects, as described in section 6. That allowed us to directly insert the observations from the M-ADS evaluation in the network. In addition the conditional probability tables for P(failure state) ware transformed into continuous normal distributions. For the merged network we performed calculations for the case of "no M-ADS observations" and for the case "with the M-ADS observations", running from N=100 to N=1000000.

For both scenarios the target was the node for the failure probability. In Figure 10 both the median and the 97.5% percentile posterior distribution values for P on the logarithmic scale are shown. The values for N=1, are the values representing the prior distributions, i.e. before starting the testing (and observing Y=0). Remark that the curves for the 97.5% percentiles are somewhat "bumpy". This due to the fact, that the values are deduced from posterior histograms.

7.4 **Experiences from merging the BBNs**

The main differences between the two studies lie in the difference of focus areas. The work by VTT mainly focuses to studying explicitly the influence of prior distributions to the reliability estimation and to the investigation of combining statistical evidence from disparate operational environments. The work described in this thesis focuses mainly on how to model a software safety guideline, DO-178B, and how to combine "soft evidences" in the safety assessment of a programmable system. The key idea is to split the larger entities of soft evidence into smaller quantities. Another difference is the comprehensive usage of continuous distributions in the VTT work, which is somewhat a different approach than the approach used in the BBNs for the DO-178B. This is however not discussed in this work.

The merged networks show how the two approaches can be merged. It gives an extended description of the quality aspects, originally modelled by the node Θ in the VTT approach, and it shows how different operational profiles, can be included in the approach described.

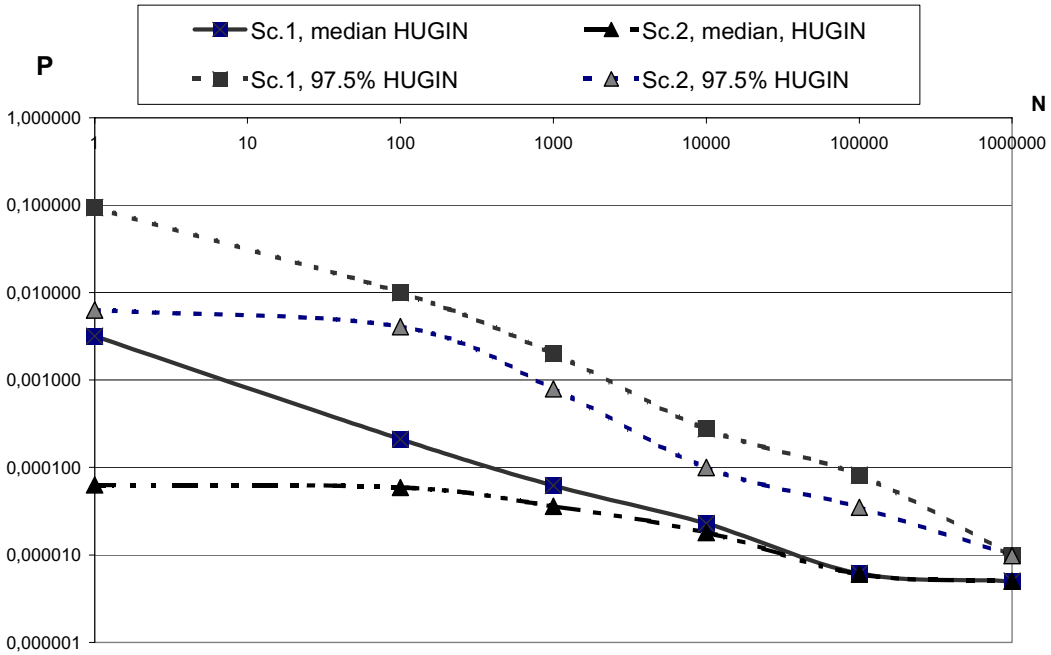

Figure 10: Median and 97.5% percentile posterior distribution values for P on the logarithmic scale, for the scenario of no observations and the scenario with the observations

8 Conclusions

The objective of the research has been to investigate the possibility to transfer the requirements of a software safety standard into Bayesian belief networks (BBNs). The BBN methodology has mainly been developed and applied in the AI society, but more recently it has been proposed to apply it to the assessment of programmable systems. The relation to AI application is relevant in the sense that the method reflects the way of an assessor's thinking during the assessment process. Conceptually, software reliability is almost impossible to compute, since many of the aspects of the software which influence the reliability are of qualitative nature and not directly measurable, but have to be estimated e.g. by expert judgement.

The conclusion from the research presented in this thesis is that the use of Bayesian Belief Networks for combining disparate sources of information in the safety assessment of software based systems, combined with questionnaires, offers a systematic way to combine quantitative and qualitative evidences of relevance for the safety assessment of programmable systems, e.g. in a licensing process or in a PSA analysis.

The BBN is constructed in two levels. The higher level is based on the four qualities: quality of the producer, quality of the production, quality of the product, and quality of the analysis. The higher-level BBN is general, and independent of the standard, and is based on the research discussed in chapter 2. The lower-level BBNs reflect the recommendations of RTCA/DO-178B. Each top node of the lower-level BBNs is linked to intermediate nodes representing the 10 lifecycle stages identified in DO-178B. Each of these nodes are again linked to other intermediate nodes, representing the objectives of each lifecycle. The further proposed step is to identify a list of questions to each objective. In the described research

these questions are based on the understanding of the text in the main part of DO-178B, and formulated so that the answer could be given by a “yes” or a “no”.

For both the higher and lower level networks there is a need for further validation. This is demonstrated through the experimental investigation with the BBNs. However, a hypothesis is that a reallocation of objectives or questions only will give local (or partial) effects, and not changes in the overall assessment. A reason for this could be that there are a few “soft evidences” and dependencies connecting these evidences that are more sensitive than the other. So fare, there has, however, not been possible to find such evidences.

Although the BBNs and results are based upon a real application, this approach has not been applied to a real development or assessment. A first try could be to apply the approach for decision support in the approval of safety critical programmable systems. Another try could be to apply the approach as decision support early in the development of a system, in order to point on where to set in the effort and thus being able to reach specific objectives of the final product.

The establishment of the BBNs and prior probability distributions can be rather time consuming. However, the process of building up the network, e.g. by making questionnaires, and doing the elicitation of the prior distributions related to a standard (RTCA/DO-178B), and not to the actual system, implies that the network and questions are of a general nature, and can be reused in many applications. They can also be gradually improved based on experience. The experiences with modelling the requirements of the avionics standard RTCA/DO-178B as BBNs, point in the direction that this approach can be transferred to the modelling of other software standards built on the same basic framework, and which follow the same principles. This holds even though they may differ in the aspect they put special emphasis on.

Conceptually, estimation of the dependability of programmable systems is nearly impossible to compute, since many of the characteristics to be considered are of qualitative nature and not directly measurable, but have to be estimated. The most difficult activity in the experiment described was to perform the expert judgment, in particular in the assignment of values to the conditional probability distributions. Even if some of the project members can be considered as experts within their fields, it is highly recommendable to make use of some expert judgment tools or expert judgment expertise. Note also that knowledge within BBN and probabilistic theory is of great advantage in the construction of the networks and the assessment of the probability distributions, and also an advantage in the evaluation of the results from the computations.

References

Aldenryd, S.H., Jensen, K.B., Nielsen, L.B., (1993). *Hugin Runtime for MS-Window,* Tool made by Hugin Expert a/s, Aalborg, <http://www.hugin.dk>

Barnes, M., Bishop, P.G., Bjarland, B., Dahll, G., Esp, D., Humphreys, P., Lahti, J., Yoshimura, S., Ball, A., Hatlewold, O., (1985). PODS (The Project on Diverse Software)*, OECD Halden Reactor Rep*. HPR-323.

Bertolino, A., Strigini, L., (1996a). Predicting Software Reliability from Testing Taking into Account Other Knowledge about a Program. *Proceedings 9th International Software Quality Week* (Software Research Institute, San Francisco).

Bertolino, A., Strigini L., (1996b). Acceptance Criteria for Critical Software Based on Testability Estimates and Test Results. *Proceedings SAFECOMP96, 15th International Conference on Computer Safety, Reliability and Security*, Schoitsch (Ed..), Springer-Verlag, pp 83-94.

Bertolino, A., Strigini, L., (1998). Assessing the risk due to software design faults: estimates of failure rate vs. evidence of perfection, *Software Testing, Verification and Reliability*, 8(3), pp 155-166.

Bishop, P., Esp, D., Barnes, M., Humphreys, P., Dahll, G., Lahti, J., (1986). PODS - The Project on Diverse Software. *IEEE Transactions on Software Engineering*, SE-12, no. 9.

Chrisman, L. (1996), *A roadmap to research on Bayesian networks and other decomposable probabilistic models*, School of Computer Science, Pittsburg, PA.

Cooke, R. M. (1991), *Experts in Uncertainty*, Oxford University Press, New York.

Cowell, R.G., Dawid, A.P., Lauritzen, S.L., Spiegelhalter, D.J., (1999). *Probabilistic Networks and Expert Systems,* Springer-Verlag.

Cudleigh, M., Catmur, J., (1992). Safety Assessment of Computer Systems using HazOp and Audit Techniques, *Safety of Computer Systems SAFECOMP'92*, Frey (Ed.), Pergamon Press.

Dahll, G., (1997). Safety Assessment of Software Based Systems. *SAFECOMP'97*, Daniel (Ed.), Springer-Verlag, pp. 14-24.

Dahll, G., (2001). Combining Disparate Sources of Information in the Safety Assessment of Software Based Systems, submitted to *Special Issue of Nuclear Engineering and Design*.

Dahll, G., Gran, B.A., (2000). The Use of Bayesian Belief Nets in Safety Assessment of Software Based Systems. *Special Issues of International Journal on Intelligent Information Systems at FLINS'98, Int. J. General Systems,* 24 (2), 205-229.

Delic, K., Mazzanti, M., Strigini, L. (1995). *Formalising a Software Safety Case via Belief Networks*, Technical report, CRS, City University, London.

Delic, K., Mazzanti, M., Strigini, L. (1997). Formalising Engineering Judgement on Software Dependability via Belief Networks. In: *DCCA-6, Sixth IFIP International Working Conference on Dependable Computing for Critical Applications, "Can We Rely on computers?",* Garmisch-Partenkirchen, Germany.

DeMillo, R.A., Offutt, A.J., (1991). Constraint-Based Automatic Test Data Generation, *IEEE Trans. Software Eng*., 17 (9), pp 900-910.

Fenton, N., Neil, M., (1999). A Critique of Software Defect Prediction Models, *IEEE Transactions on Software Engineering*, 25 (5), 675-689.

Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B., (1995). *Bayesian Data Analysis*, Chapman & Hall, London.

Gran, B.A., (2001). Applying Bayesian Belief Net in Software Safety Assessment on a Real, Safety Related Programmable System. In *Safety & Reliability, Towards a safer world*. Zio, E., Demichela, M., and Piccinini, N. (Eds), Politecnico di Torino, Torino, pp. 1045-1052.

Gran, B.A., (2002a). Assessment of programmable systems using Bayesian belief nets. Paper accepted for *Journal Safety Science, Special Issue on Safecomp-2000*.

Gran, B.A., (2002b). The use of Bayesian belief networks for combining disparate sources of information in the safety assessment of software based systems. Paper accepted for International *Journal of Systems Science, Special Issue on Intelligent Product Support Systems*.

Gran, B.A., Helminen, A., (2001). A Bayesian Belief Network for Reliability Assessment, *Computer Safety, Reliability and Security (Lecture Notes in Computer Science 2187)*, Voges (Ed.), Springer, pp. 35-45.

Gran, B.A., Thunem, H., (1998) EISTRAM - Experimental Investigation of the PIE-technique. *Safety and Reliability*, Lydersen, Hansen and Sandtorv (eds), Balkema, Rotterdam, pp 409-416.

Gran, B.A., Dahll, G., Eisinger, S., Lund, E., Norstrøm, J., Strocka, P., Ystanes, B., (2000). Estimating Dependability of Programmable Systems Using BBNs. *Computer Safety, Reliability and Security, Proceedings from Safecomp 2000,* (*Lecture Notes in Computer Science 1943*), Koornneef and van der Meulen (Eds) (Springer), pp. 309-320.

Helminen, A., (2000). *Reliability Estimation of Software-based Digital Systems Using Bayesian Networks*, (Helsinki University of Technology, Espoo), pp. 1-50.

IAEA, ID NS 264, (1999). Software for Computer systems Important to Safety in NPPs: A Draft Safety Guide.

IEC 61508, (2000). Functional safety of electrical/electronic/programmable electronic safety-related systems.

IEC 880, (1986). Software for computers in the application of industrial safety related systems..

IMPRESS, (1999). Improving the software process using Bayesian nets. EPSRC project nr. GR/L06683, <http://www.csr.city.ac.uk/csr_city/projects/impress.html>.

Jensen, F., (1996). *An Introduction to Bayesian Networks*, UCL Press, University College London.

Korhonen, J., (1997). *Combining the Evidence in Software Reliability Assessment*, Technical report, VTT, Finland.

Lauritzen, S.L., Spiegelhalter, D.J., (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussions), *Journal of the Royal Statistical Society*, Series B 50 (2), pp. 157-224.

Leveson, N.G., (1995), *Safeware – System Safety and Computers*, Addison-Wesley.

Littlewood, B., Wright, D., (1995). A Bayesian Model that Combines Disparate Evidence for the Quantitative Assessment of System Dependability. *Proceedings SAFECOMP'95*, Rabe (ed), Springer-Verlag, pp. 173-188.

Littlewood, B., Wright, D., (1997). Some conservative stopping rules for the operational testing of safety-critical software, *IEEE Transactions of Software Engineering*, 23(11), pp 673-683.

Neil, M., Fenton, N., (1996b). Predicting Software Quality using Bayesian Belief Networks, *Proceedings of 21st. Annual Software Engineering Workshop*, (NASA Goddard Space Flight Centre), pp. 217-230.

Neil, M., Fenton, N., (1998). A strategy for improving safety related software engineering standards, *IEEE Trans. on SW Eng*., 24 (11).

Neil, M., Fenton, N., Nielsen, L., (2000). Building large-scale Bayesian Networks, *The Knowledge Engineering Review*, 15 (3), pp. 257-284.

Neil, M., Fenton, N., Forey, S., Harris, R., (2001). Using Bayesian Belief Networks to Predict the Reliability of Military Vehicles, *IEEE Computing and Control Engineering*, 12 (1), pp. 11-20.

Neil, M., Littlewod, B., Fenton, N., (1996a). Applying Bayesian Belief Nets to Systems Dependability Assessment, *Proceedings of 4th Safety Critical Systems Symposium*, (Springer-Verlag), pp. 71-93.

Pearl, J., (1988). *Probabilistic Reasoning in Intelligent Systems: Networks forPlausible Inference*, Morgan Kaufman.

Pulkkinen, U., (1994). Statistical Models for Expert Judgement and Wear Prediction, *Dissertation, Helsinki University of Technology*, Finland.

Pulkkinen, U., Holmberg, J., (1997). A Method for Using Expert Judgement in PSA, (Finnish Centre for Radiation and Nuclear Safety, Helsinki), pp. 1-32.

Ramamoorthy, C.V., Bastani, F.B., (1982). Software Reliability - Status and Perspectives, *IEEE Trans. Software Eng*., SE-8 (4), pp 354-371.

Rasmussen, (1974). Reactor Safety Study, *U.S. Atomic Energy Commission Report* WASH-1400.

RTCA/DO-178B, (1999), Software Considerations in Airborne Systems and Equipment Certifications (Guideline).

SERENE, (1999). Safety and Risk Evaluation using Bayesian Nets. ESPRIT IV nr. 22187, <http://www.hugin.dk/serene/>.

Spiegelhalter, D. J., Dawid, A. P., Lauritzen, S. L., Cowell, R. G., (1993). Bayesian Analysis in Expert Systems, *Statistical Science*, 8(3), pp 219-283

Stålhane, T., (1997). Safety in Software-Intensive Systems. Presented at the Eighth European Workshop on Dependable Computing, EWDC-8, Gothenburg, Sweden.

Stålhane, T., Meulen, M.J.P van der, Cole, (1993). Reliability Assessment of PES, using Subjective and Objective Categorcial Data. Presented at workshop on production control in the process industry, 29-31 March, Dusseldorf, Germany.

Voas, J.M., Michael, C.C., Miller, K.W., (1993). Confidently Assessing a Zero Probability of Software Failure, *Proceedings of the 12th Int'l. Conference. on Computer Safety, Reliability, and Security*, Poznan and Poland (Eds), Springer-Verlag, pp 197-206.

Voas, J.M., (1992). PIE: A dynamic Failure-Based Technique, *IEEE Trans. Software Eng*., 18 (8), pp 717-727.

Welsh, A. H., (1996). *Aspects of Statistical Inference*, Wiley & Sons.

Whittaker, J., (1990). *Graphical Models in Applied Multivariate Statistics*, Wiley & Sons.

Xie, M., (1991). *Software Reliability Modelling*, World Scientific Publishing Co. Pte. Ltd.

APPENDIX A: The lower-level BBNs

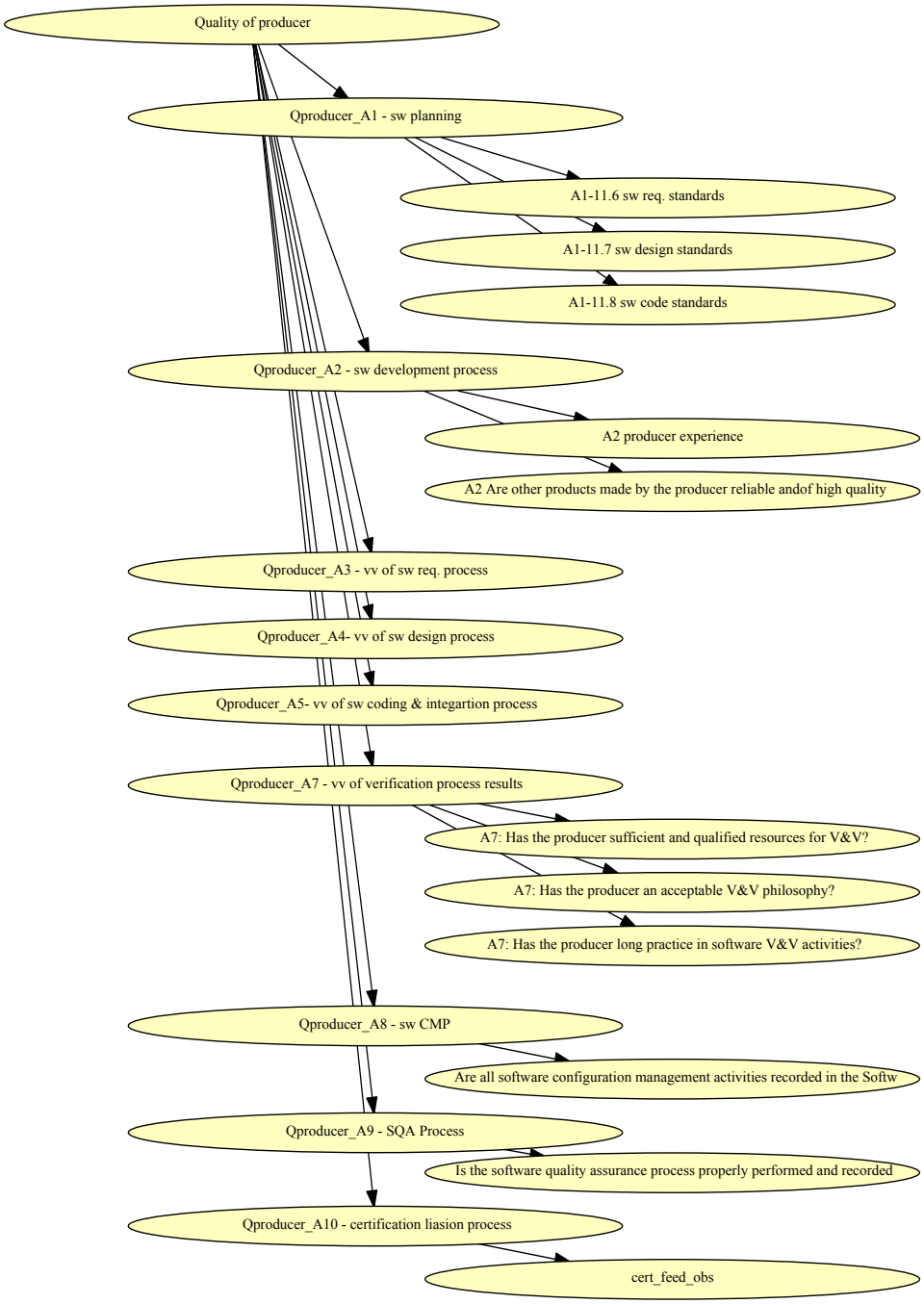

Figure A-1: The BBN for “Quality of producer”

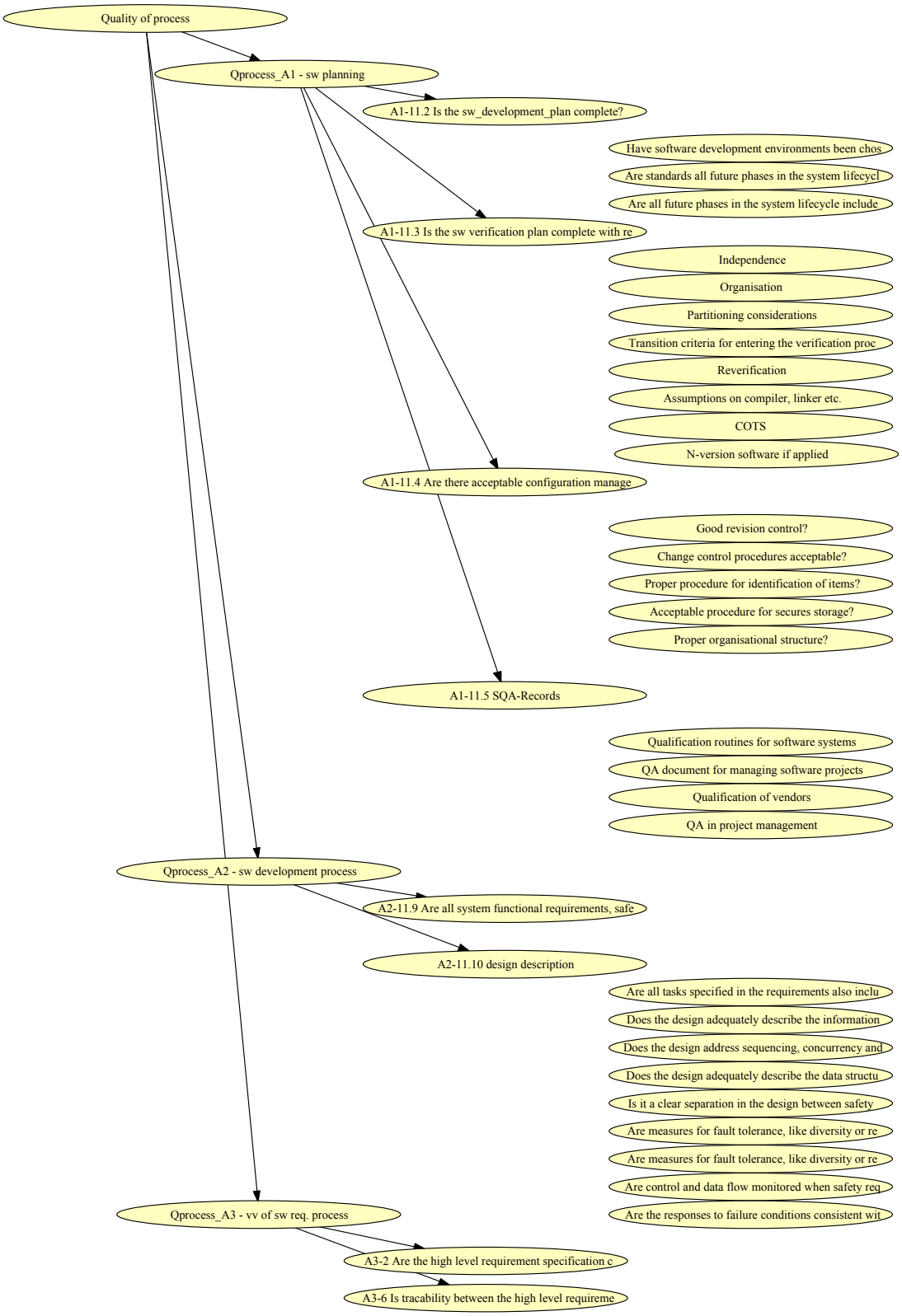

Figure A-2: The BBN for "Quality of production process" (cont. next page)

Figure cont.: The BBN for "Quality of production process" (continued)

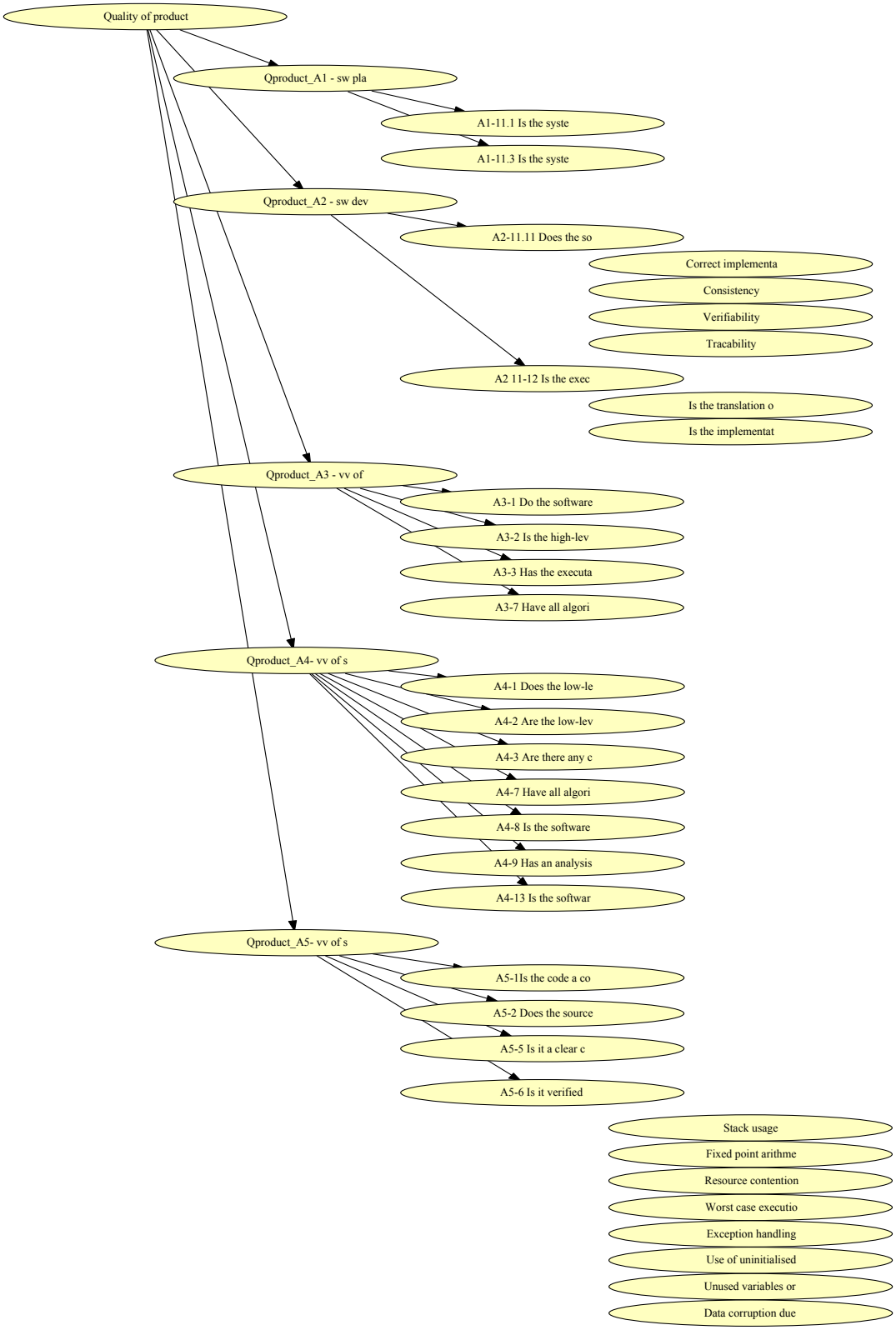

Figure A-3: The BBN for "Quality of product" (cont. next page)

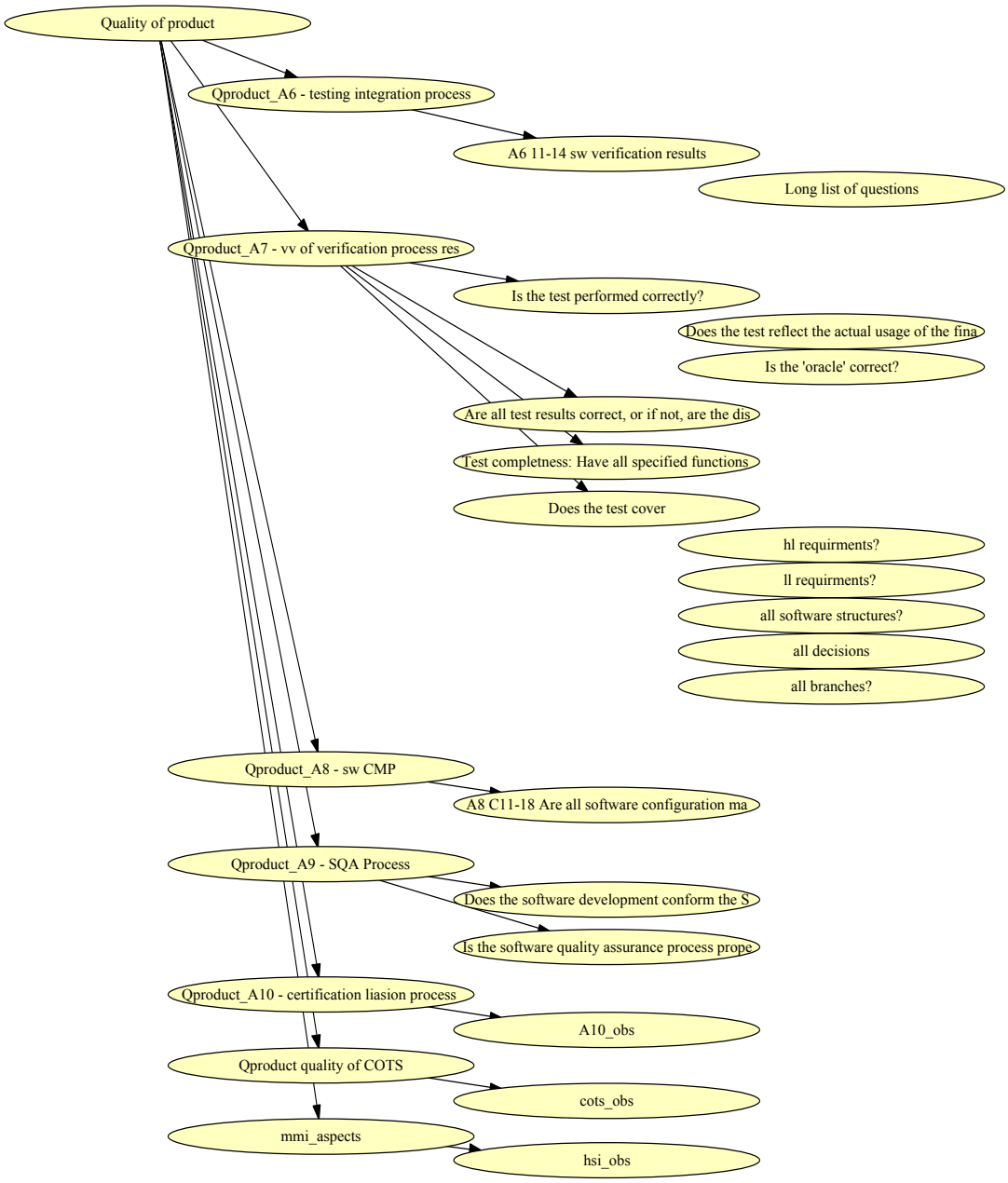

Figure cont.: The BBN for "Quality of product" (continued)

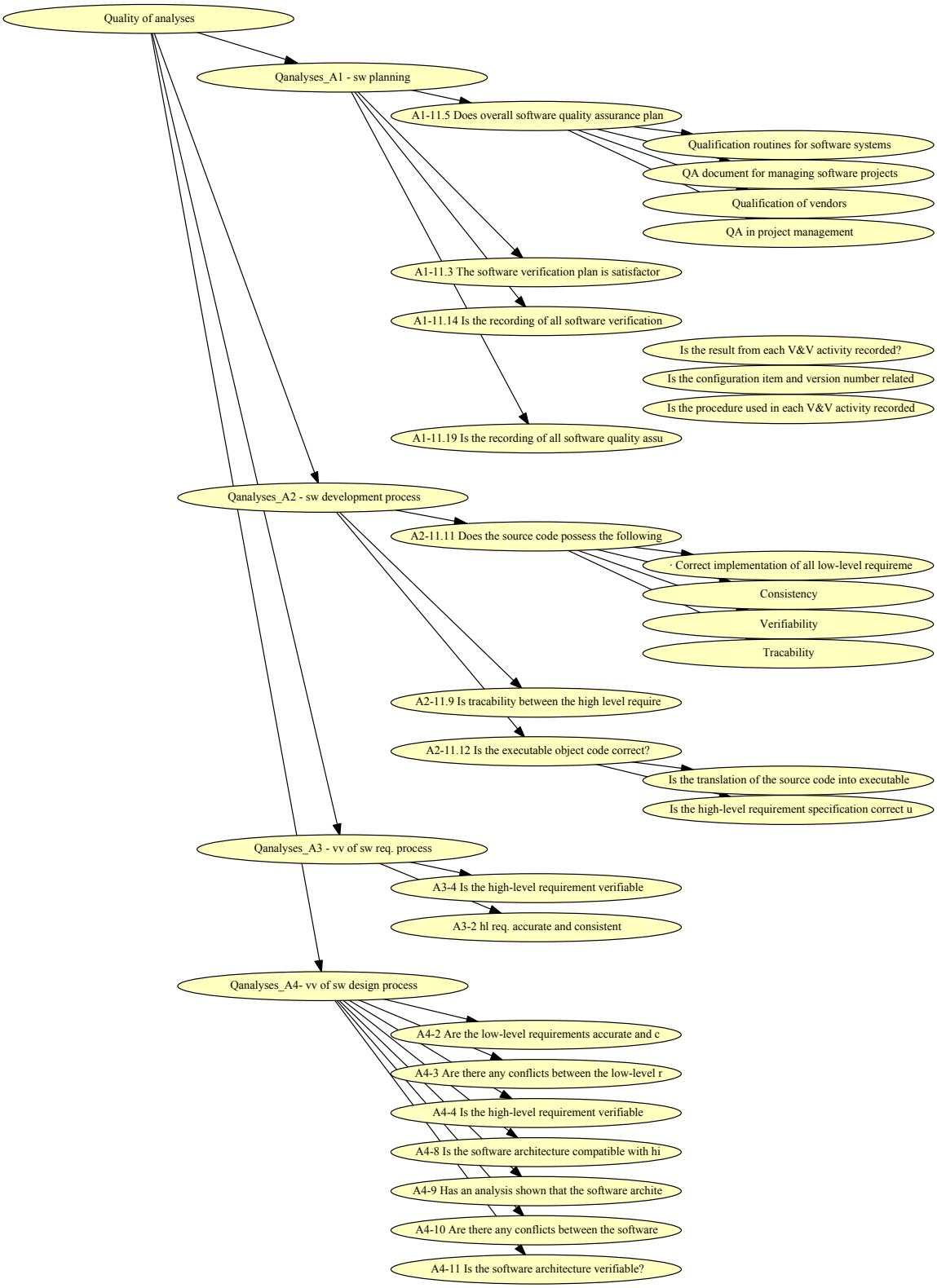

Figure A-4: The BBN for "Quality of analysis" (cont. next page)

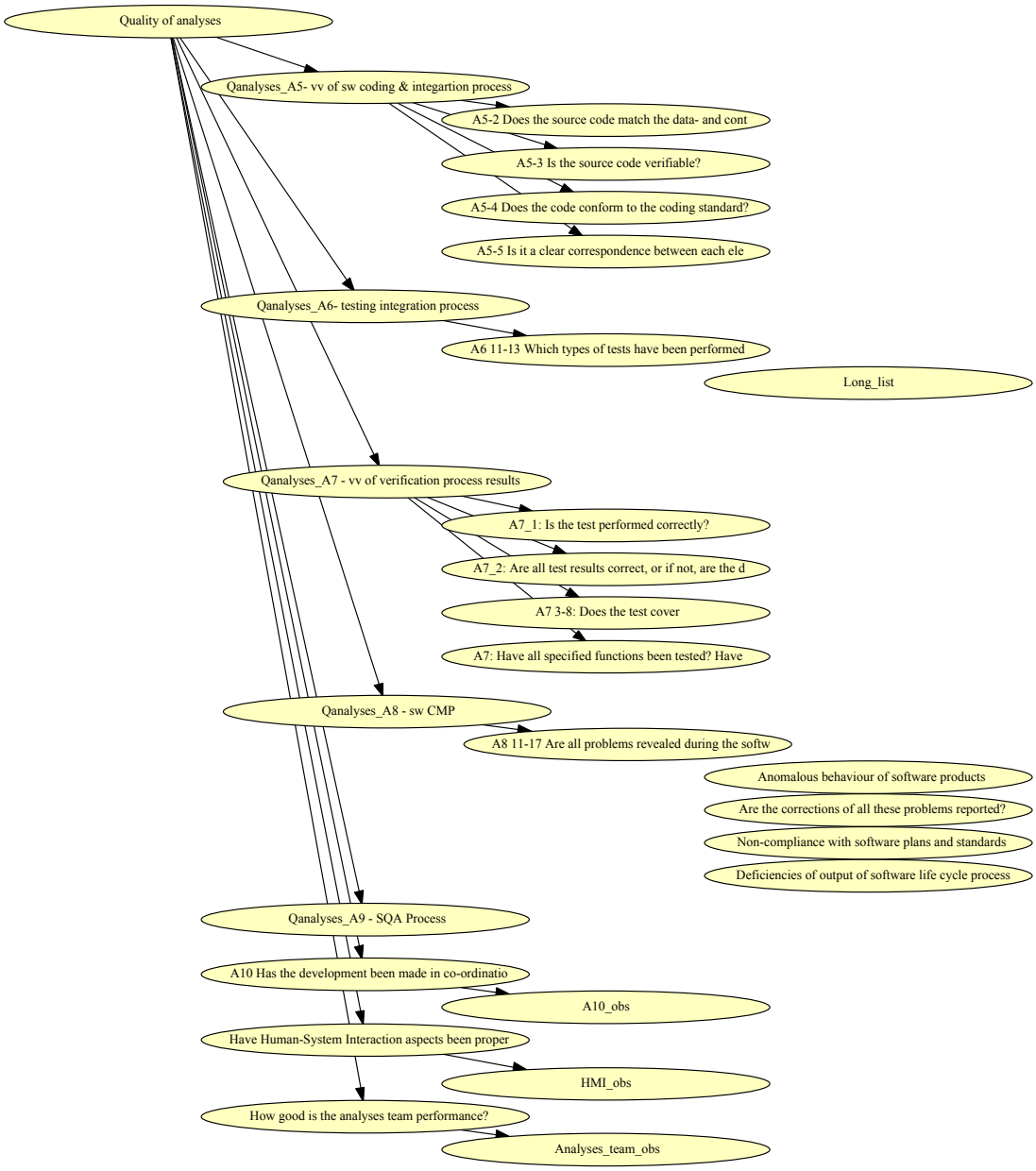

Figure cont.: The BBN for "Quality of analysis" (continued)

APPENDIX B: The questions related to DO-178B

Questionnaires for "Quality of Producer":

Node/table reference	Question
Quality_Producer	
Qproducer_sw_plan_process/ A1	
sw_req_standards/ A1-11.6	Does the producer follow software requirement standards?
sw_design_standards/ A1-11.7	Does the producer follow software design standards?
sw_code_standards/ A1-11.8	Does the producer follow software coding standards?
Qproducer_sw_dev_process/ A2	
producer_experience/ A2	Has the producer long experience in making similar systems?
Producer pedigree	Are other products made by the producer reliable and of high quality?
Qproducer_vv_sw_req_process/ A3	
hl_vs_standards/ A3-5	Do the high-level requirements conform to the producer's standard?
Qproducer_vv_sw_design_process/ A4	
sw_arch_standards/ A4-12	Do the software architecture conform to the producer's design standard?
Qproducer_vv_sw_coding_process/ A5	
code_traceable_ll/ A5-5	Is it a clear correspondence between each element of the design and corresponding code modules?
Qproducer_vv_vv_process/ A7	
vv_resources/ A7	Has the producer sufficient and qualified resources for V&V?
vv_philosophy/ A7	Has the producer an acceptable V&V philosophy?
vv_in_practice/ A7	Has the producer long practice in software V&V activities?
Qproducer_sw_CMP/ A8	
CMP_exists_used/ A8	Are all software configuration management activities recorded in the Software Configuration Records?
Qproducer_SQAP/ A9	
sqa_in_practice/ A9	Is the software quality assurance process properly performed and recorded?
Qproducer_cl_process/ A10	
certification_feedback/ A10	Have Human-System Interaction aspects been properly considered during the development of the system?

Questionnaires for "Quality of Production process":

Node/table reference	Question
Quality_Process	
Qprocess_sw_plan_process	
sw_development_plan /A1-11.2	Is the sw_development_plan complete?: o Are all future phases in the system lifecycle included in the plans? o Are standards for all phases in the system lifecycle included in the plans? o Have software development environments been chosen?
Sw_verification_plan/A1-11.3	Is the sw verification plan complete with respect to: o Organisation o Independence o Verification methods o Transition criteria for entering the verification process into the plan o Partitioning considerations o Assumptions on compiler, linker, etc. o Reverification o COTS o N-version software if applied
SCM_plan/A1-11.4	Are there acceptable configuration management plans for all phases? o Proper organisational structure? o Proper procedure for identification of items? o Acceptable procedure for secures storage? o Change control procedures acceptable? o Good revision control?

SQA_records/ A1-11.5	Does overall software quality assurance plans exist for all phases? Are there: o Qualification routines for software systems o QA document for managing software projects o Qualification of vendors o QA in project management
Qprocess_sw_dev_process	
sw_req_data/A2-11.9	Are all system functional requirements, safety requirements and auxiliary requirements specified in the software specification?
design_description/A2-11.10	o Are all tasks specified in the requirements also included in the design? o Does the design adequately describe the information flow between components? o Does the design address sequencing, concurrency and time related information? o Does the design adequately describe the data structures and their properties? o Is there a clear separation in the design between safety critical and not safety critical parts of the system? o Are measures for fault tolerance, like diversity or redundancy designed into the system? o Are control and data flow monitored when safety requirements dictate, e.g. through watchdog timers, reasonableness checks, input data checks, etc.? o Are the responses to failure conditions consistent with safety related requirements?
Qprocess_vv_sw_req_process	
hl_req_acurate/A3-2	Is the high level requirement specification correct, unique and consistent?
hl_req_traceable/A2-11.9	Is tracability between the high level requirement and the final product facilitated?
Qprocess_vv_sw_design_process	
ll_req_acurate/A4-2 ll	Is the design documents a correct and consistent refinement of the high level requirements?
ll_req_ttraceable/A4-6 ll	Is there a clear correspondence between each item of the high level requirements and corresponding elements of the design?
sw_arch_consistent/A4-9	Has an analysis shown that the software architecture does not contain any internal inconsistencies?
Qprocess_vv_sw_coding_process	
code_vs_ll_req/A5-1	Is the code a correct and consistent refinement of the low-level requirements given in design documents?
code_traceable_ll/A5-5	Is there a clear correspondence between each element of the design and corresponding code modules?
Qprocess_test_int_process/ A6	
vv_cases_procedures/A6 11-13	Which types of tests have been performed on the system? o Hardware/software integration testing o Software integration testing o Low level testing Which types of test data selection strategies have been used o Simulation testing o Coverage test o Black box tests o White box tests o Stress tests
vv_cases_procedures_1/A6 11-14	o Have all specified functions been tested? o Have all specified safety actions been tested? o Has fault injection been used to test the robustness of the system sw verification cases and procedures?
Qprocess_sw_CMP	
SLC_env_config_index/A8 11-15	Are all tools used to produce the software, in all life cycle phases, identified, controlled and retrievable?
sw_configuration/ A8 11-16	Is a software product baseline established and placed in a Software Configuration Index?
Qprocess_SQAP	

sqa_exists/ A9	Is the software quality assurance process properly performed and recorded?
Qprocess_cl_process	
certification_feedback/ A10	Has the development been made in co-ordination with, and with feedback from, the certification authorities?
mmi_aspects	Have Human-System Interaction aspects been properly considered during the development of the system?

Questionnaires for "Quality of Product":

Node/table reference	**Question**
Quality_Product	
Qproduct_sw_plan_process	
plan_sw_certification/A1-11.1	Is the system made according to a plan which includes software certification aspects?
sw_verification_plan/ A1-11.3	Is the system made according to a plan which includes software verification aspects?
Qproduct_sw_dev_process	
source_code/ A2-11.11	Does the source code possess the following properties?: o Correct implementation of all low-level requirements o Consistency o Verifiability o Tracability
ex_obj_code/ A2-11.12	o Is the executable object code correct?: o Is the translation of the source code into executable code made in a trustworthy way? o Is the implementation of the executable code onto the target computer made correctly?
Qproduct_vv_sw_req_process A3	
sw_hl_vs_system_req/ A3-1	Do the software high level requirements comply with system requirements?
hl_req_accurate/ A3-2	Is the high-level requirement specification correct, unique and consistent?
hl_req_target_computer/ A3-3	Has the executable code been verified on the target computer?
algorithm_accurate/ A3-7	Have all algorithms used in the program been verified with respect to accuracy and correctness?
Qproduct_vv_sw_design_process/ A4	
ll_req_vs_hl_req/ A4-1 ll	Do the low-level requirements (design documents) comply with the high level requirements?
ll_req_acurate/ A4-2 ll	Are the low-level requirements accurate and consistent?
ll_req_target_computer/ A4-3	Are there any conflicts between the low-level requirements and the hardware/software features of the target computer?
alg_accurate/ A4-7	Have all algorithms used in the program been verified with respect to accuracy and correctness?
sw_arch_hl_req/ A4-8	Is the software architecture compatible with high level requirements?
sw_arch_consistent/ A4-9	Has an analysis shown that the software architecture does not contain any internal inconsistencies?
sw_partitoning_int_confirmed/ A4-13	Is the software partitioning integrity confirmed?
Qproduct_vv_sw_coding_process	
code_sw_arch/ A5-2	Does the source code match the data- and control flow defined in the software architecture?
code_traceable_ll_1/ A5-5	Is there a clear correspondence between each element of the design and corresponding code modules?
code_accurate_consistent/ A5-6	Is it verified that the code is accurate and consistent, also including the attributes: o Stack usage o Fixed point arithmetic overflow and resolution o Resource contention o Worst case execution timing o Exception handling o Use of uninitialised variables or constants o Unused variables or constants o Data corruption due to task or interrupt conflicts
code_vs_ll_req/ A5-1	Is the code a correct and consistent refinement of the low-level

	requirements given in design documents?
Qproduct_test_int_process/ A6	
vv_cases_procedures/A6 11-13	Which types of tests have been performed on the system? o Hardware/software integration testing o Software integration testing o Low level testing Which types of test data selection strategies have been used? o Simulation testing o Coverage test o Black box tests o White box tests o Stress tests
Qproduct_vv_vv_process - tests/ A7	
Test procedure /A7-1	o Is the test performed correctly? o Does the test reflect the actual usage of the final system? o Is the 'oracle' correct?
Test results/A7-2	Are all test results correct, or if not, are the discrepancies explained?
Test completeness	o Have all specified functions been tested? o Have all specified safety actions been tested? o Has fault injection been used to test the robustness of the system sw verification cases and procedures?
Test coverage/ A7-3 - 8	Does the test cover o High-level requirements o Low-level requirements o All software structures o All decisions o All branches
Qproduct_sw_CMP/ A8	
SCM_records/ A8 C11-18	Are all software configuration management activities recorded in the Software Configuration Records?
Qproduct_SQAP/ A9	
sqa_in_practice	Is the software quality assurance process properly performed and recorded?
Software conformity	Does the software development conform to the SQA requirements?
Qproduct_cl_process/ A10	
certification_feedback	Has the development been made in co-ordination with, and with feedback from, the certification authorities?
Qproduct_sw_metrics	If any type of software measurement has been performed, what is the result?
Qproduct_COTS	If the quality of all COTS used in the system has been evaluated, what is the result?
mmi_aspects	Have Human-System Interaction aspects been properly considered during the development of the system?

Questionnaires for "Quality of Analysis":

Node/table reference	**Question**
Quality_Analyses	
Qanalysis_sw_plan_process/ A1	
sw_verification_plan/ A1-11.3	Is the system made according to a plan which includes software certification aspects?
SQA_plan/ A1-11.5	Is the system made according to a plan which includes software verification aspects?
sw_verification_results/ A1-11.14	
Qanalysis_sw_dev_process/ A2	
sw_req_data/ A2-11.9	Does the source code possess the following properties?: o Correct implementation of all low-level requirements o Consistency o Verifiability o Tracability
source_code/ A2-11.11	o Is the executable object code correct?: o Is the translation of the source code into executable code made in a trustworthy way? o Is the implementation of the executable code onto the target

	computer made correctly?
ex_object_code/ A2-11.12	
Qanalysis_vv_sw_req_process/A-3	
hl_req_accurat/ A3-2e	Is the high-level requirement specification correct unique and consistent?
hl_req_target_computer/A3-3	Has the executable code been verified on the target computer?
hl_req_verifiable/ A3-4	Is the high-level requirement verifiable?
Qanalysis_vv_sw_design_process	
ll_req_acurate/ A4-2	Are the low-level requirements accurate and consistent?
ll_req_target_computer/ A4-3	Are there any conflicts between the low-level requirements and the hardware/software features of the target computer?
ll_req_verifiable/ A4-4	Is the high-level requirement verifiable.
sw_arch_hl_req/ A4-8	Is the software architecture compatible with high level requirements?
sw_arch_consistent/ A4-9	Has an analysis shown that the software architecture does not contain any internal inconsistencies?
sw_arch_target_computer/A4-11	Are there any conflicts between the software architecture and the hardware/software features of the target computer?
sw_partitoning_int_confirmed/ A4-13	Is the software architecture verifiable?
Qanalysis_vv_sw_coding_process/ A5	
code_sw_arch/ A5-2	Does the source code match the data- and control flow defined in the software architecture?
code_verifiable/ A5-3	Is the source code verifiable?
code_standards/ A5-4	Does the code conform to the coding standard?
code_traceable_ll/ A5-5	Is there a clear correspondence between each element of the design and corresponding code modules?
Qanalysis_test_int_process/ A6	
vv_cases_procedures/ A6 11-13	Which types of tests have been performed on the system? o Hardware/software integration testing o Software integration testing o Low level testing Which types of test data selection strategies have been used? o Simulation testing o Coverage test o Black box tests o White box tests o Stress tests
Qanalysis_vv_vv_process/A7	
Test procedure /A7-1	o Is the test performed correctly? o Does the test reflect the actual usage of the final system? o Is the 'oracle' correct? o Are all test results correct, or if not, are the discrepancies explained?
Test results/A7-2	Are all test results correct, or if not, are the discrepancies explained?
Test completeness	o Have all specified functions been tested? o Have all specified safety actions been tested? o Has fault injection been used to test the robustness of the system sw verification cases and procedures?
Test coverage/ A7-3 – 8	Does the test cover o High-level requirements o Low-level requirements o All software structures o All decisions o All branches
Qproduct_sw_CMP/ A8	
problem_reports/ A8 11-17	Are all problems revealed during the software development reported, including o Non-compliance with software plans and standards? o Deficiencies of output of software life cycle processes? o Anomalous behaviour of software products? o Are the corrections of all these problems reported?
Qproduct_SQAP/ A9	
sqa_in_practice	Is the software quality assurance process properly performed and recorded?

	Qproduct_cl_process/ A10	
	certification_feedback	Has the development been made in co-ordination with, and with feedback from, the certification authorities?
	Qproduct_sw_metrics	If any type of software measurement has been performed, what is the result?
	mmi_aspects	Have Human-System Interaction aspects been properly considered during the development of the system?
	analysis_team	How good is the analysis team performance?

I

The Use of Bayesian Belief Nets in Safety Assessment of Software Based Systems

(with Gustav Dahll)

In Special Issues of International Journal on Intelligent Information Systems at FLINS'98,
*Int. J. General System*s, **24** (2), pp 205-229, 2000.

Paper I is not included due to copyright.

II

Assessment of programmable systems using Bayesian belief nets

Submitted and accepted for *Journal Safety Science*, Special Issue on Safecomp-2000.

To be published 2002.

Extended version of the paper: Gran, B.A., Dahll, G., Eisinger, S., Lund, E., Norstrøm, J., Strocka, P., and Ystanes, B.: Estimating Dependability of Programmable Systems Using BBNs. *Computer Safety, Reliability and Security, Proceedings from Safecomp 2000,* (*LNCS 1943*), Koornneef F. and van der Meulen, M. (Eds), Springer, Berlin , pp. 309-320, 2000.

URN:NBN:no-2126

ASSESSMENT OF PROGRAMMABLE SYSTEMS USING BAYESIAN BELIEF NETS

Bjørn Axel Gran
OECD Halden Reactor Project, Institutt for energiteknikk,
P.O.Box 173, N-1751 Halden, Norway
<bjorn.axel.gran@hrp.no>

Abstract.

This paper discusses some software safety standards, with respect to how they can be used to measure software safety. The possibility to transfer the requirements of a software safety standard into Bayesian Belief Nets is also investigated. The aim is to utilise the BBN methodology and associated tools, to transfer the software safety measurement into a probabilistic quantity. In this way software can be included in probabilistic safety analysis of the total programmable system. A project was performed in which the method was applied for evaluation of a real, safety related programmable system that was developed according to the avionic standard DO-178B. The test case, the standard, and the BBN methodology are shortly described, followed by a description of the construction of the BBN used in this project. Also a summary of some of the findings and experiences from the study is provided.

1. Introduction

During the last decades the digital revolution has made society increasingly dependent on programmable digital equipment. Such equipment has to an increasing degree become of importance for our safety, and one must therefore trust that it performs its tasks in a correct and reliable way. Traditionally, there are various kinds of equipment one places confidence in, from car breaks and train stop signals to emergency shut down systems in nuclear power plants. The introduction of digital technology in safety critical systems has many advantages, both concerning flexibility and reliability. In later years it is also becoming a necessity, as conventional equipment is often no longer produced. There is, however, one aspect that has caused a certain reluctance to use programmable equipment in safety critical systems, viz. the complexity of safety assessment of the software in these systems.

The research programme at the Halden Project on software safety assessment was argued through a joint project with Kongsberg Defence & Aerospace AS (KDA) and Det Norske Veritas (DNV) (Gran et. al. 2000). The objective of this project was to investigate the possibility of combining the Bayesian Belief Net (BBN) methodology with a software safety standard, DO-178B (RTCA/DO-178B) for safety analysis of a programmable system. Please note that this paper represents by no means any official policy of KDA.

2. Standards and Guidelines for Safety Related Software

Recently much effort has been taken to make international standards and guidelines for the development of programmable systems for safety related applications. A generic standard is IEC 61508 'Functional safety of electrical/electronic/programmable electronic safety-related systems' (IEC 61508). This standard will constitute a framework for other, more specific standards. Examples of branch specific standards are IEC-880 (IEC880), IAEA software safety guide (IAEA ID NS 264) in the nuclear industry, and DO-178B for safety critical software in civil aviation. A general impression from these standards is that they are built on the same basic framework, and follow the same principles, although they may differ in the aspects they put special emphasis on. The common framework is expressed

in a software lifecycle model, where the different stages in the system development are placed. For each of these stages requirements or recommendations are given. The division into stages, and the starting and end stages of the lifecycle model, may differ between the standards. The standards also differ in the requirements they are particularly emphasising. Even if different standards vary in the degree of detail, a general characteristic of software standards is that the requirements and recommendations are of qualitative nature, in distinction from hardware standards where there in general are clear and objective requirements. Ideally a requirement in a standard should be objective in two ways: the requirement itself should be objective in an unambiguous way, and there should be an objective way to state whether the requirement is fulfilled or not. This problem is thoroughly discussed in (Neil and Fenton 1998).

A question in connection with software safety standards is whether the fulfilment of their requirements actually guarantees that the system is safe. A standard is in general developed, over a long time period, by a group of experts. Other experts around the world then review the draft international standards. Such a thorough preparation by internationally renowned experts should strongly indicate that a system made according to this standard is safe. There is, however, no objective evidence that guarantees that this is true. Even the views of experts are to a large degree based on judgement. In addition, the experts in this field constitute a fairly limited society, so it is likely that they are strongly influenced by each other. Of course, the safety assessment is not necessarily based on qualitative judgement only. There are analytical methods like e.g. fault tree analysis, reverse engineering, formal verification, etc., as well as statistical reliability evaluation based on operating experience or testing. Testing is essential for a safety assessment of the final product. A general impression is, however, that the standards are not very precise on required strategies for testing, but leave this to human judgement.

A conclusion from these considerations is that it is not straightforward to decide objectively whether a software-based system is sufficiently safe on the basis of the criteria given in a standard only. There is a need for a systematic decision support system associated with a standard, which can help the licensing authority or any safety assessor. It is suggested that Bayesian Belief Nets and associated tools can provide this help.

3. Bayesian Belief Nets

Applying Bayesian Belief Nets (BBN) is a method to make reliability estimates based on information of disparate nature, by combining quantitative observations and human judgments. The objective of using BBNs in software safety assessment is to show the link between observable properties and the confidence one can have in a system. The theory of BBNs is well established, and the method has been applied with success in various areas, including medical diagnosis and geological exploration. Recently, work has been performed in the European projects SERENE (1999) and IMPRESS (2000) and research has resulted in various papers, e.g. by Bertolino and Strigini (1996a, b, 1998), Neil et. al. (1996a, b, 1998, 2000, 2001), Fenton and Neil (1999), Littlewood and Wright (1995, 1997) and Dahll and Gran 2000). There are tools to perform computations with BBNs. Here it is particularly referred to the HUGIN tool (HUGIN; Alderyd et al. 1993; Jensen 1996) and the SERENE methodology (SERENE).

A Bayesian Belief Net (BBN) is a connected and directed graph, consisting of a set of nodes and a set of directed arcs between them. Uncertain variables, both events as well as singular propositions, are associated to each node where the uncertainty is expressed by a probability density. The probability density expresses our belief or confidence in the various possible outcomes of the variable. This probability depends conditionally on the status of

other nodes at the incoming edges to the node (the parent nodes). Some nodes are denoted as 'observable'. They represent the different observable properties of the software system and its development. Network edges model relations between adjacent nodes, and the strength of these relations is represented as conditional probability distributions. The computation of the belief about a specific node (target node) is based on the rules for conditional probability calculations backward and forward along the edges, from the observable nodes, through the intermediate nodes to the target node (Casella and Berger 1990; Cowell et. al. 1999, Welsh 1996).

The construction of the BBN is normally made gradually. Information about the software system and the standard (such as DO-178B) is collected and expressed via the nodes. The nodes are connected together to a directed graph that expresses the conditional relationship between the variables. The aim is to combine information in the net. One way is to start from a target node and draw edges to influencing nodes. To decide the direction of an edge, one follows the causal direction. However, this direction is not always obvious, in particular between nodes representing qualitative variables. In these cases the direction of the arrow often goes from higher to lower abstraction level, or from the general to the detailed concept. For computations of a realistic BBN computer tools are necessary.

4. The test case M-ADS and the standard DO-178B

4.1 The M-ADS Airborne Equipment

The M-ADS airborne equipment was designed by KDA for installation in helicopter aircrafts. The system provides air traffic services transmitting aircraft parameters upon request from the air traffic control where personnel will request positioning data. The M-ADS system is designed to automatically transmit flight information via data link to one or more requesting air control centres. M-ADS uses existing avionics on board the aircraft to provide aircraft position, speed and additional optional data. The most important data are the aircraft position, position accuracy, altitude and time stamp for the data validity. The main purpose of the M-ADS Airborne Equipment is to aid in a rescue operation if the helicopter has made an emergency landing on the sea. A correct localization is necessary for a successful rescue operation, the system is therefore safety critical, and the system had to be approved by the Norwegian Civil Aviation Authority. The software development process was performed according to the DO-178B standard.

4.2 The DO-178B Standard

The DO-178B standard is a mandatory guideline for the production of safety critical software for airborne systems. This guideline was chosen for the study since the M-ADS system is applied in civil aviation, and was previously qualified on the basis of this standard. DO-178B discusses aspects of airworthiness certification that pertain to the production of software for airborne systems and equipment used in aircrafts. To aid in understanding the certification process the system life cycle is briefly discussed to show relationship to the software life cycle process. It does not provide guidelines concerning the structure of the applicant’s organization, relations to suppliers and personnel qualification criteria.

DO-178B defines, similar to IEC 61508, a set of five software levels (A, B, to E), based on the contribution from software to potential failure conditions as determined by the system safety assessment process. The main recommendations in DO-178B are given in a set of 10 tables. Each table relates to a certain stage in the development and validation process, and contains a set of objectives. The 10 stages in the development and validation process are:

- A1 Software planning process.
- A2 Software development process.
- A3 Verification of outputs of software requirements process.
- A4 Verification of outputs of software design process.
- A5 Verification of outputs of software coding & integration process.
- A6 Testing of outputs of integration process.
- A7 Verification of verification process results.
- A8 Software configuration management process.
- A9 Software quality assurance process.
- A10 Certification liaison process.

A difference between the two standards is that most of the requirements are mandatory in IEC61508, while the requirements are guidelines in DO-178B (Neil and Fenton 1998).

5. Construction of the BBN for M-ADS

5.1 The Construction Process

The basic philosophy of the proposed process is to relate the safety of the system to the fulfilment of the requirements in an internationally accepted safety standard. This philosophy can of course be questioned, but such standards are based on consensus among experts in the area relevant for an actual safety critical system. Even if conformance to a safety standard does not imply safety, it is a strong indication of the effort put into making the system safe. This indication can also be used as prior probability in a Bayesian model for a further safety assessment based on safety testing. Recall that one want to achieve a way of stating how well the development of a safety critical system conforms to the requirements of the standard. However, such standards do not contain any measures of conformity, but rather a large number of requirements of rather disparate nature, which should be fulfilled. The objective of the project was to use BBN methodology to construct such a measure.

The first action in the construction was to identify the main characteristics that may influence the dependability of a system. One can distinguish between characteristics that are related to the system itself and characteristic that are related to the interaction between the system and its environment (usage of the system, potential hazards etc.). The former includes quality characteristics, which were divided into four types:

- *Quality of the producer*. (*Qproducer*) This includes the reputation and experience of the producer, quality assurance policy, quality of staff etc.
- *Quality of the production process*. (*Qprocess*) A high quality implies that the system is developed according to guidelines for good software engineering, that all phases are well documented, and that the documentation shows that the system at all development phases possesses desirable quality attributes as completeness, consistency, traceability etc.
- *Quality of the product*. (*Qproduct*) This includes quality attributes for the final product, as reliability, simplicity, verifiability etc.
- *Quality of the analysis*. (*Qanalysis*) This includes all activities performed to validate the correctness of the system during all stages of the system development. Such activities may include model checking of the specifications, inspections and walkthroughs of the documentation, static analysis of code and testing of the system.

The BBN was constructed in two levels. The higher level shows how nodes representing the four types of characteristics listed above are combined with other nodes in

the net and lead to nodes representing the reliability and safety of the system. At the lower level there are four BBNs, where the four characteristics are represented as top nodes.

5.2 The Higher Level BBN

The higher-level network consists of two parts: the *quality-part* (or soft-evidence part) and the *testing-part*, as presented in Figure 1.

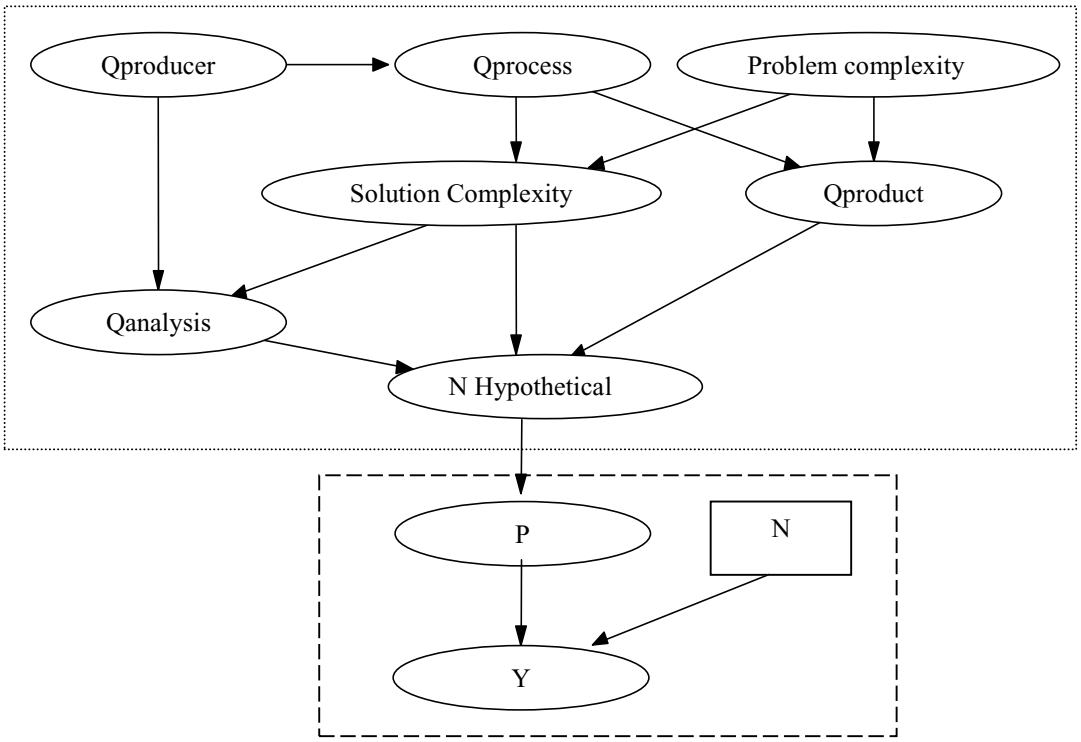

Figure 1. The higher-level network, the nodes enclosed by '...' represent the 'quality-part', and the nodes grouped by '---' represent the 'testing part'.

The *quality-par*t consists of the four quality nodes listed in the previous section. In addition it includes the nodes *problem complexity* and *solution complexity*. The initial nodes or top nodes are the quality node *Qproducer* and the *problem complexity*, where the latter is an attribute of the system to be developed, and can be measured. It is assumed that the *Qproducer* directly influences the *Qprocess*, and that the *solution complexity* is influenced by the *problem complexity* and the *Qprocess*. The same dependencies are assumed for the *Qproduct*. The product quality depends upon how difficult it is to fulfil the requirements (the complexity of the problem), and upon the ability of the development process to handle complex systems. The *Qanalysis* is assumed to be influenced by the *Qproducer*, how well prepared the organization is to perform an analysis, and the *solution complexity*, how difficult it is to analyse. All these assumptions are in accordance with BBNs presented in the SERENE project and networks presented by the HRP-project (Dahll and Gran 2000). The higher-level network leads to an end node *N-hypothetical*. The intention is to express that the information in the quality-part is equivalent to that the system is tested with *N* randomly chosen inputs without failure. The computation of the 'quality-part' of the BBN is based on observations in the lower level networks, and conditional probability tables associated with the edges in the BBN.

The *testing-part* represented by the node '*Y*: failures in *N* new tests', describes the

connection between hard evidences, *Y=0* failures in *N* tests, and the failure probability of the system (in the context, usage, environment, etc. the system is tested). The failure probability can be interpreted either as a number of failures on a defined number of demands, or as a number of failures on a defined time period. For the defined number of demands *N* with the constant failure probability *P* the random number of failures *Y* has a binomial distribution.

The failure probability *P* can be linked to a node representing the system safety, which in addition is also depending on the usage of the system and the consequences of eventual failures. In the described project no modelling of the dependencies with respect to the system safety was made. Of this reason these nodes are not included in Figure 1, and no calculations related to this node were done.

The link between the *quality-part* and the *testing-part* is given by the edge between *N-Hypothetical* and *P*. The dependency associated with this edge, leading to the results presented, was given by '*P = 1/ N-Hypothetical*'. However, it was applied in the way that P(*P*∈[p,q]) = P(*N-Hypothetical* ∈[1/q,1/p]). The same dependency would have arisen by assuming direct dependencies between P and the nodes *Qanalysis*, *Solution Complexity* and *QProduct*. For the expert team it was, however, conceptual easier by this two-step procedure.

5.3 The Lower Level BBN Identification Based on DO-178B

The lower-level BBNs were constructed by applying the quality characteristics with top-nodes in four BBNs. Each top node was linked to intermediate nodes representing the 10 process stages of DO-178B (A1 – A10). Each of these nodes was again linked to other intermediate nodes representing the objectives of the tables. In addition some additional objectives to be considered in the networks were identified.

The further step was to identify a list of questions to each objective. These questions were based on the understanding of the text in the main part of DO-178B, and they were in general formulated so that the answer can be given by a *yes* or a *no*. However, as the questions often are of a qualitative nature, it may be difficult to give a straight answer. It was therefore possible to answer the question with a number between 0 and 1 as an expression of the strength in the belief that the answer is *yes* (1) or *no* (0). This number was then used as input to the computation of the BBN. In some cases a question was linked directly to an end-node, in other cases the questions introduced a list of help questions to be considered, when assessing ones belief in answering *yes* on the end-node.

An illustrating example is given in Figure 2. Start with the top node *Qanalyses* and continue through the node referring to stage A2 '*software development process'*, to the node '*does the source code',* referring to the objective A2-11.11. To this node there is associated four questions:

- Does the source code possess correct implementation of all low-level requirements?
- Does the source code possess consistency?
- Does the source code possess tracability?
- Does the source code possess verifiability?

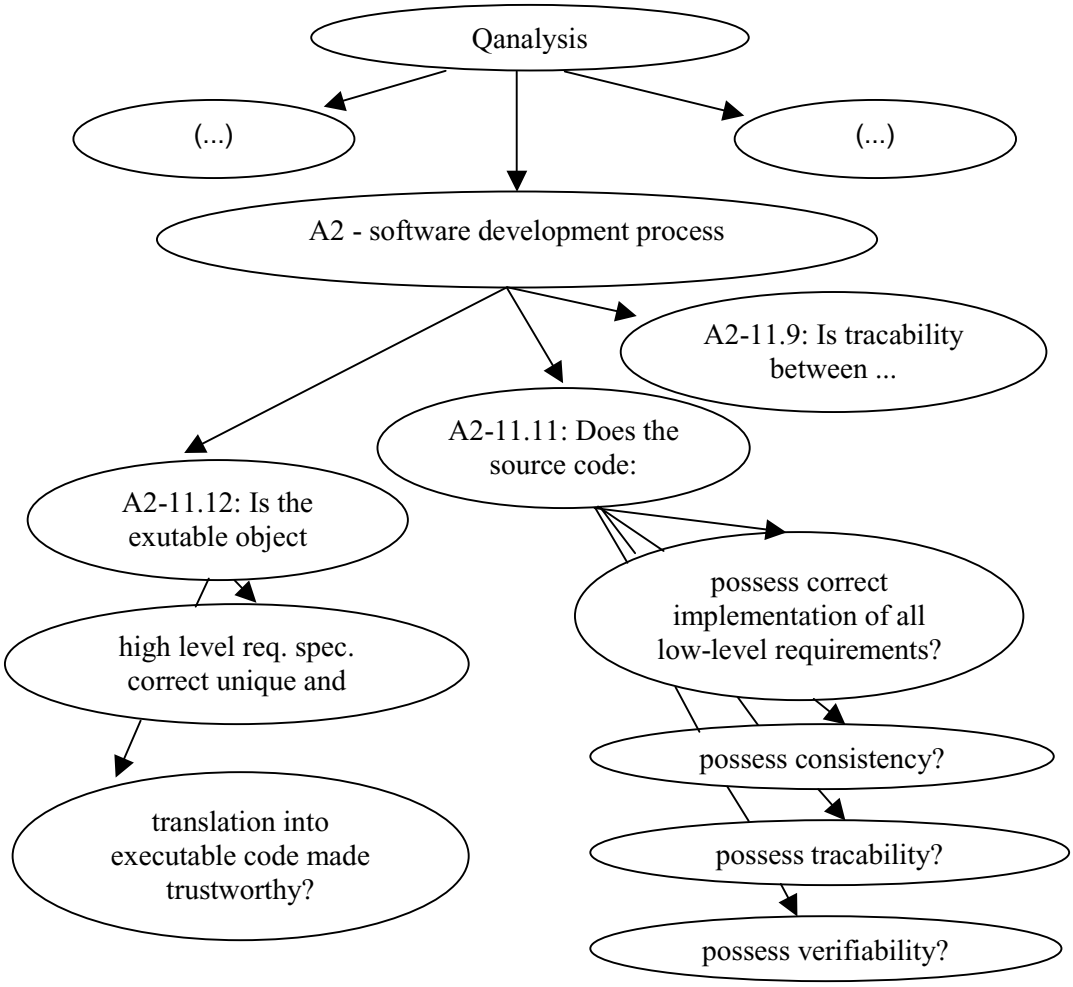

Figure 2. A part of the BBN for the *Quality of the analysis*

5.4 The prior and conditional probability tables

The probability distribution functions (pdfs) to the nodes and edges were based on discrete probability tables. The advantages of the pdfs in discrete form are that it becomes conceptually easier in an expert judgement to assign discrete values, and that it makes the computation simple. An expert group assessed the conditional probability tables (cpt) to the nodes and edges. This elicitation was done as a brainstorming exercise. In general, this means that for each node, the expert group had to assess two conditional probabilities of the type P(good measurement | good quality) and P(bad measurement | bad quality). Based on general knowledge and experience in software development and evaluation, it was mostly done by ranking the importance of the different sub nodes, and giving them probabilities from a predefined such as {0.5, 0.7. 0.9, 0.95, 0.99}. The probability tables representing the higher-level network, leading to the results in the next section, are presented in the Appendix. However note, since the objective of this project was to investigate the possibility of combining the BBN methodology with a software safety standard, the tables are not validated. For the lower-level network about 130 conditional probability tables were assessed.

6. Computation on the M-ADS BBN

6.1 Description of Assumptions, Restrictions and Scenarios

As the project created a rather complex BBN (or system of BBNs), there were a large number of conditional probability tables to be assessed. These were estimated based on judgments in a brainstorming activity among the project participants. Of course, this opens for some subjectivity. On the other hand, some of the project members were considered as experts within their field.

After observing the results of four initial scenarios: *no observations*, *KDA observations*, *best case* and *worst case*, some additional scenarios were defined. For all scenarios observations had been made with respect to the four quality characteristics and the node *P*. In this paper some results from these scenarios are presented:

- Partial: the effect of observations during only one stage in the development and validation process, such as A1, A2... A10.
- Incremental: the effect of first observing during the stage A1, then A2, and so on, representing the fact that the processes can be viewed as sequential.
- Sensitivity analysis for the node *P* given different values on *N*.

6.2 Observations on the End Nodes

The observations were done by KDA through several interview sessions with experts involved in the project. Totally, experts representing the software design and coding role, as well as project management role, were involved. In each session the questions associated with the end nodes in the network were used to assess the module in view of the scope defined by the node. The answers were, as discussed in section 5.3, given as weighted values on the scale from zero to one. In general the value zero means objective achieved with poor quality, while the value one means objective achieved at highest level of quality. There also were a few cases where a score, say 0.95, indicated objective achieved at highest level of quality for 95% of the modules. As an example refer to a question for the BBN for *Qanalysis*: 'is the software quality assurance process properly performed and recorded?' The answer, 0.95, means that the expert board judged that software quality assurance process is properly performed and recorded for 95% of the modules.

6.3 Results

6.3.1 The Partial Scenarios

The effect of the observations during only one stage in the development and validation process showed that with respect to the *Qproducer* the processes with largest effect were the software planning process (A1), the software development process (A2) and verification of process results (A7). Note, however, that the effects were approximately the same for the other processes. With respect to the *Qprocess* the processes with largest effect were verification of outputs of software requirements process (A3), software configuration management process (A8) and certification liaison process (A10), while 'other aspects' had lowest effect.

With respect to the *Qproduct* the processes with largest effect were the process A7 and 'other aspects' including aspects such as e.g. human machine interfaces. Quite low effect was observed for the processes: verification of outputs of software design processes (A4) and the process A10. In particular it was noted that one observation, with respect to process A4, was given the value 0. This value corresponded to a negative answer to the question 'is the

software partitioning integrity confirmed?' Whether this answer was meant to be negative; i.e. that this question is of importance to the reliability of the product, or if this question was ranked as irrelevant, was not further discussed. In the latter case it would have been better not to give any value to this observable node at all. This is equivalent to cutting the edge to this node. A walk-through of the observations (Gran 2001) also identified 6 additional questions with questionable observations. Of these 6 questions one belonged to the process A10. The result of correcting these faults was that the surprising low effect for A4 and A10 disappeared. Further, the processes with low effect were now observed to be A1 and A8. These were both also identified as contributors to low effect for the other quality characteristics. On the other hand, if one assumes that the observations in fact were negative or as low as entered, then there is identified a case where only a few negative observations can lead to negative significant changes in the partial scenarios.

With respect to the *Qanalysis* the process 'other aspects' had largest effect, but also all the other processes had a large effect. For the node *P* the largest effects were for the processes A3, A7 and "other aspects", while the lowest effects were for the processes A4 and A10. These results are in accordance with the dependency between the node *P* and the nodes *Qproduct* and *Qanalysis*.

6.3.2 The Incremental Scenarios

The observations could also be added subsequently, first during process A1, then A2 and so on. This illustrates how the posterior probability distributions change from the initial prior values towards a scenario given by all the KDA observations. For the *Qproducer* the expected value came up to a *top level* already after observations during processes A1 and A2 were made. This does not mean that the quality of the producer will remain on this level independent of other additional observations, but means that making additional 'good' observations does not change our posterior results. With respect to the nodes *Qprocess*, *Qproduct*, and *Qanalysis* we had to make positive observations on all the processes A1 towards A8 before the posterior probability distributions achieved the top level. For the node *P*, the posterior distribution was at its top level after observations were made during process A1 up to A3. This is the similar effect as for the *Qproducer*. Note that, although there is no direct link between these two nodes, they behave in the same manner due to the propagation of positive measurements.

6.3.3 Sensitivity Cases

A sensitivity analysis was performed for the node *P* given future observations on the node *N (new tests)*. That is, with all the observations on the quality characteristics, represented in the node *N_hypothetical*, different measurements were made on the node *N*. Note that making a measurement equal to *m* assumes that a failure occurred after *m* failure free tests. The posterior probability distributions for *P* are shown in Figure 3. Compared to testing alone, these results show that observing *m* failure free tests, where *m* is higher than the hypothetical *N* failure free tests, will increase our belief in a shift left of the distribution for *P*. In the same way, observing *m* lower than *N* will shift it right, due to the situation that our prior belief is not in accordance with the real measurements.

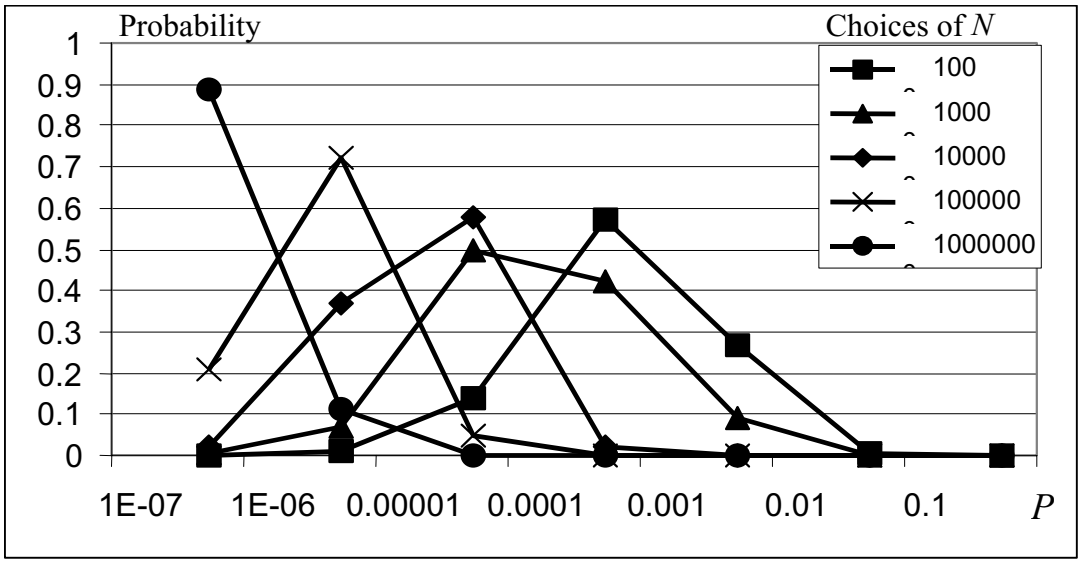

Figure 3. The posterior probability distribution for *P* for different number of new tests.

7. Experiences from the Evaluation

The BBN methodology was mainly developed and applied in the AI society, but more recently there have been attempts to apply it for estimation of the dependability of programmable safety related systems. From the experiences from applying the methodology on the M-ADS system, including a BBN modelling of the DO-178B guideline, we can conclude that the BBN methodology offers a systematic way to combine quantitative and qualitative evidences of relevance for the safety assessment of programmable systems.

Conceptually, estimation of the dependability of programmable systems is nearly impossible to compute, since many of the characteristics to be considered are of qualitative nature and not directly measurable, but have to be estimated. The most difficult activity was to perform the expert judgment, in particular in the assignment of values to the conditional probability distributions. Even if some of the project members can be considered as experts within their fields, it is highly recommendable to make use of some expert judgment tools or expert judgment expertise.

Another observation is that the establishment of the BBNs and prior probability distributions was rather time consuming. However, the process of building up the network, e.g. by making questionnaires, and the elicitation of the prior distributions were related to DO-178B, and not to the actual system. This implies that the network and questions are of a general nature, and can be reused in many applications. They can also be gradually improved based on experience. Furthermore, it may be feasible to transfer this knowledge to other safety related software engineering standards. It should also be remarked, that the project provided an improved understanding of the DO-178B.

The experiences from collecting the different observable properties to be used in the calculations, and performing the calculations, are that these tasks are fairly easy and not so time consuming. Since tool support is necessary, we applied the HUGIN tool and the SERENE methodology (Demo 1.0), and found them satisfactory.

Knowledge within BBN and probabilistic theory is of great advantage in the construction of the networks and the assessment of the probability distributions. This knowledge is also an advantage in the evaluation of the results from the computations.

Another finding from the project is that the BBN methodology is not only applicable in the final assessment of a system, but could be used at all stages throughout the whole software lifecycle. The network could e.g. in this specific project be used to evaluate the difference between two different safety levels before any other measurements are collected. In this way it is possible to make assessments about the system before it is even designed or implemented. In such a way corrections to e.g. the development process can be made early in the project, in order to be able to reach specific objectives of the final product.

8. Further Work

The work presented in this paper is a part of a long-term research activity by the OECD Halden Reactor Project (HRP), for example is a further analysis of the results presented in (Gran 2001), and an approach to merge the networks presented here with a network representing evidence from disparate operational environments is evaluated through a joint-project between HRP and VTT Automation, Finland (Gran and Helminen 2001).

It has also been identified that the sub networks based upon DO-178B should be revised and more clearly defined. That is, it should be checked that each selected node belongs to the network, and one should also check for missing nodes. Further, the meaning of each node state should be more clearly defined. The application of expert judgment tools in order to obtain better expert judgments on the prior probability distributions is a related possibility that should be exploited. The possibility of validating the BBNs or a sub network including the topology (which node is connected to which node), and the probability distributions (the probability of observing a certain state given that the parent node is in a certain state) should also be investigated.

Acknowledgement

I want to acknowledge the rest of the team taking part in the modelling and evaluation of the M-ADS case: Siegfried Eisinger, Det Norske Veritas; Eivind J. Lund, Jan Gerhard Norstrøm, Peter Strocka and Britt J. Ystanes, all Kongsberg Defence & Aerospace; and Gustav Dahll, OECD Halden Reactor Project. I also want to acknowledge Martin Neil for advice and interest with respect to the use of SERENE (Demo 1.0), and Hugin Expert A/S for allowing me the use of the HUGIN tool for my Ph.D. work, and for supporting help.

References

Aldenryd, S.H., Jensen, K.B., and Nielsen, L.B., 1993, *Hugin Runtime for MS-Window*, Tool made by Hugin Expert a/s, Aalborg, <http://www.hugin.dk>

Bertolino, A., and Strigini, L., 1996a, Predicting Software Reliability from Testing Taking into Account Other Knowledge about a Program. *Proceedings 9th International Software Quality Week* (Software Research Institute, San Francisco).

Bertolino, A., and Strigini L., 1996b, Acceptance Criteria for Critical Software Based on Testability Estimates and Test Results. *Proceedings SAFECOMP96, 15th International Conference on Computer Safety, Reliability and Security*, Schoitsch (ed) (Springer-Verlag), pp 83-94.

Bertolino, A., and Strigini, L., 1998, Assessing the risk due to software design faults: estimates of failure rate vs. evidence of perfection, *Software Testing, Verification and Reliability*, **8**(3), 155-166.

Casella, G., Berger, R. L., 1990. *Statistical Inference*, Wadsworth & Brooks/Cole Advanced Books & Software.

Cowell, R.G., Dawid, A.P., Lauritzen, S.L., and Spiegelhalter, D.J., 1999, *Probabilistic Networks and Expert Systems* (Springer-Verlag).

Dahll, G., and Gran, B.A., 2000, The Use of Bayesian Belief Nets in Safety Assessment of Software Based Systems. *Special Issues of International Journal on Intelligent Information Systems at FLINS'98, Int. J. General Sys*tems, **24**(2), 205-229.

Fenton, N., and Neil, M., 1999, A Critique of Software Defect Prediction Models, *IEEE Transactions on Software Engineering*, **25**(5), 675-689.

Gran, B.A. 2001. Applying Bayesian Belief Net in Software Safety Assessment on a Real, Safety Related Programmable System. Paper accepted for ESREL 2001, 16.-20. September 2001, Torino, Italy.

Gran, B.A., Dahll, G., Eisinger, S., Lund, E., Norstrøm, J., Strocka, P., and Ystanes, B., 2000, Estimating Dependability of Programmable Systems Using BBNs. *Computer Safety, Reliability and Security, Proceedings from Safecomp 2000, (Lecture Notes in Computer Science 1943)*, Koornneef and van der Meulen (Eds) (Springer), pp. 309-320.

Gran, B.A., and Helminen, A., 2001, A Bayesian Belief Network for Reliability Assessment, *Computer Safety, Reliability and Security (Lecture Notes in Computer Science 2187)*, Voges (Ed) (Springer), pp. 35-45.

IAEA, ID NS 264, 1999. Software for Computer systems Important to Safety in NPPs: A Draft Safety Guide.

IEC publication 61508, 2000, Functional safety of electrical/electronic/programmable electronic safety-related systems.

IEC 880, 1986, Software for computers in the application of industrial safety related systems.

IMPRESS, 1999, Improving the software process using Bayesian nets. EPSRC project nr. GR/L06683, <http://www.csr.city.ac.uk/csr_city/projects/impress.html>.

Jensen, F., 1996, *An Introduction to Bayesian Networks*, (UCL Press, University College London).

Neil, M., Littlewood, B., and Fenton, N., 1996a, Applying Bayesian Belief Nets to Systems Dependability Assessment, *Proceedings of 4th Safety Critical Systems Symposium*, (Springer-Verlag), pp. 71-93.

Neil, M., and Fenton, N., 1996b, Predicting Software Quality using Bayesian Belief Networks, *Proceedings of 21st. Annual Software Engineering Workshop*, (NASA Goddard Space Flight Centre), pp. 217-230.

Neil, M., and Fenton, N., 1998, A strategy for improving safety related software engineering standards, *IEEE Trans. on SW Eng*., **24**(11).

Neil, M., Fenton, N., and Nielsen, L., 2000, Building large-scale Bayesian Networks, *The Knowledge Engineering Review*, **15**(3), 257-284.

Neil, M., Fenton, N., Forey, S., and Harris, R., 2001, Using Bayesian Belief Networks to Predict the Reliability of Military Vehicles, *IEEE Computing and Control Engineering*, **12**(1), 11-20.

RTCA/DO-178B, 1999, Software Considerations in Airborne Systems and Equipment Certifications (Guideline).

SERENE, 1999, Safety and Risk Evaluation using Bayesian Nets. ESPRIT Framework IV nr. 22187, <http://www.hugin.dk/serene/>.

Welsh, A. H., 1996, *Aspects of Statistical Inference*, (Wiley & Sons).

Appendix: The probability tables for the higher-level network (7 tables)

Quality of producer	5 good	.06
	4	.44
	3	.45
	2	.05
	1 bad	.01

Problem complexity	low	.167
	medium	.500
	high	.333

Quality of producer		5 good	4	3	2	1 bad
Quality of process	good	.90	.80	.50	.10	.05
	bad	.10	.20	.50	.90	.95

Problem complexity		low		medium		high	
Quality of process		good	bad	good	bad	good	bad
Quality of product	good	.90	.30	.60	.40	.30	.10
	bad	.10	.70	.40	.60	.70	.90

Problem complexity		low		medium		high	
Quality of process		good	bad	good	bad	good	bad
Solution complexity	low	.90	.70	.10	.10	.01	.00
	medium	.10	.20	.80	.60	.19	.05
	high	.00	.10	.10	.30	.80	.95

Solution complexity		low					medium					high				
Quality of producer		5	4	3	2	1	5	4	3	2	1	5	4	3	2	1
Quality of analyses	good	.99	.95	.90	.70	.30	.95	.90	.70	.30	.10	.80	.70	.50	.30	.10
	bad	.01	.05	.10	.30	.70	.05	.10	.30	.70	.90	.20	.30	.50	.70	.90

Quality of product		good						bad					
Solution complexity		low		medium		high		low		medium		high	
Quality of analyses		g	b	g	b	g	b	g	b	g	b	g	b
N hypothetical	<1000000, Inf>	.01	.01	0	0	0	0	0	0	0	0	0	0
	<100000, 1000000]	.70	.50	.01	.01	0	0	0	0	0	0	0	0
	<10000, 100000]	.25	.40	.70	.50	.01	.01	.01	.01	0	0	0	0
	<1000, 10000]	.04	.09	.25	.40	.70	.50	.70	.50	.01	.01	0	0
	<100, 1000]	0	0	.04	.09	.25	.40	.25	.40	.70	.50	.10	0
	<10, 100]	0	0	0	0	.04	.09	.04	.09	.25	.40	.80	.10
	[0, 10]	0	0	0	0	0	0	0	0	.04	.09	.10	.90

III

The use of Bayesian belief networks for combining disparate sources of information in the safety assessment of software based systems

Submitted and accepted for *International Journal of Systems Science,* Special Issue on Intelligent Product Support Systems.

To be published 2002.

The use of Bayesian belief networks for combining disparate sources of information in the safety assessment of software based systems

Bjørn Axel Gran

OECD Halden Reactor Project,
Institutt for energiteknikk,
P.O.Box 173, N-1751 Halden, Norway
<bjorn.axel.gran@hrp.no>

ABSTRACT: The paper discusses how disparate sources of information can be combined in the safety assessment of software-based systems. The emphasis is put on an emerging methodology, relevant for intelligent product support systems, to combine information about disparate evidences in a systematic way based on Bayesian Belief Networks. The objective is to show the link between basic information and the confidence one can have in a system. It is also illustrated and discussed how to combine the Bayesian Belief Net (BBN) method with a software safety standard (RTCA/DO-178B) for safety assessment of software-based systems. Finally the applicability of the BBN methodology, and experiences from co-operative research work together with Kongsberg Defence & Aerospace and Det Norske Veritas, and ongoing research with VTT Automation are presented.

1 Introduction

With the use of programmable equipment in safety critical systems a new aspect was introduced, to produce safe software. Therefore there is in many application areas necessary with a thorough safety assessment of the system, including intelligent product support systems, for a final acceptance or licensing of the system. For a hardware component, even in a safety critical system, it is accepted to assume that a failure can occur during the lifetime of the system, given that the expected frequency and/or consequence of the failure is sufficiently low. The reliability of a hardware system is thereby based on failure statistics, i.e. one measures the failure frequency in standard components and computes the system reliability on the basis of this, although that this practise may ignore the inherent faults in the hardware. The characteristics of software make it difficult to carry out such a reliability assessment. Software is not subject to ageing, and any failure that occurs during operation is due to faults that are inherent in the software from the beginning. Furthermore, any randomness in software failure is due to randomness in the input data, because software behaviour change over time due to maintenance activities, or due to the fact that environments, such as hardware, operating system and user needs, change over time. As a consequence, there is a problem with the assessment and licensing of systems, both hardware and software, with inherent faults.

As discussed by Dahll and Gran (Dahll and Gran 2000) one can distinguish between three principles for licensing: rule based, risk based, and judgement based. Rule based licensing implies that an assessor checks that a system fulfils a set of criteria given in a safety standard. The rules are easy to follow for the developer and easy to check for the assessor. On the other side, this method easily gets very rigid and inadequate to handle new technology. The rules for safe software are normally based on consensus among experts of what is required for safety critical software. This is expressed through standards and

guidelines. In a risk based assessment the objective is rather to base the licensing on assessing the probability of potential risks associated with the system. This means to identify potential hazards, and demonstrate that the probabilities of these hazards are kept under a certain safety integrity level. In practice, however, the assessment of safety critical software are often faced with the problem of approving systems for which there are no clear rules, and for which it is difficult to apply probabilistic methods. The rules given in standards and guidelines are often imprecise, or they are not directly applicable for an actual system. One possibility for assessors and licensing authorities is to make their judgement based on the opinion of experts in various fields, including process knowledge, reliability engineering, human factors etc. The combined judgement of the different evidences about the system and its environment constitutes the basis for approval or not.

Another approach to assess software with inherent faults is to apply various reliability growth models (Xie, 1991). They are, however, mainly applicable to large commercial systems, and not to safety critical software. The main reason is that a computer program implemented in a safety critical system presumably contains no known faults, since any revealed fault would be corrected. There is a possibility that it contains unknown faults. An alternative reliability measure is then the confidence in fault freeness of the program, or more generally the upper limit of the 'bug-size' (Voas et al. 1993, Gran and Thunem 1998). A way to measure this confidence is based on statistical testing. The validity of this measurement is, nevertheless, highly dependent on a proper choice of test data (Leveson 1995). Another problem with these measures is that they do not take into account that there are several factors that are important to software reliability (Dahll 1997), even if they cannot be put directly into a reliability formula. Some of these are of qualitative nature, like the producer's reputation, the development quality etc. Others are measurable, but not directly connected to reliability estimation, like program size, program complexity etc. The connection between these quantities and software reliability is also of qualitative nature. It is also suggested to apply traditional methods in probabilistic safety assessment (PSA) to software (Leveson 1995, Dahll, 1997, Cudleigh and Catmur 1992). As reasons for this choice it is argued that these methods are well tried, standardised, documented and familiar to the customers (Stålhane 1997). Furthermore, it allows the customer to contribute with their knowledge about the system. An approach of combining traditional methods for risk analysis with semi-formal modelling is argued through the description of the EU-project CORAS (IST-2000-25031).

The focus of this paper is on the use of Bayesian Belief Nets (BBN) to combine evidences from several information sources in the safety assessment of software based systems. This methodology has mainly been developed and applied in the AI society. More recently, it has also been applied to software safety assessment. Work in this area has been performed in the European projects SERENE (1999), IMPRESS (2000) and DeVa, in particular through previous research at the Centre for Software Reliability at City University, and present research at Queen Mary, University in London. The research has resulted in various papers, e.g. by Bertolino and Strigini (1996a, b, 1998), Neil et. al. (1996a, b, 1998, 2000, 2001), Fenton and Neil (1999) and Littlewood and Wright (1995, 1997).

This has also been the topic for research at the OECD Halden Reactor Project (HRP) (Dahll and Gran 2000). An attempt to combine the BBN technology with the rules of a standard for safety critical software, RTCA/DO-178B (1999) was done in an experimental project, (Gran et al. 2000), carried out by a consortium composed of HRP, and the Norwegian companies Kongsberg Defence & Aerospace AS (KDA) and Det Norske Veritas (DNV). Another sub-project of this long-term research is a co-operative project between HRP and VTT-Automation in Finland. One objective of this work is to investigate how the

network, representing the software safety guideline RTCA/DO-178B and different quality aspects, (Gran et al. 2000), can be merged with a network, developed by VTT (Helminen 2000), representing evidence from disparate operational environments. This paper describes experiences from both projects.

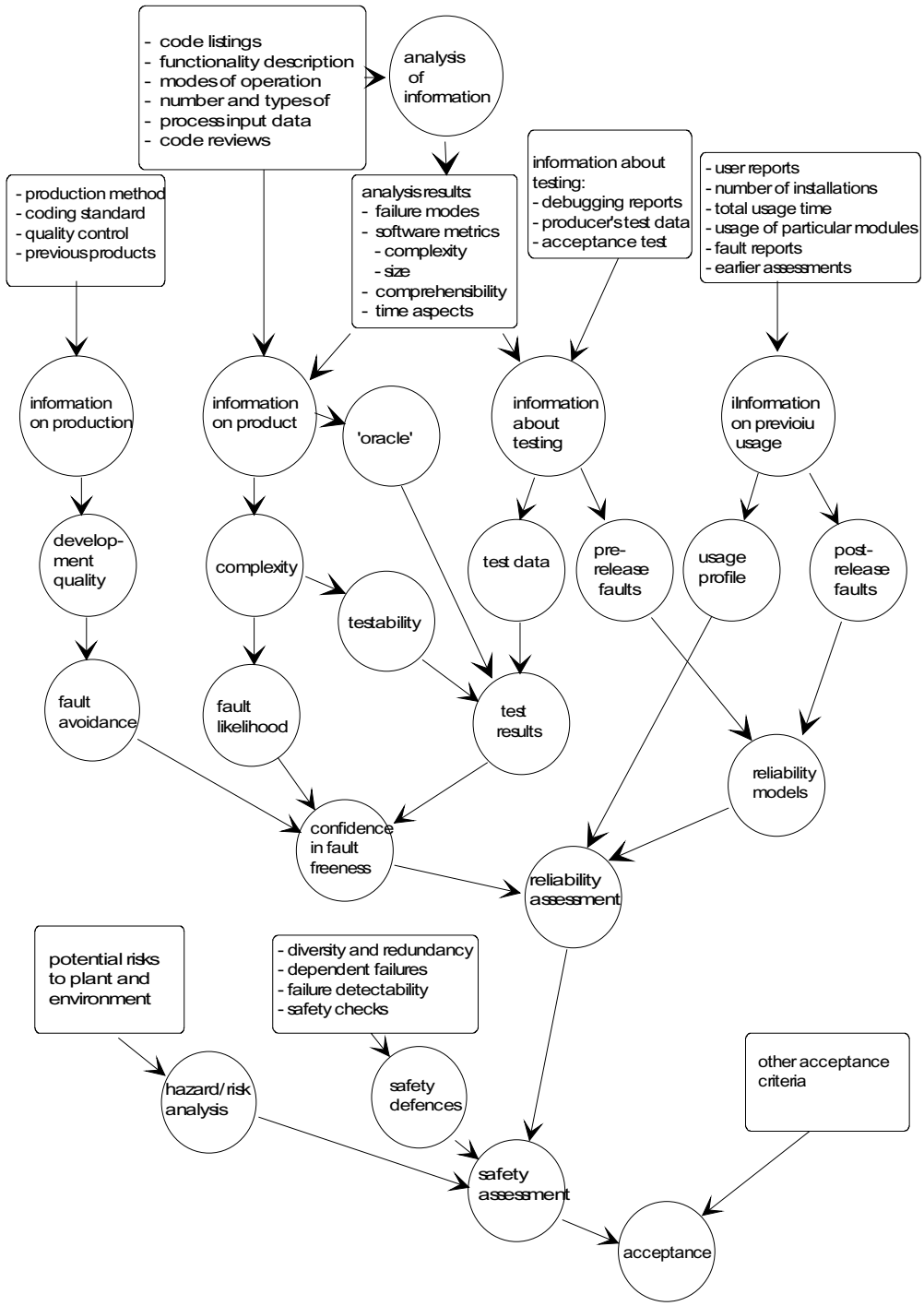

Figure 1. Influence graph of a safety acceptance and acceptance process of software

2 The BBN methodology

2.1 Background

Bayesian Belief Networks (BBN) methodology was introduced in the 1980s, and is in particular described in the book by Pearl (1988) and the paper by Lauritzen and Spiegelhalter (1988). In 1993 the tool HUGIN (Aldenryd, Jensen and Nielsen 1993, Jensen 1996) was introduced, which made BBN's feasible. The theory, however, is based on the Bayes Rule, discovered by Sir Thomas Bayes (1744-1809) which says for two variables X and Y that $P(X|Y)= P(Y|X)*P(X)/P(Y)$. By allowing $\{X_i\}$ be a complete set of mutually exclusive instances of X, this formula can be extended. A description of Bayesian interference, Bayesian Network methodology and theory for calculations on BBNs can also be found in the books by Gelman et al. (1995), Welch (1996), Cowell et al. (1999), the report by Pulkkinen and Holmberg (1997), and older references such as Whittaker (1990), and Speigelhalter et al. (1993).

A Bayesian Belief Net (BBN) is a connected and directed graph, consisting of a set of nodes and a set of directed arcs (or links) between them. Uncertain variables, both events and singular propositions, are associated to each node where the uncertainty is expressed by a probability density. The probability density expresses our confidence to the various variable outcomes, and depends conditionally on the status of the 'parent' nodes at the incoming edges. The nodes and associated variables can be classified into three groups:

- Target node(s) - the node(s) about which the objective of the network is to make an assessment. A typical example of such nodes is 'No faults in a program'.
- Intermediate nodes - nodes for which one have limited information, or only 'beliefs'. The associated variables are the hidden variables. Typical hidden variables represent quality aspects such as 'development quality', 'producer's reputation', or 'quality at a certain stage of the development' without discussing 'quality' in detail.
- Observable nodes - nodes which can be directly observed. Some examples are nodes representing observable properties about the system for evaluation: 'no failures during testing' and 'all quality requirements are fulfilled'.

Application of the BBN method consists of three tasks:

- construction of BBN topology,
- elicitation of probabilities to nodes and edges, and
- making computations.

2.2 The construction of BBN topology

The literature on BNN has mostly presented small 'complete' BBN's (Neil et al. 2000). The construction of small BBN can be made gradually. Information about the system is collected and expressed via the nodes. The nodes are connected to a directed graph that expresses the conditional relationship between the variables. The aim is to combine information in the net. One way is to start from a target node and draw edges to influencing nodes. To decide the direction of an edge, one follows the causal direction. However, this direction is not always obvious, in particular between nodes representing qualitative variables. In these cases the direction of the arrow often goes from higher abstraction to lower abstraction, or from the more general concept to the more detailed. A general interpretation of an arrow between two nodes A and B is that a 'belief' in A implies expectations on B. The practical procedure is to start with constructing a BBN, containing nodes representing high-level information. In figure 1 the influence graph of a 'safety

acceptance and acceptance process of the software' (Dahll 2001) is presented. It is not itself a BBN, but quite similar, so it is fairly straightforward to construct a high level BBN for safety assessment based on this (Dahll and Gran 2000), see the BBN shown in figure 2.
When building larger-scale BBN's this procedure is rather effort consuming. Neil, Fenton and Nielsen (2000) offers a solution based on building blocks (idioms), which serve solution patterns. These can then be combined into larger BBN's. This approach is for example applied in the SERENE project (1999), and has been applied to construct large-scale BBN's for predicting software safety. The use of idioms is also applied for the construction of the BBN's presented in the next chapter. However, the BBN's are not of such large-scale, so it is also possible to argue through the 'causal direction approach'.

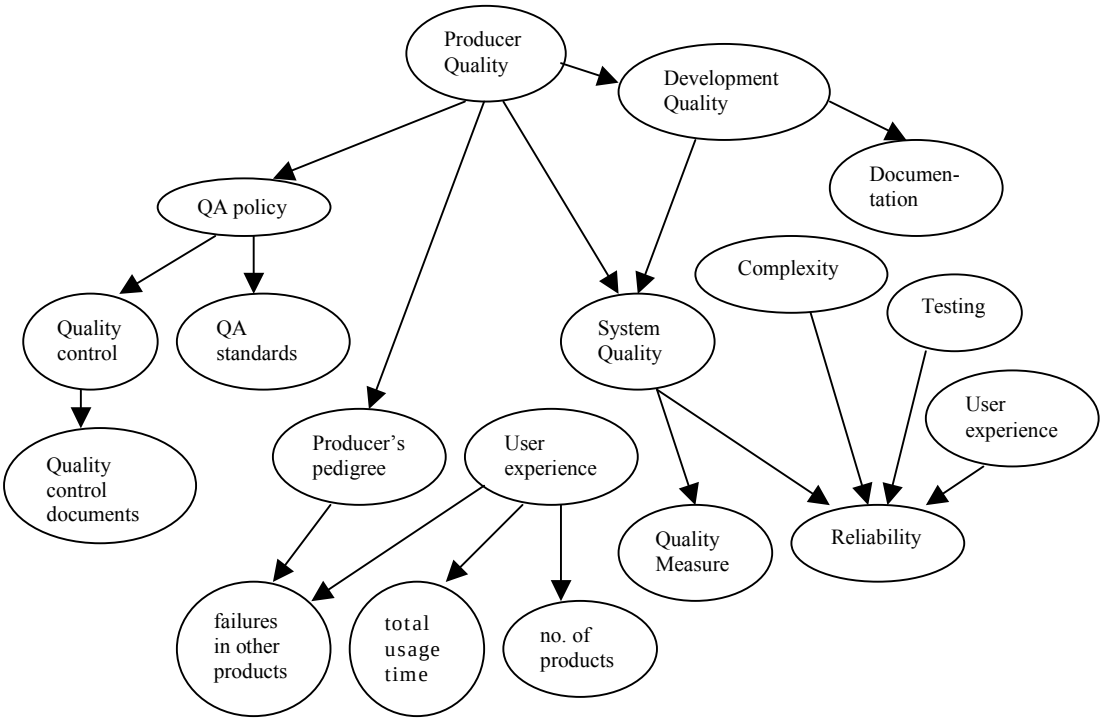

Figure 2. BBN for System Quality

2.3 **The elicitation of probabilities**

The second step is the elicitation of probability distribution functions (pdfs) to the nodes and edges. To begin with, one gives prior pdfs the top nodes, and conditional pdfs for the influences represented by the edges. These pdfs may be either continuous functions or they have a discrete form. The latter means that the ranges of the variables are divided into finite number of states. A typical example of a continuous conditional pdf is when the start node is reliability and the end node is an observable node: 'number of failures in test'. The variable corresponding to the start node could then be the probability (p) of failure per test run, and the observation could be: n failures in N test runs. The conditional pdf would then be given by the binomial distribution with parameters N and p.

The advantages of the pdfs in discrete form are that it becomes conceptually easier in an expert judgement to assign discrete values, and that it makes the computation simpler. The conditional probabilities for edges between discrete variables are given as conditional

probability tables between the states of the variables associated with the start node and the end node of the edge respectively. However, since many of the aspects to be considered are of qualitative nature and not directly measurable estimation may be difficult. This was observed for the co-operative project between HRP, KDA and DNV (Gran et. al. 2000), even if some of the project members can be considered as experts in their fields. It is therefore highly recommendable to make use of some expert judgment tools or expert judgment expertise. Another observation was that the establishment of the BBNs and prior conditional pdfs was rather time consuming.

The problem of defining the node probability tables is also addressed by Neil, Fenton and Nielsen (2000). They applied a 'divide and conquer' approach to build the BBN's. This manages the complexity of the BBN's, and thereby reduces the number of probability values to be addressed.

2.4 **Making computations**

Making computations with BBNs above a certain size and complexity is rather difficult by hand, but is rather easy by applying the latest computerised tools. At HRP the HUGIN tool (Aldenryd, Jensen and Nielsen 1993) has been used, and in the co-operative project with KDA and DNV both HUGIN and the SERENE methodology (1999) was applied.

The computation of our belief about a specific node (target node) is based on the rules for conditional probability calculations given by the Bayesian methodology. The procedure is to insert observations in the observable nodes, and then use the rules for probability calculation backward and forward along the edges, from the observable nodes, through the intermediate nodes to the target node, which again can be an intermediate node in a BBN at a higher level. Forward calculation is straight forward, while backward computation is more complicated (Spiegelhalter et. al. 1993). For details on computations see the references in the beginning of this chapter, and for good examples on making computations with BBN's see for example Pearl (1988) and Jensen (1996).

3 **BBN's based upon RTCA/DO-178B, the M-ADS project**

3.1 **Background**

The attempt to combine the Bayesian Belief Nets technology with the rules of a standard for safety critical software, RTCA/DO-178B (1999), hereafter referred to as DO-178B, was done in an experimental project carried out by a consortium composed of KDA, DNV, and the HRP. First of all the project goal was to evaluate the use of BBN for investigating the implementation of the DO-178B standard for software approval in the commercials world. To reach that objectives a computerized system for automated transmission of graphical position information from helicopters to land based control stations was selected and studied (Gran et. al. 2000). Please note that references to the system developed by KDA and conclusions here represent by no mean any official policy of KDA.

The project emphasized the practical evaluation of the BBN methodology by trying it out on a realistic test case: a computerized system for automated transmission of graphical position information from helicopters to land based control stations (M-ADS). The M-ADS airborne equipment was designed by KDA for installation in helicopter aircrafts. The system provides air traffic services with aircraft parameters upon request from the air traffic control where personnel will request positioning data. The work described below uses parts of the

M-ADS system to exemplify the software development process according to DO-178B standard.

3.2 **RTCA/DO-178B**

The purpose of the DO-178B standard (1999) is to provide guidelines for the production of safety critical software for airborne systems. This guideline was chosen for the study since the M-ADS system is applied in civil aviation, and was previously qualified on the basis of this standard. DO-178B discusses aspects of airworthiness certification pertaining to the production of software for airborne systems and equipment used in aircraft. To aid in understanding the certification process the system life cycle is briefly discussed to show relationship to the software life cycle process. It does not provide guidelines concerning the structure of the applicant's organization, relations to suppliers and personnel qualification criteria.

DO-178B defines a set of five software levels (A to E), based on the contribution from software to potential failure conditions as determined by the system safety assessment process. The main recommendations in DO-178B are given in a set of 10 tables, see description in table 1. Each table relates to a certain stage in the development and validation process, and contains a set of objectives. A difference between the DO-178B and e.g. IEC61508 (2000) is that most of the requirements are mandatory in the latter, while the requirements are guidelines in DO-178B (Neil and Fenton 1998).

Table 1: The stages in the development and validation process given by DO-178B

	Stage in the development and validation process
A1	Software planning process.
A2	Software development process.
A3	Verification of outputs of software requirements process.
A4	Verification of outputs of software design process.
A5	Verification of outputs of software coding & integration process.
A6	Testing of outputs of integration process.
A7	Verification of verification process results.
A8	Software configuration management process.
A9	Software quality assurance process.
A10	Certification liaison process.

3.3 **The Construction of BBN on the higher level**

The BBN for DO-178B was constructed at two levels. The higher level shows how nodes representing four quality aspects are combined with other nodes in the net, and leads to a node 'P(failed state)', representing the 'probability of finding the system in a failed state', see figure 3. The lower level shows how nodes representing the four quality aspects are related to objectives of D0-178B. The four quality aspects were:

- *Quality of the producer*. (Qproducer) This includes the reputation and experience of the producer, quality assurance policy, quality of staff etc.
- *Quality of the production process*. (Qprocess) A high quality implies that the system is developed according to guidelines for good software engineering, that all phases are well documented, and that the documentation shows that the system at all development phases possesses desirable quality attributes as completeness, consistency, traceability etc.
- *Quality of the product*. (Qproduct) This includes quality attributes for the final product, as reliability, simplicity, verifiability etc.

- *Quality of the analysis*. (Qanalysis) This includes all activities performed to validate the correctness of the system during all stages of the system development. Such activities may include model checking of the specifications, inspections and walkthroughs of the documentation, static analysis of code and testing of the system.

In addition to the quality nodes it includes the nodes 'problem complexity' and 'solution complexity'. The initial nodes or top nodes are the nodes: 'Qproducer' and 'problem complexity', where the latter is an attribute of the system to be developed, and can be assessed. It was assumed that the 'Qproducer' directly influences the 'Qprocess', ant that the 'solution complexity' was influenced by the initial 'problem complexity' and the 'Qprocess'. The same dependencies were assumed for the 'Qproduct'. Remark however, that this does not mean that the product quality depends only upon how difficult it is to fulfil the requirements (the complexity of the problem), and upon how good the development process handle complex systems. An assessment of the product will also be based upon assessments of the lower nets. The 'Qanalysis' was assumed to be influenced by the 'Qproducer', how well prepared the organization is to perform an analysis, and the 'solution complexity', how difficult it is to analyse. All these assumptions were in accordance with BBNs presented in the SERENE project (1999) and networks presented by HRP-project (Dahll and Gran 2000).

Finally it was assumed that a node representing the 'P(failed state)'is dependent on the factors 'Qanalysis', the 'Qproduct' and the 'solution complexity'. This node is not to be viewed as a failure rate representing a specific usage or safety function, but rather as a deterministic property of the system expressing fault content. One interpretation is the size of the inherent faults in the software. Assuming that no failures are found or modifications are made during later testing of the system, this true failure rate is not changed; only the confidence in the reliability, or freeness of faults, of the program is enhanced. Thereby it also offers a support in the assessment of the software.

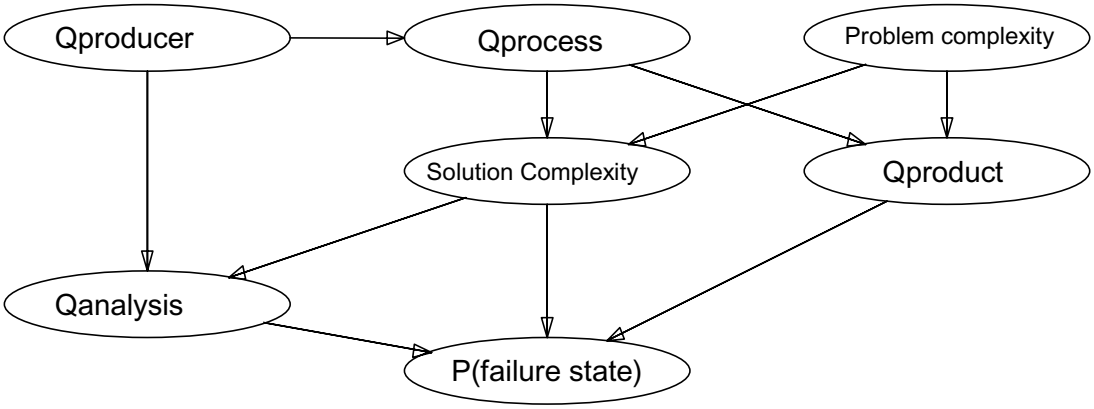

Figure 3. The upper network for DO-178B

3.4 **The Construction of BBN on the lower level**

We constructed a lower level BBN for each of the four quality aspects. This was done by first putting the quality aspects as top-nodes in the BBNs. Each top node was then linked to intermediate nodes representing the 10 lifecycle stages represented by the tables A1 to A10 of DO-178B. Each of these nodes were again linked to other intermediate nodes, representing the objectives of the tables.

The associating of the different objectives to the different quality aspects was done by a group of experts, consisting of experts related to the standard itself, development in accordance with the standard, and experts within safety assessment of critical systems. Each objective was identified to belong to one or more of the quality aspects. In addition a stage 'hmi-aspects' representing objectives related to human-machine interfaces was added.

The further step was to identify a list of questions to each objective. These questions were based on the understanding of the text in the main part of DO-178B, and then formulated so that the answer could be given by a 'yes' or a 'no'. Figure 4 presents an example representing the list of questions associated with two of the objectives for the software development process (A2) related to the quality of analyses. A list of the questions identified related to the 'quality of product' for (A2) is presented in table 2.

Table 2: The questions related to the lifecycle stage A2: software development process

Objective	Question:
sw req. data	Are all system functional requirements, safety requirements and auxiliary requirements specified in the software specification?
design descrip-tion	Are all tasks specified in the requirements also included in the design?
	Does the design adequately describe the information flow between components?
	Does the design address sequencing, concurrency and time related information?
	Does the design adequately describe the data structures and their properties?
	Is it a clear separation in the design between safety critical and not safety critical parts of the system?
	Are measures for fault tolerance, like diversity or redundancy designed into the system?
	Are control and data flow monitored when safety requirements dictate, e.g. through watchdog timers, reasonableness checks, input data checks etc.?
	Are the responses to failure conditions consistent with safety related requirements?

3.5 **The elicitation of probability tables**

The elicitation of conditional probability tables (cpt) to the nodes and edges was done as a brainstorming exercise by the expert group. In general, this means that for each node, the expert group had to assess two conditional probabilities of the type P(good measurement | good quality) and P(bad measurement | bad quality).

The first probability was relatively easy to assess. Based on general knowledge and experience in software development and evaluation, it was mostly done by ranking the importance of the different sub nodes, and giving them probabilities from a predefined such as {0.5, 0.7. 0.9, 0.95, 0.99}. The latter, however, became very difficult. Often, where the experts had stated that there was a dependency between good quality and a specific good measurement, they could not state the opposite effect. The approach of ranking the nodes had also restricted success. Even if some of the project members can be considered as experts within their fields, it is, however, highly recommendable to make use of some expert judgement tools or expert judgement expertise.

The establishment of the BBNs and prior pdfs was rather time consuming, and would be even more so for a system. However, the generation of the BBNs was related to DO-178B and on safety assessment in general, and not to the actual system. This implies that the BBNs have a general nature, and can be reused in many applications. They can also be gradually improved based on experience. Remark also that on the lower level, as illustrated in figure 4, all nodes had only one parent. This made the complexity of the BBN's manageable. In the case of nodes with more incoming edges, it would be a good solution to apply the approach suggested by Neil et al. (2000)

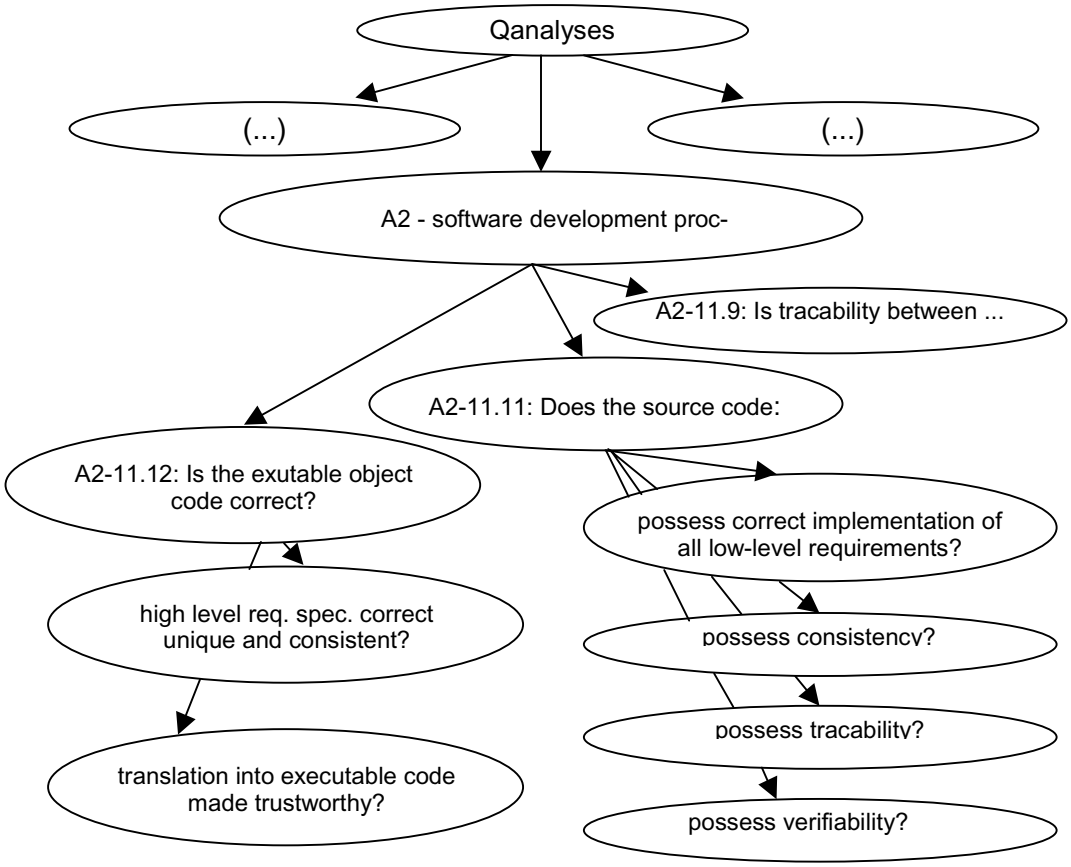

Figure 4. Example of a list of questions associated with two of the objectives for the software development process related to the quality of analyses

3.6 **Computations**

Finally all the BBNs were implemented, and all the conditional probability tables together with observations from the system development (KDA) were fed into the HUGIN and SERENE tools. This made it possible to make a variety of computations (Gran et al. 2000), with the aim to investigate different aspects of the methodology, such as:

- What is the effect of observations during only one lifecycle process?
- How does the result change by subsequent inclusion of observations from the lifecycle processes?
- How sensitive is the result to changes in individual observations?

Since the number of possible scenarios is exploding when one wants to explore both different sets of observations and prior cpts, a limited number of computations were made. However, an interesting observation was that we rapidly found surprising results that required further discussion and calculations. These results provided a list of topics for further research, both with respect to topological issues and with respect to different cases of observations. The topics are all issues addressed through ongoing research activities.

One example from this research is the importance of a good quality assurance of the observations entered into a BBN. The trigging event was the discovery of a wrong entered observation. Correcting this error demonstrated that one negative observations can have a

significant effect on a partial results. The evaluation also showed that one negative observation, or a set of a few negative observations is not enough to change the overall results.

3.7 Experiences from the M-ADS project

One objective of the project was to investigate the possibility to transfer the requirements of a software safety standard into a BBN. A review of various software standards revealed that they are built on the same basic framework, and follow the same principles, although they may differ in the aspect they put special emphasis on. The results and experiences with using the avionics standard DO-178B in the test case can therefore be seen as representative also for other software safety standards, including those used in the nuclear industry.

The BBN was constructed in two levels. The higher level based on the four qualities: Quality of the producer, Quality of the production, Quality of the product, and Quality of the analysis is general, and independent of the standard. The lower-level BBN reflects the recommendations of DO-178B, in the way that the four qualities were represented as top nodes in four sub-BBNs, whereas the objectives given in the tables in appendix A of DO-178B were represented as observable end nodes in the BBNs. These objectives were transformed into questions, and the answers to these were the observations used in the BBN computation. The construction of the BBN as it was done in this study is not unique, but should be considered as one possible solution in an experimental investigation.

The prior probability distributions and conditional belief distributions represent quantities that reflect a confidence in the standard, and can therefore not be generated on the basis of the standard itself. They were therefore determined in the 'expert judgement session'. The use of subjective numbers, and the numbers itself can be a separate topic for discussion. The objective of determining the numbers in this project was also not to find the best or most correct numbers, but to illustrate the approach. The conclusion from the project was that this way to construct the networks, combined with questionnaires, seems to be the promising mode of proceeding. Furthermore, the BBN methodology offers a systematic way to combine quantitative and qualitative evidences of relevance for the safety assessment of programmable systems

Another observation through the project was that the BBN methodology is not only applicable in the final assessment of a system, but could be used at all stages throughout the software lifecycle. The network could e.g. in this specific project be used to evaluate the difference between two different safety levels before any other measurements were collected. In this way it is possible to make assessments about the system even before it is designed or implemented. In such a way corrections to e.g. the development process can be made early in the project, in order to be able to reach specific objectives of the final product.

4 Extending the BBNs based upon RTCA/DO-178B

Within the nuclear field there is an increased focus on risk based regulation of nuclear power plants. This is in accordance with the new generic guideline for programmable safety related systems, IEC-61508 (2000), where probabilistic safety integrity levels are given as requirements for safe operation. Therefore, there is a need to establish methods to assess the reliability of programmable systems, including the software. One approach in this research is an on-going long-term joint research activity between HRP and VTT Automation (VTT) in Finland.

One objective of this co-operative project is to investigate how a network, representing the software safety guideline and different quality aspects, as described in the previous chapter, can be merged with a network, developed by VTT, representing evidence from disparate operational environments (Helminen 2000).

4.1 **The VTT Approach**

The main sources of reliability evidence in the case of safety critical systems considered in the VTT approach are depicted in fig 5, (Neil et. al 1996a). A similar version of this model has been presented by Stålhane et al. (1993). Part of the evidence may be directly measurable statistical evidence, such as the evidence obtained through operational experience and testing. Part of the evidence may be qualitative characterization of the system such as the design features and the development process of the system.

The qualitative characterization of the design features and the development process follows certain quality assurance and quality control principles, which are based on applicable standards. Running a good development process alone does not guarantee a more reliable product. However, the more strict standards the characterizations fulfil, combined with good testing results, the more confidence one will become in having a reliable system. The evidence based on qualitative characterization can be considered as soft evidence, while evidence obtained from operational experience and testing can be considered as hard evidence. The exploitation of soft evidence in the reliability analysis of software-based system requires extensive use of expert judgment making it quite an unforeseeable matter and therefore the VTT approach is mainly focused to the utilization of hard evidence.

The reliability of a software-based system is modelled as a failure probability parameter, which reflects the probability that the automation system does not operate when demanded. Information for the estimation of the failure probability parameter can be obtained from the disparate sources of hard and soft evidence. To obtain the best possible estimate for the failure probability parameter of the target system all evidence should to be combined.

The principle idea of the estimation method is to build a priori estimate for the failure probability parameter of software-based system using the soft and hard evidence obtained from the system development process, pre-testing and evaluating system design features while system is produced, but before it is deployed. The prior estimation is then updated to a posterior estimate using the hard evidence obtained from testing after the system is deployed and from operational experience while the system is operational. The difference between disparate evidence sources can be taken care in the structural modelling of the Bayesian Network model.

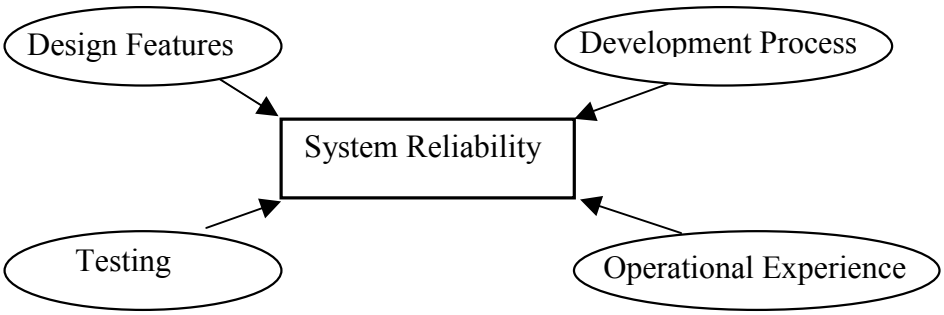

Figure 5. Main sources of reliability evidence in a case of safety critical system

To analyse the applicability of Bayesian Networks to the reliability estimation of software-based systems Bayesian Network models for safety critical systems are built. The different models are distinguished by the evidence, which is collected from different systems and from different operational profiles. The modelling is done using the WinBUGS program (Spiegelhalter et. al. 1996).

4.2 Evidence from one system with one operational profile

The Bayesian Network shown in the left part of figure 6 describes a system, for which the observed number of failures Y is binomial distributed with parameters N and P (Helminen 2000). Parameter N describes the number of demands in the single test cycle and parameter P is the random failure probability parameter. This model can be further extended to represent a system with several test cycles using the same operational profile (Helminen 2000).

To increase the flexibility of the model depicted in the left part, a logit-transformed P parameter Θ is included into the network, and the network becomes as shown to right in figure 6. The Bayesian network represented in model 1 can be used in the reliability estimation of a software-based system attached with binomial distributed hard evidence under unchanged operational profile.

The hard evidence obtained for the reliability estimation of software-based systems is usually obtained from both testing and operational experience. If the testing has been carried out under the same operational profile as the operational experience, the Bayesian Network becomes same as the network shown in figure 6.

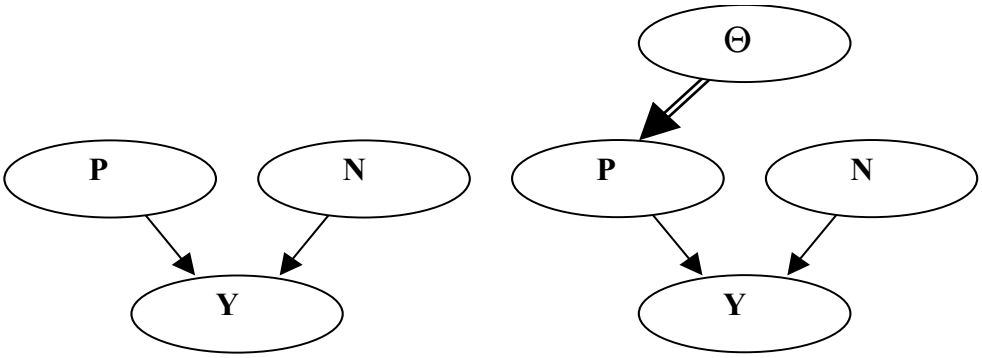

Figure 6. Bayesian network for one test cycle (right), and the Model 1 (left)

4.3 Evidence from one system with more operational profiles

Often the system is tested with different operational profiles under different operational environments. The results from applying the different operational profiles provide different failure probabilities for the same system. However, the failure probability from testing gives us some information about the failure probability of the system functioning in a different operational profile than where the testing was made. This evidence provided by testing is valuable and one should make a good use of it by taking into account the difference in the operational profiles when building the model.

Helminen (2000) solve the problem of different operational profiles by first connecting the binomial distributed evidence from different operational profiles to separate failure probability parameters. Then the logit-transformed failure probability parameters are connected to equal each other. The difference in the operational profile of two failure probability parameters is introduced in the model by adding a normal distributed random

term Ω*, with parameters μ* and σ*, to the logit-transformed failure probability parameter obtained from testing. The parameters of the normally distributed random term correspond to our belief of the difference between the two operational profiles. The Bayesian Network representing the case is illustrated in figure 7 when considering only the upper layer.

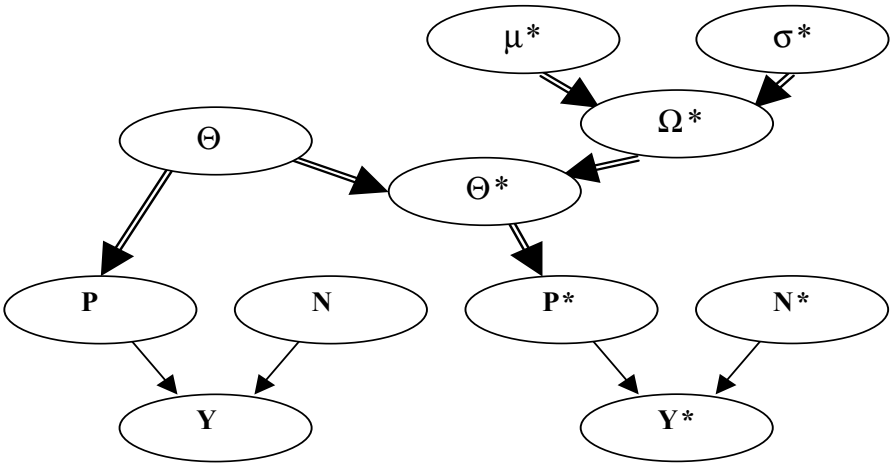

Figure 7. The model for two operational profiles

4.4 **Merging the HRP approach and the VTT approach**

The merging of the two approaches is based on the on the network presented in figure 3 and the network shown in figure 6 (Gran and Helminen 2001). The merged network is displayed in figure 8. The merging was done by replacing the node 'P(failure state)' by the node Θ_{priori}. This was done by transformation of the conditional probability tables for P(failure state) into continuous normal distributions.

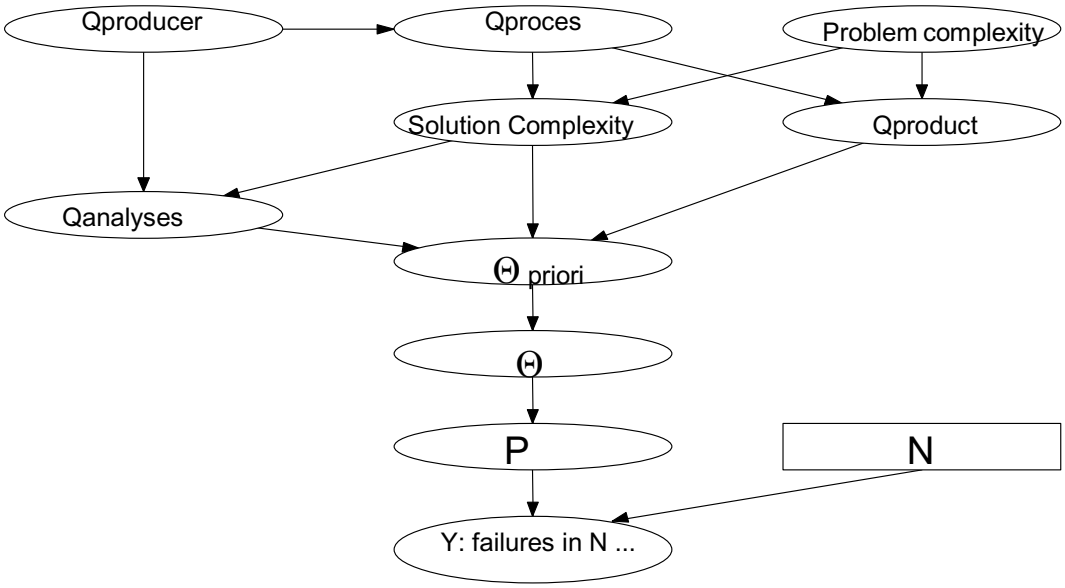

Figure 8. The merged network from the VTT and HRP approaches

Each of the quality aspect nodes was connected to quality aspects, as described in section 3.4. That allowed us to directly insert the observations from the M-ADS evaluation in the network, and for the merged network we performed calculations for two the case of no M-ADS observations and with the M-ADS observations, running from N=100 to N=1000000.

That allowed us to directly insert the observations from the M-ADS evaluation in the network, and for the merged network we performed calculations for two different scenarios:

- For were we have no M-ADS observations, but zero failures (Y=0), running from N=100 to N=1000000.
- For were we have the M-ADS observations, and zero failures (Y=0), running from N=100 to N=1000000.

For both scenarios the target was the node for the failure probability. In figure 9 both the median and the 97.5% percentile posterior distribution values for P on the logarithmic scale are shown. The values for N=1, are the values representing the prior distributions, i.e. before starting the testing (and observing Y=0). Remark that the curves for the 97.5% percentiles are somewhat "bumpy". This due to the fact, that the values are deduced from posterior histograms.

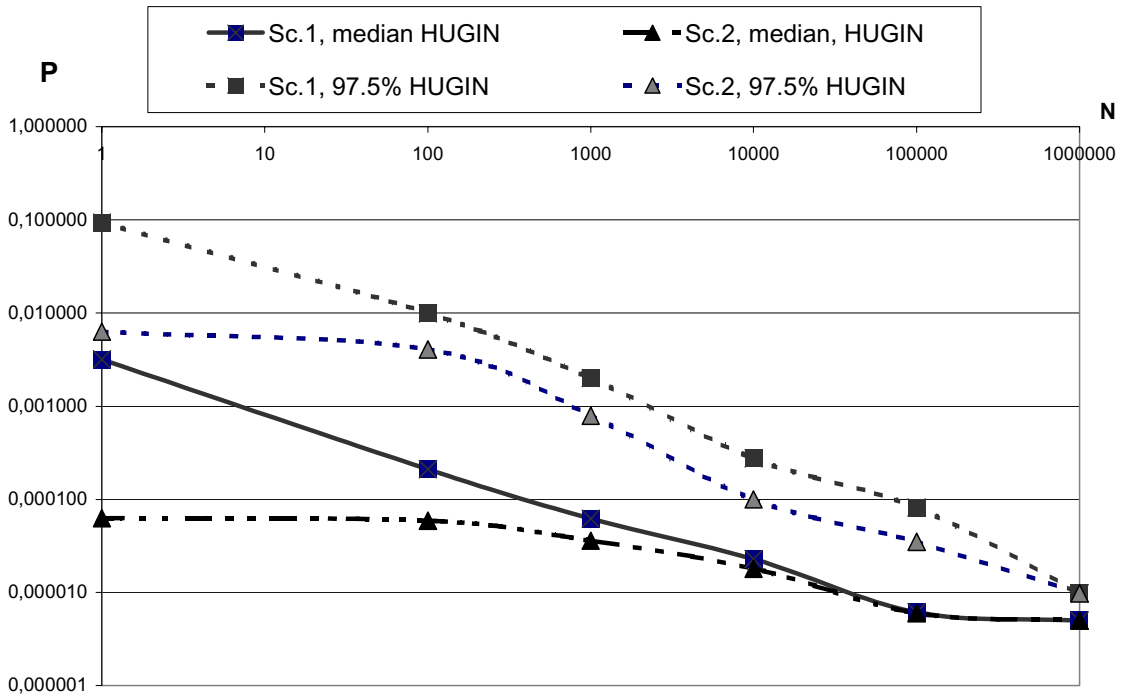

Figure 9. Median and 97.5% percentile posterior distribution values for P on the logarithmic scale, for the scenario of no observations and the scenario with the observations

4.5 **Experiences from the HRP-VTT project**

The main differences between the two studies lie in the difference of focus areas. The work by VTT mainly focuses to studying explicitly the influence of prior distributions to the reliability estimation and to the investigation of combining statistical evidence from disparate operational environments. The work by the HRP has mainly focused on how to

model a software safety guideline, DO-178B, and how to combine 'soft evidences' in the safety assessment of a programmable system. The key idea is to split the larger entities of soft evidence into smaller quantities. Another difference is the comprehensive usage of continuous distributions in the VTT work, which is somewhat a different approach than the approach used in the HRP study. This is however not discussed in this paper. The merged networks show how the two approaches can be merged. It gives an extended description of the quality aspects, originally modelled by the node Θ in the VTT approach, and it shows how different operational profiles, can be included in the approach from HRP.

5 Conclusions

The conclusion from the research presented in this paper is that the use of Bayesian Belief Networks for combining disparate sources of information in the safety assessment of software based systems, combined with questionnaires, offers a systematic and promising mode of proceeding.

The experiences with modelling the requirements of the avionics standard RTCA/DO-178B as a BBN, point in the direction that this approach can be transferred to the modelling of other software standards built on the same basic framework, and which follow the same principles. This holds even though they may differ in the aspect they put special emphasis on.

The BBN was constructed in two levels. The higher level was based on the four qualities: quality of the producer, quality of the production, quality of the product, and quality of the analysis is general, and independent of the standard. The BBN was based on the research discussed in chapter 2 but there is a need for validation and experimental investigation with respect to the network. Results obtained from calculations on the BBN, as presented in (Gran et. al. 2000), seems not only to be a consequence of the 'soft evidences' in the lower level networks or the 'hard evidences' in the testing, but also a result of the underlying topology. The lower-level BBN, reflecting the recommendations of RTCA/DO-178B, could also need a validation. A hypothesis is that a reallocation of objectives or questions only will give local (or partial) effects, and not changes in the overall assessment. A reason for this could be that there are a few 'soft evidences' and dependencies connecting this evidences which are more sensitive than the other. So fare, there has, however, not been possible to find such evidences.

Although the BBNs and results were based upon a real application, this approach has not been applied to a real development or assessment. A first try could be to apply the approach for decision support in the approval of safety critical programmable systems. Another try could be to apply the approach as decision support early in the development of a system, for example as an intelligent product support system, in order to point on where to set in the effort and thus being able to reach specific objectives of the final product.

6 Acknowledgement

I want to acknowledge the different persons that have taken part in this research. First of all Gustav Dahll, OECD Halden Reactor Project, which has taken active part in all the discussions behind this research. Then the rest of the project team that performed the 'M-ADS project': Siegfried Eisinger from Det Norske Veritas, and Eivind J. Lund, Jan Gerhard Norstrøm, Peter Strocka, and Britt J. Ystanes from Kongsberg Defence & Aerospace AS. I want to thank to KDA for allowing me to further work applying their observations. Also thanks to Atte Helminen and his colleagues at VTT Automation for bringing in new ideas

and co-operative work. Finally, acknowledge to Hugin Expert A/S for allowing me to use the HUGIN tool for my Ph.D.

References

Aldenryd, S.H., Jensen, K.B., and Nielsen, L.B., 1993, *Hugin Runtime for MS-Window,* Tool made by Hugin Expert a/s, Aalborg, <http://www.hugin.dk>

Bertolino, A., and Strigini, L., 1996a, Predicting Software Reliability from Testing Taking into Account Other Knowledge about a Program. *Proceedings 9th International Software Quality Week* (Software Research Institute, San Francisco).

Bertolino, A., and Strigini L., 1996b, Acceptance Criteria for Critical Software Based on Testability Estimates and Test Results. *Proceedings SAFECOMP96, 15th International Conference on Computer Safety, Reliability and Security*, Schoitsch (ed) (Springer-Verlag), pp 83-94.

Bertolino, A., and Strigini, L., 1998, Assessing the risk due to software design faults: estimates of failure rate vs. evidence of perfection, *Software Testing, Verification and Reliability*, **8**(3), 155-166.

CORAS, 2000, A Platform for Risk Analysis of Security Critical Systems. IST project nr.2000-25031, <http://www.nr.no/coras/>.

Cowell, R.G., Dawid, A.P., Lauritzen, S.L., and Spiegelhalter, D.J., 1999, *Probabilistic Networks and Expert Systems* (Springer-Verlag).

Cudleigh, M., and Catmur, J., 1992, Safety Assessment of Computer Systems using HazOp and Audit Techniques, *Safety of Computer Systems SAFECOMP'92*, Frey (ed) (Pergamon Press).

Dahll, G., 1997, Safety Assessment of Software Based Systems. *SAFECOMP'97*, Daniel (ed) (Springer-Verlag), pp. 14-24.

Dahll, G., 2001, Combining Disparate Sources of Information in the Safety Assessment of Software Based Systems, submitted to *Special Issue of Nuclear Engineering and Design*.

Dahll, G., and Gran, B.A., 2000, The Use of Bayesian Belief Nets in Safety Assessment of Software Based Systems. *Special Issues of International Journal on Intelligent Information Systems at FLINS'98, Int. J. General Systems,* **24** (2), 205-229.

Fenton, N., and Neil, M., 1999, A Critique of Software Defect Prediction Models, *IEEE Transactions on Software Engineering*, **25** (5), 675-689.

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B., 1995, *Bayesian Data Analysis*, (Chapman & Hall, London), 1-526.

Gran, B.A., and Thunem, H, 1998, EISTRAM - Experimental Investigation of the PIE-technique. *Safety and Reliability*, Lydersen, Hansen and Sandtorv (eds), (Balkema, Rotterdam), pp 409-416.

Gran, B.A., Dahll, G., Eisinger, S., Lund, E., Norstrøm, J., Strocka, P., and Ystanes, B., 2000, Estimating Dependability of Programmable Systems Using BBNs. *Computer Safety, Reliability and Security, Proceedings from Safecomp 2000,* (*Lecture Notes in Computer Science 1943*), Koornneef and van der Meulen (Eds) (Springer), pp. 309-320.

Gran, B.A., and Helminen, A., 2001, A Bayesian Belief Network for Reliability Assessment, paper to be presented at Safecomp 2001, 25–28 September 2001, Budapest, Hungary

Helminen, A., 2000, *Reliability Estimation of Software-based Digital Systems Using Bayesian Networks*, (Helsinki University of Technology, Espoo), pp. 1-50.

IEC publication 61508, 2000, Functional safety of electrical/electronic/programmable electronic safety-related systems.

IMPRESS, 1999, Improving the software process using Bayesian nets. EPSRC project nr. GR/L06683, <http://www.csr.city.ac.uk/csr_city/projects/impress.html>.

Jensen, F., 1996, *An Introduction to Bayesian Networks*, (UCL Press, University College London).

Lauritzen, S.L., and Spiegelhalter, D.J., 1988, Local computations with probabilities on graphical structures and their application to expert systems (with discussions), *Journal of the Royal Statistical Society*, Series B **50** (2), 157-224.

Leveson, N.G., 1995, *Safeware – System Safety and Computers*, (Addison-Wesley publishing company).

Littlewood, B., and Wright, D., 1995, A Bayesian Model that Combines Disparate Evidence for the Quantitative Assessment of System Dependability. *Proceedings SAFECOMP'95*, Rabe (ed), (Springer-Verlag), pp. 173-188.

Littlewood, B., and Wright, D., 1997, Some conservative stopping rules for the operational testing of safety-critical software, *IEEE Transactions of Software Engineering*, **23**(11), pp 673-683.

Neil, M., Littlewod, B., and Fenton, N., 1996a, Applying Bayesian Belief Nets to Systems Dependability Assessment, *Proceedings of 4th Safety Critical Systems Symposium*, (Springer-Verlag), pp. 71-93.

Neil, M., and Fenton, N., 1996b, Predicting Software Quality using Bayesian Belief Networks, *Proceedings of 21st. Annual Software Engineering Workshop*, (NASA Goddard Space Flight Centre), pp. 217-230.

Neil, M., and Fenton, N., 1998, A strategy for improving safety related software engineering standards, *IEEE Trans. on SW Eng*., **24** (11).

Neil, M., Fenton, N., and Nielsen, L., 2000, Building large-scale Bayesian Networks, *The Knowledge Engineering Review*, **15** (3), 257-284.

Neil, M., Fenton, N., Forey, S., and Harris, R., 2001, Using Bayesian Belief Networks to Predict the Reliability of Military Vehicles, *IEEE Computing and Control Engineering*, **12** (1), 11-20.

Pearl, J., 1988, *Probabilistic Reasoning in Intelligent Systems: Networks forPlausible Inference*, (Morgan Kaufman).

Pulkkinen, U., and Holmberg, J., 1997, A Method for Using Expert Judgement in PSA, (Finnish Centre for Radiation and Nuclear Safety, Helsinki), pp. 1-32.

RTCA/DO-178B, 1999, Software Considerations in Airborne Systems and Equipment Certifications (Guideline).

SERENE, 1999, Safety and Risk Evaluation using Bayesian Nets. ESPRIT Framework IV nr. 22187, <http://www.hugin.dk/serene/>.

Spiegelhalter, D. J., Dawid, A. P., Lauritzen, S. L., and Cowell, R. G., 1993, Bayesian Analysis in Expert Systems, *Statistical Science*, **8**(3), pp 219-283

Spiegelhalter, D., Thomas, A., Best. N., and Gilks. W., 1996, *BUGS 0.5 Bayesian Inference Using Gibbs Sampling Manual* (version ii), (MRC Biostatistic Unit, Cambridge), pp.1-59.

Stålhane, T., Meulen, M.J.P van der, Cole, 1993, Reliability Assessment of PES, using Subjective and Objective Categorcial Data. Presented at workshop on production control in the process industry, 29-31 March, Dusseldorf, Germany.

Stålhane, T., 1997, Safety in Software-Intensive Systems. Presented at the Eighth European Workshop on Dependable Computing, EWDC-8, Gothenburg, Sweden.

Voas, J. M., Michael, C.C., and Miller, K. W., 1993, Confidently Assessing a Zero Probability of Software Failure, *Proceedings of the 12th Int'l. Conference. on Computer Safety, Reliability, and Security*, Poznan and Poland (eds), (Springer-Verlag), pp 197-206.

Welsh, A. H., 1996, *Aspects of Statistical Inference*, (Wiley & Sons).

Whittaker, J., 1990, *Graphical Models in Applied Multivariate Statistics*, (J. Wiley & Sons).

Xie, M., 1991, *Software Reliability Modelling*, (World Scientific Publishing Co. Pte. Ltd).

IV

Applying Bayesian belief net in software safety assessment on a real, safety related programmable system.

In *Safety & Reliability, Towards a safer world*. Zio, E., Demichela, M., and Piccinini, N. (Eds), Politecnico di Torino, Torino, pp. 1045-1052, 2001.

APPLYING BAYESIAN BELIEF NET IN SOFTWARE SAFETY ASSESSMENT ON A REAL, SAFETY RELATED PROGRAMMABLE SYSTEM

Bjørn Axel Gran

OECD Halden Reactor Project,
Institutt for energiteknikk,
P.O.Box 173, N-1751 Halden, Norway
<bjorn.axel.gran@hrp.no>

ABSTRACT: This paper describes an attempt to combine the Bayesian Belief Nets technology with the rules of a standard for safety critical software, DO-178B, together with the evaluation of some of the results obtained by applying the approach on a real, safety related, programmable system. The research was done in an experimental project carried out by a consortium composed of Kongsberg Defence & Aerospace AS, Det Norske Veritas, and the OECD Halden Reactor Project. First of all the project goal was to evaluate the use of BBN to investigate the implementation of the DO-178B standard for software approval in the commercials world. To reach that objectives a computerized system for atomised transmission of graphical position information from helicopters to land based control stations was selected and studied. This paper describes some of the findings from the project, and discusses some of the results that were pinpointed as interesting, strange or counter-intuitive.

1 INTRODUCTION

There has been an increasing use of programmable digital equipment in safety critical systems. A problem in this respect has been the licensing of these systems, in particular of the embedded software. In practice the assessment of safety critical software is a matter of consensus among experts based on judgement of a variety of evidences. To combine evidences from different information sources the use of Bayesian Belief Nets (BBN) has been proposed in quantitative assessment of the confidence in a programmable system.

This methodology has mainly been developed and applied in the AI society. More recently, however, it has also been applied to software safety assessment. Work in this area has been performed in two ESPRIT projects: SERENE and DeVa, and at the Centre for Software Reliability at City University in London and at the OECD Halden Reactor Project [1]. An attempt to combine the BBN technology with the rules of a standard for safety critical software, DO-178B [2], was done in an experimental project, [3], carried out by a consortium composed of the OECD Halden Reactor Project, Kongsberg Defence & Aerospace AS and Det Norske Veritas. This paper shortly describes the test case (M-ADS) evaluated by the consortium, and puts emphasis on the results that were pinpointed as interesting, strange or counter-intuitive, and thereby needed an explanation.

2 THE BBN METHODOLOGY

A Bayesian Belief Net (BBN) is a connected and directed graph, consisting of a set of nodes and a set of directed arcs between them. Uncertain variables, both events and singular propositions are associated to each node where the uncertainty is expressed by a probability density. The probability density expresses our confidence to the various variable outcomes, and depends conditionally on the status of the "parent" nodes at the incoming edges. Some

nodes are denoted as “observables”. They represent the different observable properties about the system for evaluation. The computation of our belief about a specific node (target node) is based on the rules for conditional probability calculations backward and forward along the edges, from the observable nodes, through the intermediate nodes to the target node [4], [5].

The construction of the BBN is normally made gradually. Information about the system is collected and expressed via the nodes. The nodes are connected together to a directed graph that expresses the conditional relationship between the variables. The aim is to combine information in the net. One way is to start from a target node and draw edges to influencing nodes. To decide the direction of an edge, one follows the causal direction. However, this direction is not always obvious, in particular between nodes representing qualitative variables. In these cases the direction of the arrow often goes from higher abstraction to lower abstraction, or from the more general concept to the more detailed. For computations of a realistic BBN computer tools are necessary. We have applied both the SERENE methodology [6] and the HUGIN tool [7].

3 THE TEST CASE

3.1 M-ADS

The project emphasized the practical evaluation of the BBN methodology by trying it out on a realistic test case: a computerized system for atomised transmission of graphical position information from helicopters to land based control stations (M-ADS). The M-ADS airborne equipment was designed by Kongsberg Defence & Aerospace AS for installation in helicopter aircraft. The system provides air traffic services with aircraft parameters upon request from the air traffic control where personnel will request positioning data. The M-ADS system is designed to automatically transmit flight information via data link to one or more requesting air control centers. M-ADS uses existing avionics on board the aircraft to provide aircraft position, speed and additional optional data. Most important are the aircraft position, position accuracy, altitude and time stamp for the data validity. The work described below uses parts of the M-ADS system to exemplify the software development process according to DO-178B standard.

3.2 DO-178B

The purpose of the DO-178B standard [2] is to provide a required guideline for the production of safety critical software for airborne systems. This guideline was chosen for the study since the M-ADS system is applied in civil aviation, and was previously qualified on the basis of this standard. DO-178B discusses aspects of airworthiness certification that pertain to the production of software for airborne systems and equipment used in aircraft. To aid in understanding the certification process the system life cycle is briefly discussed to show relationship to the software life cycle process. It does not provide guidelines concerning the structure of the applicant’s organization, relations to suppliers and personnel qualification criteria.

The main recommendations in DO-178B are given in a set of 10 tables, see Table 3.1. Each table relates to a certain stage in the development and validation process, and contains a set of objectives. A difference between the DO-178B and IEC 61508 [8] is that most of the requirements are mandatory in IEC 61508, while the requirements are guidelines in DO-178B, [9].

Table 3.1: The main recommendations in DO-178B

	Stage in the development and validation process
A1	Software planning process.
A2	Software development process.
A3	Verification of outputs of software requirements process.
A4	Verification of outputs of software design process.
A5	Verification of outputs of software coding & integration process.
A6	Testing of outputs of integration process.
A7	Verification of verification process results.
A8	Software configuration management process.
A9	Software quality assurance process.
A10	Certification liaison process.

3.3 **The M-ADS Evaluation**

The M-ADS evaluation consisted of several tasks. The first was to construct BBNs on the basis of DO-178B. The BBN was constructed in two levels. The higher level shows how nodes representing four quality aspects are combined with other nodes in the net, and leads to a node "P(failed state)", representing the "probability of finding the system in a failed state", see Figure 3.1. The four quality aspects were:

- *Quality of the producer*. (Qproducer) This includes the reputation and experience of the producer, quality assurance policy, quality of staff etc.
- *Quality of the production process*. (Qprocess) A high quality implies that the system is developed according to guidelines for good software engineering, that all phases are well documented, and that the documentation shows that the system at all development phases possesses desirable quality attributes as completeness, consistency, traceability etc.
- *Quality of the product*. (Qproduct) This includes quality attributes for the final product, as reliability, simplicity, verifiability etc.
- *Quality of the analysis*. (Qanalysis) This includes all activities performed to validate the correctness of the system during all stages of the system development. Such activities may include model checking of the specifications, inspections and walkthroughs of the documentation, static analysis of code and testing of the system.

The lower level BBNs were constructed by identifying the quality aspects with top-nodes in four BBNs. Each top node was linked to intermediate nodes representing the 10 lifecycle processes represented by the tables A1 to A10 of DO-178B. Each of these nodes was again linked to other intermediate nodes, representing the objectives of the tables. The further step was to identify a list of questions to each objective, see example in Figure 3.2. These questions were based on the understanding of the text in the main part of DO-178B, and then in general formulated so that the answer could be given by a "yes" or a "no".

The elicitation of conditional probability tables (cpt) to the nodes and edges was done as "brainstorming" exercises by all project participants, based on general knowledge and experience in software development and evaluation.

Finally all this information together with observations from the system development (KDA) were fed into the HUGIN and SERENE tools, to make a variety of computations, with the aim to investigate different aspects of the methodology [3]. What is the effect of observations during only one lifecycle process? How does the result change by subsequent inclusion of observations from the lifecycle processes? How sensitive is the result to changes in individual observations?

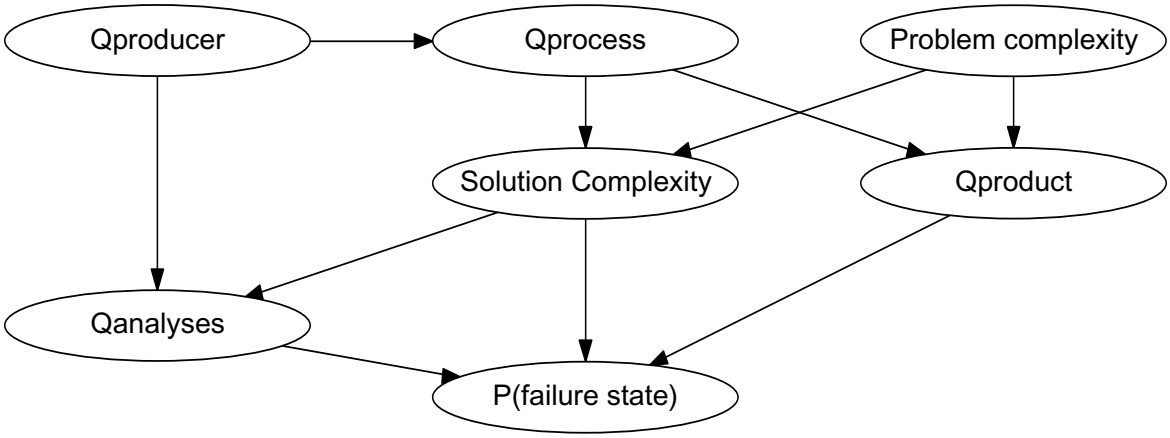

Fig. 3.1: The upper network

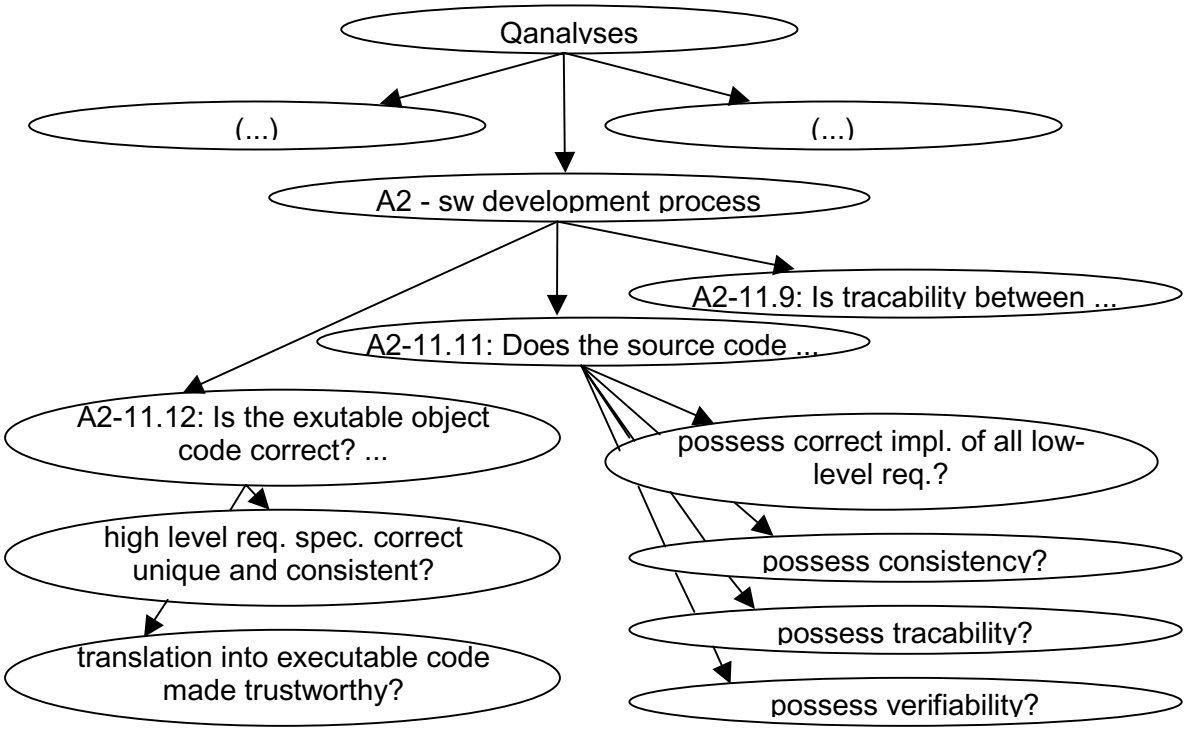

Fig. 3.2: Example of a list of questions associated with a objective for the software development process

4 RESULTS

4.1 What If More Observations Were Negative?

4.1.1 *The Partial Scenarios Results*

The effect of the observations during only one stage in the development and validation process showed with respect to the "Qproducer" that the effects were approximately the same for all the processes. With respect to the "Qprocess" the processes with largest effect were

“verification of outputs of sw requirements process (A3)”, “sw configuration management process (A8)” and “certification liaison process (A10)”, while an additional class “other aspects”, including aspects such as e.g. human machine interfaces, had lowest effect. With respect to the “Qanalysis” the process “other aspects” had largest effect, but also all the other processes had a large effect. With respect to the “Qproduct” the processes with largest effect were identified as the “verification of verification process results (A7)” and “other aspects”. Quite low effect was observed for the processes: “verification of outputs of sw design processes (A4)” and the process A10.

4.1.2 *The effect of one negative and one “not positive” observations*

One particular notice about the applied observations from KDA was that one observation, with respect to process A4, was given the value 0. This value corresponds to a negative answer to the question “Is the software partitioning integrity confirmed?” However, whether this answer was meant to be negative; i.e. that this question is of importance to the reliability of the product, or if this question was ranked as irrelevant, was not further discussed. In the latter case it would have been better not to give any value to this observable node at all. This is equivalent to cutting the edge to this node.

A further walk-through of the observations also identified that 6 questions, which belong to two or more of the quality aspect networks, were given different observations in the different sub-networks. Of these 6 questions, 1 belonged to the process A10, and was given a very low score for the “Qproduct”. (For the other divergences, the differences were smaller.)

The result of correcting these faults was that the “surprising low effect for A4 and A10” disappeared. And, the processes with low effect were now observed to be A1 and A8. These were both also identified as contributors to low effect for the other quality aspects.

On the other hand, if one assumes that the questions should have been non-identical, and that the observations on these in fact were negative or low as entered, then we have identified a case where only two negative observations can lead to negative significant changes in the partial scenarios.

4.1.3 *How to select negative observations*

The latter result is related to the fact that the observations applied in the project were in general positive. An open question was therefore: what would be the result if more observations were negative? In particular, we were interested in the overall results after entering observations in all phases, and we wanted to search for a set of “negative observations” that belonged to all or more phases. The reason for the latter is that it is very little realistic to have good observation within 9 phases, and negative observations within the others. More realistic is that the negative observations are distributed over all phases.

An attempt to find a such set of observations (out of a total of 71 observations) was to look into the set of observations (19 observations) that is related to two or three processes. These 19 observations can be divided into 5 groups as shown in Table 4.1.

Table 4.1: The 5 groups of observations related to more quality aspects

Group	Related to quality aspect:	Processes:
1	”QProduct”, “Qananlysis”	A4, A5, A7, other
2	”QProcess”, “Qananlysis”	A1, A2, A6
3	”QProduct”, “QProcess”	A5, A6
4	”QProduct”, “QProcess”, “QProducer”	A9, A10
5	”QProduct”, “QProcess”, “Qananlysis”	A3, A5

4.1.4 *The effect of some negative observations*

By entering negative observations to the questions related to the three groups we observed the effects as shown in Table 4.2. Remark that all the other observations are held positive, and the effect of change is observed related to "as observed by KDA", that is more or less all positive. As shown in the table, we see that there was only a significant effect on the "Qproducer". That means that we by entering negative observations on the two questions related to processes A9 and A10, we achieve a lower confidence in good quality of the producer.

Table 4.2: The effect of negative observations related to the questions from the groups 1-5.

Gr.	Observed Effect
1	Minor effect to "QProduct" and "Qananlysis"
2	Minor effect on "QProcess", no effect on "Qananlysis"
3	Minor effect on "QProcess", no effect on "QProduct"
4	No effect on "QProduct", minor effect on "QProcess", but significant effect on "QProducer"
5	No effect on "QProduct", minor effect on "QProcess", and no effect on "Qananlysis"

4.1.5 *The effect of 19 negative observations*

Based on the results presented above, the next scenario was to enter a negative observation on all the questions related to all the groups presented in Table 4.2. This had a significant effect on all the quality aspects, and also the node "P(failure state)"as shown in Table 4.3 and Table 4.4. An issue for further investigation is to look the combinations of these 19 to see how the results turn from positive towards negative.

Table 4.3: The effect of 19 negative observations on the node QProducer.

Scenario	good=5	4	3	2	bad=1
KDA original[1]	0.145	0.782	0.070	8E-6	6E-8
KDA corrected[2]	0.184	0.804	0.011	1E-6	1E-8
19 negative	0.018	0.359	0.621	0.01	1E-6

(1) as presented in 4.1.1, (2) after corrections as described in 4.1.2

Table 4.4: The effect of 19 negative obs. on the nodes Qprocess, Qproduct and Qanalysis.

Scenario	Process		Product		Analysis	
	good	bad	good	bad	good	bad
KDA original[1]	1.0	4E-7	1.0	3E-9	1.0	1E-9
KDA corrected[2]	1.0	2E-7	1.0	1E-10	1.0	1E-9
19 negative	0.039	0.961	0.117	0.883	0.993	0.007

(1) as presented in 4.1.1, (2) after corrections as described in 4.1.2

4.1.6 *Discussion of the results*

One other observation from the results from the incremental scenario was that they reached stable maxima very fast. This indicates that the activities in the later stages in the development and validation process have little effect. Similarly, the partial results were almost as good as complete results. These results were not as expected.

One possible explanation is that a good score in one table is an indication of high quality during all phases, so that there also should be high scores in other tables. Another explanation is that 19 questions are repeated in two or three tables. However, these two explanations are not necessarily different. The latter can be a way to use the BBN topology to express the

first explanation; i.e. that certain types of observations are relevant for several of the development phases associated with the tables.

4.2 **The difference in partial results for A4 and A5**

A third observation from the project was obtained by comparing the partial results from the lifecycle processes "verification of outputs of software design processes" (A4) and "verification of outputs of software coding and integration process" (A5). The "good" score for these were the same for the quality aspects "producer" and "analysis", but A5 scored better on "process" and in particular on "product". To explain this difference, the differences in BBN topology, in the cpts, and in the observations are investigated.

The investigation showed related to the partial results for the processes A4 and A5, that we have the effects of both neutralizing, conformity, enlargement and the effect of the observations alone, see details in Table 4.5.

Table 4.5: The partial results for the processes A4 and A5.

Quality aspect	Observed difference in A4 and A5	Difference in topology	Difference in observations	Effect of observations and topology (A4 vs. A5)
Producer	"A4 = A5"	differences in the cpts	different observations	neutralize each other
Analyses	"A4 = A5"	different number of questions	no large differences	conformity, i.e. "A4 = A5"
Process	"A4 ≠ A5"	differences in the cpts and topology	different observations	topology and observations pull in same direction (enlargement)
Product	"A4 ≠≠ A5	different number of questions	different observations	although a different number of questions, the observations alone give the difference

4.3 **Consequences of the "A4 vs. A5" evaluation**

The comparison of the results for A4 and A5 can also explain the difficulty of finding a subset of observations that turns the results negative. Accordingly we shall expect problems with finding a subset of positive observations leading to stable maxims.

These results also indicate the problems of performing a verification of a Bayesian Belief Network. The reason is that two different groups of experts can come up with two different BBNs. If one then enters somehow different observations into these networks, there is a good chance of observing the same results for the target nodes. On the other hand, these results also point in the direction that two different BBNs should be based up on the same observations. This again is an argument in favour of the BBN-construction process applied: each objective in the guideline associated to a list of questions.

5 **DISCUSSION**

The research conducted addresses some of the observations pinpointed as interesting, strange or counter-intuitive in the project on combining the Bayesian Belief Nets technology with the rules of a standard for safety critical software, DO-178B for a real, safety related, programmable system. One results is the importance of a good quality assurance of the observations entered into a BBN. One the other hand, it also demonstrates that one negative observations can have a significant effect on a partial results. The evaluation has also showed that

one negative observation, or a set of a few negative observations is not enough to change the overall results.

The results also show that there can be a rapid change in the overall results, given a specific order of turning the observations. This work indicates that this change takes place somewhere in the "middle" of "negative observations on a few repeated questions" and "negative observations on all repeated questions". A further evaluation can give more specific results. However, the evaluation has also showed that there is an effect of the combination of topology, cpts and observations. A pinpointed set of observations could therefore change by a change in the topology or a conditional probability table.

Finally the evaluation points on some of the problems that one will be faced with wanting to perform a validation or verification of the BBN. One hypothesis is that the use of questionnaires can be a vital point. This is also a topic for further investigation.

6 ACKNOWLEDGEMENT

I want to acknowledge the rest of the project team that performed the "M-ADS project": Siegfried Eisinger from Det Norske Veritas, Gustav Dahll from the OECD Halden Reactor Project, and Eivind J. Lund, Jan Gerhard Norstrøm, Peter Strocka, and Britt J. Ystanes from Kongsberg Defence & Aerospace AS. In particular acknowledgement to KDA for allowing me to further work applying their observations. Also note that this paper represents by no mean any official policy of KDA.I also want to acknowledge Hugin Expert A/S for allowing Bjørn Axel Gran the use of the HUGIN tool for his Ph.D. work.

REFERENCES

[1] Dahll, G. & Gran, B.A.: "The Use of Bayesian Belief Nets in Safety Assessment of Software Based Systems". In Special Issues of International Journal on Intelligent Information Systems at FLINS'98, the special issue of *Int. J. General Systems* v24, no2 (2000).

[2] RTCA/DO-178B: "Software Considerations in Airborne Systems and Equipment Certifications", (1992).

[3] Gran, B.A. et al. (Dahll, Eisinger, Lund, Norstrøm, Strocka, Ystanes) "Estimating Dependability of Programmable Systems Using BBNs". Printed in Koornneef, van der Meulen (Ed.): "Computer Safety, Reliability and Security", Proceedings from Safecomp 2000, Springer, (Lecture Notes in Computer Science 1943), pp 309-320 (2000).

[4] Casella, G., Berger, R. L.: "Statistical Inference", Wadsworth & Brooks/Cole Advanced Books & Software (1990).

[5] Spiegelhalter, D.J., Dawid, A.P., Lauritzen, S.L., and Cowell, R.G.: "Bayesian Analysis in Expert Systems", Statistical Science, Vol. 8-3 (1999).

[6] SERENE: "Safety and Risk Evaluation using Bayesian Nets". ESPRIT Framework IV nr. 22187, (1999). (http://www.hugin.dk/serene/).

[7] HUGIN: Tool made by Hugin Expert a/s, Aalborg, Denmark (http://www.hugin.dk).

[8] IEC publication 61508: "Functional safety of electrical/electronic/programmable electronic safety-related systems", version 4.0 (1997).

[9] Fenton, N., Neil, M.: "A Strategy for Improving Safety Related Software Engineering Standards". *IEEE Transactions on Software Engineering*, Vol. 24(11), pp 1002-1013 (1998).

Appendix:

EISTRAM - Experimental Investigation of the PIE-technique

(with Harald Thunem)

In *Safety and Reliability.* Lydersen, S., Hansen, G., and Sandtorv, H., (Eds), Balkema, Rotterdam, pp 409-416, 1998.

Appendix paper is not included due to copyright.