Software reliability analysis in probabilistic risk analysis

HOLMBERG Jan-Erik

VTT Technical Research Centre of Finland, P.O. Box 1000, FI-02044 VTT, Finland (jan-erik.holmberg@vtt.fi)

Abstract: Probabilistic Risk Analysis (PRA) is a tool which can reveal shortcomings of the NPP design in general. PRA analysts have not had sufficient guiding principles in modelling particular digital components malfunctions. Digital I&C systems are mostly analysed simply and the software reliability estimates are engineering judgments often lacking a proper justification. The OECD/NEA Working Group RISK's task DIGREL develops a taxonomy of failure modes of digital I&C systems. The EU FP7 project HARMONICS develops software reliability estimation method based on an analytic approach and Bayesian belief network.
Keyword: nuclear safety; software reliability; probabilistic risk analysis; Bayesian belief network

1 Introduction

To assess the risk of nuclear power plant (NPP) operation and to determine the risk impact of digital systems, there is a need to quantitatively assess the reliability of the digital systems in a justifiable manner. The Probabilistic Risk Analysis (PRA) is a tool which can reveal shortcomings of the NPP design in general and PRA analysts have not had sufficient guiding principles in modelling particular digital components malfunctions.

Currently digital instrumentation and control (I&C) systems are mostly analysed simply and conventionally in PRA. The software reliability estimates are engineering judgments often lacking a proper justification. The use of probabilities for software reliability is based on some common understanding rather than a proper reference.

This paper gives an overview of the state-of-the-art in software reliability analysis in PRA. It also presents interim results of two on-going international research activities: 1) the task group DIGREL of the OECD/NEA CSNI Working Group on Risk Assessment (WGRisk) on the taxonomy of failure modes of digital components and 2) The European Union Euratom Framework Programme 7 project Harmonised Assessment of Reliability of Modern Nuclear I&C Software (HARMONICS).

The aim of DIGREL is to develop a best practice guidelines on failure modes taxonomy for the reliability assessment of digital I&C systems for PRA. HARMONICS tackles the problem of software reliability quantification using analytical and Bayesian approaches that take into consideration all the information available, in particular evidence obtained by verification and validation (V&V).

2 State-of-the-art of software reliability in PRA for nuclear power plants

2.1 Software reliability

Software failures are in general mainly caused by systematic (*i.e*. design specification or modification) faults, and not by random errors. Software based systems cannot easily be decomposed into components, and the interdependence of the components cannot easily be identified and modelled. Applying software reliability models in the PRA context is hence not a trivial matter.

Software reliability models usually rely on assumptions and statistical data collected from non-nuclear domain and therefore may not be directly applicable for software products implemented in nuclear power plants. More important than the exact values of failure probabilities are the proper descriptions of the impact that software-based systems has on the dependence between the safety functions and the structure of accident sequences. Conventional fault tree approach is, on the other hand, considered sufficient for the modelling of reactor protection system (RPS) like functions.

In spite of the unsolved issue of addressing software failures there seems be a consensus regarding some

Received date: December 3, 2012

philosophical aspects of software failures and their use in developing a probabilistic model. The basic question: "What is the probability that a safety system or a function fails when demanded" is a fully feasible and well-formed question for all components or systems independently of the technology on which the systems are based.[1] A similar conclusion was made in the Workshop on Philosophical Basis for Incorporating Software Failures in a Probabilistic Risk Assessment.[2]

2.2 Software reliability quantification

For the quantification of software failure rates and probabilities there are several general approaches, *e.g.*, reliability growth methods, Bayesian belief network (BBN) methods, test based methods, rule based methods[1] and software metrics based methods.[3][4] These methods are reviewed by Chu et al.[5]

Software reliability models can be also divided into white box models, which consider the structure of the software in estimating reliability, and black box models, which only consider failure data, or metrics that are gathered if test data is not available.[6] Black box models generally assume that faults are perfectly fixed upon observation without introducing any new faults. Software reliability is thus increasing over time. Such models are known as Software Reliability Growth Models (SRGMs). Many of these have predictive power only over the short term, but long term models have also been developed.[7]

The BBN methodology has been adapted to software safety assessment[8] and the methodology can be considered as promising. One of the main drawbacks is that a different BBN has to be built for each software development environment. This problem may be solved by using generalized BBN templates which are not restricted to a specific development environment.[9] Application of BBN is further discussed in Section 4.

In test based methods a program is executed with selected data and the answer is checked against an 'oracle'. A reliability measure can be generated, by running a number of tests and measuring the number of failures. Test-based reliability models assume that the input data profile used during the test corresponds to the input profile during real operation. Unfortunately, this correspondence cannot often be guaranteed. Statistical testing may be, however, used as an input to a BBN model.

Context-based Software Risk Model (CSRM) allows assessing the contribution of software and software-intensive digital systems to overall system risk in a way that can be integrated with the PRA format used by NASA.[10][11][12] Combination of operating experience and engineering judgment can be used to provide estimates for digital system software common-cause failures (CCF).[13]

2.3 Software reliability estimation in PRA

In the context of PRA for NPPs, there is an on-going discussion on how to treat software reliability in the quantification of reliability of systems important to safety. It is mostly agreed that software could and should be treated probabilistically[1][2] but the question is to agree on a feasible approach.

Software reliability estimation methods described in academic literature, shortly discussed in the previous chapter, are not applied in real industrial PRAs for NPPs. Software failures are either omitted in PRA or modelled in a very simple way as common cause failure (CCF) related to the application software of operating system (platform). It is difficult to find any basis for the numbers used except the reference to a standard statement that 1E-4 per demand is a limit to reliability claims, which limit is then categorically used as a screening value for software CCF.

The engineering judgement approaches used in PRA can be divided into the following categories depending on the argumentation and evidence they use:[14]

- screening out approach
- screening value approach
- expert judgement approach
- operating experience approach.

The reliability model used for software failures is practically always the simple "probability of failure per demand" (pfd).

2.3.1 Screening out approach

Screening out approach means that software failures are screened out from the model. The main arguments to omit software are that 1) the contribution of software failures is insignificant or that 2) no practical method to assess the probability of software failure (systematic failure).

One approach is to model software failures but not to define reliability values. The impact of software failures is assessed through sensitivity approaches. This approach has been utilized, for instance, in Ringhals 2.[15] In another approach, values 0, 1E-4 and 1E-3 for pfd were used in sensitivity analyses as software failure probabilities to analyse the impact of software failures on the system unavailability and the plant risk. [16]

2.3.2 Screening value approach

Screening value approach means that some reliability number, like pfd = 1E-4, is chosen without detailed assessment of the reliability, and it is claimed that this is a conservative number for a software CCF. The screening value is taken from a reference like IEC 61226.[17] Accordingly, the "Common Position" document states that reliability claims "q < 1E-4" for a single software based system important to safety shall be treated with extreme caution.[18] This derives partly due to the fact that demonstrating lower probabilities, *e.g*., by statistical testing is very laborious.

2.3.3 Expert judgement approach

Expert judgement approach relies on the assessment of the features of the software system which are assumed to have correlation with the reliability. The two questions are 1) which features should be considered and 2) what is the correlation between the features and the reliability. This kind of approaches is used extensively in PRA, *e.g*., in human reliability analysis. Such models are difficult to validate.

In a case study on quantitative reliability estimation of a software-based motor protection relay, Bayesian networks were used to combine evidence from expert judgment and operational experience.[18]

In one study, it was assumed that the contribution from software failure to total failure probability is 10% of the hardware failure probabilities.[19] The rationale to this was that there are two well recognized aspects of software reliability: 1) the contribution of software failures to total failure of a digital system is smaller compared to exclusive failure of hardware, 2) there is a threat of software related common cause failures for a group of identical and redundant components. The second aspect was addressed by selecting a suitable value for β in the beta-factor CCF model. Value β = 0.03 was given, including CCFs due to hardware and software.

SIL-value (safety integrity level of IEC 61508[20]) approach is also an example of an expert judgement approach, where the reliability target implied by the SIL is interpreted as the unavailability of the item. To apply SIL-values is a controversial issue, and at least the following weaknesses may be mentioned:[21] it does not differentiate between functions implemented by the system and the failure modes of the system; it is silent regarding the contribution of systematic failures; it does not give any indication for the estimation of beta-factors or other parameters that can be used to characterize CCFs; the notion of "system" is not defined.

2.3.4 Operating experience approach

Operating experience approach means an assessment based on operational data. In reality, operating experience approach is like the expert judgement approach since operational data need to be interpreted in some way to be used for reliability estimation.

In the PRA study of the Swedish NPP Ringhals 1, the contribution of software CCF to the unavailability of a safety system was assessed based on operational experience.[22] The operational experience of over 60 similar systems showed no CCF caused by platform properties and thus the contribution of platform CCF was estimated at 1E-8. Two events could be considered as a CCF, which leads to an unavailability of safety I&C systems as 1E-6. This value was applied for redundant I&C units.

In one study,[13] reasonable estimates for the relative contribution of software to digital system reliability software CCF probabilities were developed based on operational experience and engineering judgment. The

CCF of operating system software was estimated as 1E-7 based on data gathered from dozens of plants during a time period of more than 10 years. For the application software, the CCF probability was estimated as 1E-5 for each function group. The SIL-4 targets were used as a general guide in the estimate. Additionally, it is suggested that if multiple application software CCFs appeared in same cut set the dependency between the two CCFs should be assessed. One way to take this into consideration is to assume a beta factor between the two software CCF events. Values $0.001 < \beta < 0.1$ were recommended, depending on the similarity of the software.

2.4 Conclusions on software reliability in PRA

Generally, only common cause failures are modelled in PRA. One reason for this is that there has not been a methodology available to correctly describe and incorporate software failures into a fault tree model. The only reliability model which is applied is constant unavailability and this is used to represent the probability of CCF per demand. Spurious actuations due to software failures are not modelled or no need to consider software failure caused spurious actuations has been concluded.

Software CCF is usually understood as the application software CCF or its meaning has not been specified. Software CCF is generally modelled between processors performing redundant functions, having the same application software and on the same platform. One of the exceptions is the design phase PRA made for the automation renewal of the Loviisa NPP, where four different levels of software failures are considered: 1) single failure, 2) CCF of a single automation system, 3) CCF of programmed systems with same platforms and or software, and 4) CCF of programmed systems with different platforms and or software.[14]

With regard to the reliability numbers used in PRA, it is difficult to trace back where they come from — even in the case of using operating experience. The references indicate the sort of engineering judgement but lacks supporting argumentation.

3 Failure modes taxonomy

3.1 Background

In 2007, the OECD/NEA CSNI directed the Working Group on Risk Assessment (WGRisk) to set up a task group to coordinate an activity in the digital system reliability field. One of the recommendations of this activity was to develop a taxonomy of failure modes of digital components for the purposes of PRA.[23] This resulted in a follow-up task group called DIGREL. An activity focused on development of a common taxonomy of failure modes was seen as an important step towards standardised digital I&C reliability assessment techniques for PRA.

The DIGREL task has taken advantage from on-going R&D activities, actual PRA applications as well as analyses of operating experience related to digital systems in the OECD/NEA member countries. The scope of the taxonomy includes both protection and control systems of a nuclear power plant, though primary focus is on protection systems. Results presents here should be considered preliminary proposals and not as the Task Group consensus thoughts. DIGREL work has been reported in references [15][24][25].

3.2 General approach

Failure modes taxonomy is a framework of describing, classifying and naming failure modes associated with a system. Standard technological equipment of NPP protection systems, like pumps, are either in the running or standby mode. On the opposite, computer based systems are typically always in the running mode – the difference in the modes is that they process different sets of input parameters and consequently solve different branches of algorithms. The need of specific taxonomy establishment is hence obvious.

One of the main uses of digital equipment failure modes taxonomy is to support the performance of reliability analyses and to unify the operational experience data collection of digital I&C systems. In PRA, failure modes taxonomy is applied in the systems analysis, including the performance of FMEA (failure modes and effects analysis) and the fault tree modelling. Systems analysis is a

combination of top down and bottom up approaches. Fault tree modelling is a top down method starting from the top level failure modes defined for the system. In the system level, the two main failure modes are 1) failed function and 2) spurious function. For the failed function more descriptive definitions may be given such as "no function", "not sufficient output", "no state transition", "broken barrier", "loss of integrity", and "masking failure", depending on the nature of the system. In the fault tree analysis, the system level failure modes are broken down further into sub-system and component level failure modes. The system level failure modes appear thus as fault tree gates in the PRA model, while component level failure modes appear as basic events.

Basically, the same failure modes taxonomy can be applied for components as at the system level (failed function, spurious function), but the definitions are usually more characterising, *e.g.*, "sensor freeze of value", and are closer related to the failure mechanisms or unavailability causes. The component level failure modes are applied in the performance of the FMEA, which is a bottom-up analysis approach. The analysis follows the list of components of the system and for each component failure modes, failure causes (mechanisms) and associated effects are identified.

In PRA, the definitions for the failure modes and the related level of details in the fault tree modelling can be kept in a high level as long as relevant dependencies are captured and reliability data can be found.

3.3 Requirements for the failure modes taxonomy

The development of a taxonomy is dependent on the overall requirements and prerequisites since they will set boundary conditions *e.g.* for the needed level of detail of hardware components and for the structure of the failure modes. A different set of requirements may result in a different taxonomy. The following targets for the taxonomy have been defined:

- Defined unambiguously and distinctly
- Forms a complete/exhaustive set, mutually exclusive failure modes
- Organized hierarchically
- Data to support the taxonomy should be available
- Analogy between failure modes of different components
- The lowest level of the taxonomy should be sufficient to pinpoint existing dependencies of importance to PRA modelling
- Supports PRA practice, *i.e*. appropriate level for PRA, and fulfil PRA requirements/conditions
- Captures defensive measures against fault propagation and other essential design features of digital I&C.

3.4 Levels of details of the taxonomy

With regard to the analysis and modelling of protection systems, the following levels of details have to be distinguished:

1. the entire system
2. a division (or channel)
3. I&C units
4. modules
5. basic components.

A safety system is the entity performing a safety function or part of it. In PRA, RPS is never treated as a black box, but the analysis is always broken down into the protection functions and at least to the divisional level.

The divisions may be of the same or different architectures but in general all perform the same functions. Each division comprises an entity from power supply and physical separation point of view, although some cross-connections of power supply between divisions may be applied for certain components. From the PRA modelling point of view, a usual simplification is to assume a loss of complete division in case of a hazard affecting the division (fire or flooding initiating event). Loss of AC or DC power supply is also division wide functional failure to be considered in PRA.

Each division consists of several I&C units (*e.g.* acquisition and processing and voting units) and data buses between them. I&C units are installed in cabinets, each of which has a specific power supply route and condition monitoring. Cabinet level is the most detailed level from the power supply and room dependency point of view.

An I&C unit is a computerised system designed to receive input signals, perform computing and send output. It consists of modules such as input module, processing module, communication module and output module. Modules may be further broken down into basic components such as an analog/digital converter, a multiplexer, a microprocessor and its associated components, a demultiplexer, an A/D converter and channels of an I/O module, *e.g.*, depending on the available failure data. Modules and I/O channels are the most detailed level from the hardware functional dependency point of view. Also the software components can be associated with the modules (see Table 1).

Table 1 Software modules in a typical reactor protection system

I&C unit	Software modules
Acquisition and processing unit (APU)	Operating system APU elementary functions APU application specific software APU communication modules/features
Voting unit (VU)	Operating system VU elementary functions VU application specific software VU communication modules/features

3.5 Failure modes

In DIGREL, the main approach is to define failure modes functionally. At the system and division level, there are basically two failure modes: "failure to actuate the function" and "spurious actuation".

At lower levels (I&C unit, module, basic component), it is relevant to consider more aspects of failure modes, *i.e.*,

- The fault location (in which hardware or software module the fault is located)
- Local effect: 1) Fatal, ordered failure (generation of outputs ceases, outputs are set to specified, supposedly safe values), 2) Fatal, haphazard failure (generation of outputs ceases, outputs are in unpredictable states), 3) Non-fatal, plausible behaviour (generation of outputs continues, an external observer cannot determine whether the I&C unit or the hardware module has failed or not), 4) Non-fatal, non-plausible behaviour (generation of outputs continues, an external observer can decide that the I&C unit or the hardware module has failed).
- Detection situation: On-line detection, off-line detection, revealed only by demand, spurious effect.

The combination of fault location, local effect, detection situation together with the fault tolerant design of the system are usually sufficient to determine the functional end effect, such as

- Loss of all functions (outputs) of the I&C unit (APU/VU),
- Loss of a specific function,
- Spurious function.

The above list is not exhaustive, and, *e.g.*, for voting logics or in case of intelligent validation of input signals the functional end effect may be more complex (*e.g.* degraded voting logic). Anyway, the module level (both hardware and software) seems to be sufficient to analyse dependencies important to PRA, at least for protection systems.

4 Safety justification framework

4.1 Safety case

A Safety Case is a structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment.[26] Current safety case practice uses a basic approach where claims are supported by evidence and a "warrant" that links the evidence to the claim.

The actual claim decomposition and structuring is normally very informal and argumentation is seldom explicit. In practice, the emphasis is on communication and knowledge management of the case, with little guidance on what claim or claim decomposition should be performed.

The aim of the HARMONICS project is to improve safety justifications by integrating the goal-based, rule-based, and risk-informed approaches to get a coherent process for justifying software-based systems.[27] The project will analyse the domain of applicability and acceptability of each approach, and will provide practical guidelines based in particular on the information gathered with the proposed V&V

techniques. The claim-argument-evidence approach suggested by the CEMSIS project[28] can be used to shape the claims and argument, and to provide the evidence part of the justification. To overcome the shortcomings of the present approaches for software failure rate estimation, an analytical approach [21] and Bayesian belief network are considered in HARMONICS.

4.2 Software reliability assessment case

Assessment of the reliability of safety-critical software in NPP includes several types of reasoning. The "software reliability assessment case" structures the issues related into smaller, manageable issues, which can be discussed as independently (or conditionally independently) of each other as reasonably possible.[29] Three categories of issues may be distinguished.

Firstly, the purpose and scope of the software reliability assessment need to be defined, which — in simple terms — can be associated with the needs of PRA and its applications, *e.g.*, to show the compliance with the numerical risk criteria for the NPP. The needs of PRA can be presented by system failure mode related claims which aim at defining the basic events in the PRA model, as discussed in Section 3.

A second category of reasoning is related to the properties of the software in order to limit the scope of residual software faults that are of concern. Cat. A software systems are designed following strict design principles and they go through a rigorous V&V process, which gives well-justified arguments to rule out a number of software fault types, *e.g.*, software is designed to behave cyclically, behaviour is time-based rather than event-based, and the operating system is designed not to be affected by the plant conditions.

The third category of claims concerns the actual assumptions used in the modelling of the software reliability once the scope of the assessment has been defined, *i.e.*, the system failure mode and conceivable software fault modes are given.

4.3 Software reliability claims

Software reliability can be measured in several manners, *e.g.*:

- the probability that the software is imperfect (not fault-free): P(SW imperfect)
- the probability distribution for number of residual faults: P(N = n), N = number of faults
- the probability (or failure rate) of the critical digital system failure (due to a software fault): P(SW failure)
- the conditional probability of a common cause failure (called also β-factor).

The two first ones are clearly interrelated, since P(SW imperfect) = P(N > 0), although there may be difference in which way evidence is used to assess them. While Fault-freeness is a true-false property to be assessed, P(N = n) requires a construction of a probability model for the number of residual faults.

The third and fourth metrics are needed for PRA. It may be difficult to directly estimate P(SW failure), and we may need to build the reasoning via the assessment of the imperfectness or number of faults. Estimation of beta-factor must be essentially based on the assessment of the software diversity.

4.4 Bayesian belief network (BBN)

BBN is a general model for probabilistic inference so that the conditional dependences between the random variables are presented in a directed acyclic graph.[30] In the HARMONICS context, the random variables are reliability claims related to the software and various pieces of evidence available for reliability assessment.

BBNs have been suggested for software reliability estimation in several references by modelling features such as:

- quality of the producer and complexity of the problem as the top nodes, and the testing part[31]
- the design process quality, the complexity of the problem, the testing quality and the operational usage[32]
- the evidence on the reliability provided by the SW development team[8]
- fault insertion-and-removal process during the SW development[33]

- evidence related to both the development process and the final product.[34]

A more thorough discussion on the same topic, including idioms of BBN fragments commonly occurring in safety assessments and instructions on how to construct safety arguments with BBN is found in reference [35].

4.5 Types of evidence

The key issue is how different pieces of evidence are interpreted in a probability model context and how their interrelationships are assessed. Table 2 lists various sources of evidence available for the independent confidence building.

Table 2 Types of evidence for SW reliability estimation

Source of evidence	Types of verification	Reliability evidence
Process quality	Compliance with the requirements	Indirect evidence
	Developers' experience	
	Correctness of the development tools	Fault freeness with certain confidence
Product analysis	Functional and structural properties of software	Indirect evidence
	Logic proof of correctness	Fault freeness with certain confidence
Product testing	Validation of correct operation	Fault freeness with certain confidence
	Confirmation of the reliability	Direct evidence given the representativeness of test cases

Indirect evidence needs to be measured by some rating scale which needs to be interpreted in terms of reliability. The use of indirect evidence in a justifiable manner would require some validation of the relationship between the evidence and reliability.

Fault freeness evidence can be used to exclude certain software fault modes. From the reliability point of view there may remain doubt on the correctness of the evidence, which can be expressed as a probability. The way of expressing numerically (probabilistically) confidence on fault freeness evidence is needed not only from the reliability assessment point of view but such an assessment is a relevant part of the overall safety justification of software systems.

Direct evidence is in principle the optimal case, but in reality the confidence on the representativeness of the data may need to be included in the assessment.

4.6 BBN model suggestions in HARMONICS

The viewpoint of HARMONICS is that of the user of the software or of an independent evaluator that has to make his conclusions based on the information provided by the system vendor or the power utility applying for the license.

The BBN model should represent a class of similar kind of SW fault induced system failures. For instance, we may have a common BBN model for application SW failures of a certain platform. Same model can be used to estimate a number of pfd's (and frequencies of spurious actuations, as well), there may be some variation in the evidence for different pfd's. For another platform or for operating system SW failures or for elementary function failures, different BBN model may be required.

The evidence (observable or potentially observable quantities) on SW reliability can partitioned into

1. those affecting pfd (kind of "performance shaping factors" as in human reliability analysis context)
2. those affected by pfd, such as test results and operating experience.

Given pfd, the first and the second set of quantities are independent on each other (Fig. 1). There is no direct link from the set 1 to set 2. This limitation could naturally be relaxed, but it would complicate considerably the mathematics. Therefore, the aim is to search for and define the pieces of evidence (set 1 and 2) in such a way that the conditional independence can be justified.

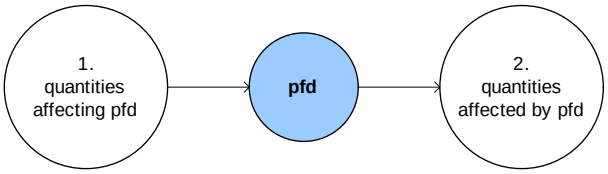

Fig.1 Basic structure of the BBN model for estimating pfd.

5 Conclusions

The advent of digital I&C systems in nuclear power plants has created new challenges for safety analysis. To assess the risk of nuclear power plant operation and to determine the risk impact of digital systems, there is a need to quantitatively assess the reliability of the digital systems in a justifiable manner. Due to the many unique attributes of digital systems, a number of modelling and data collection challenges exist, and consensus has not yet been reached.

Currently in PRA computer-based systems are mostly analysed simply and conventionally. The conventional failure mode and effects analysis and failure tree modelling are utilized. As basic events, I&C unit failures, application software failures and CCFs between identical components are modelled. However it is not clear which failure modes or system parts CCFs should be postulated. The primary goal is to model dependencies.

A clear distinction can be made between the treatment of protection and control systems controlling *e.g*. the turbine plant. There is a general consensus that protection systems shall be included in PRA, while control systems can be treated in a limited manner.

The survey of literature and PRA shows that software failures are either omitted in PRA or modelled in a very simple way as CCF related to the application software of operating system. It is a difficult basis for the numbers used except the reference to a standard statement that a failure probability 1E-4 per demand is a limit to reliability claims, which limit is then categorically used as a screening value for software CCF.

In the OECD/NEA DIGREL task, the taxonomy will be developed jointly by PRA and I&C experts. An activity focused on the development of a common taxonomy of failure modes is seen as an important step towards standardised digital I&C reliability assessment techniques in PRA. PRA needs will guide the work, meaning *e.g*. that I&C system and its failures are studied from their functional significance point of view. The taxonomy will be the basis of future modelling and quantification efforts. It will also help define a structure for data collection and to review PRA studies.

Bayesian belief network is a potential approach for the estimation of software reliability. HARMONICS explores the use of SIL, operational experience or testing and software complexity as evidence when evaluating the pfd of the software. An advantage of BBNs is that they enable combining different types of evidence in the same model. While such empirical data set seem hard to be found, expert elicitations are needed.

Acknowledgement

Contributions from the WGRISK/DIGREL task group members are acknowledged. The following organisations form November 2012 the task group, being responsible for planning and organisation of work meetings and preparation of the best practice guidelines: VTT, Finland; Risk Pilot, Sweden; IRSN, France; EDF, France; AREVA, France; GRS, Germany; KAERI, Korea; NRC, USA; Ohio State University, USA; NRI, Czech; JNES, Japan; VEIKI, Hungary; ENEL, Italy; NRG, the Netherlands; RELKO, Slovakia and CSNC, Canada. The Finnish work has been financed by NKS (Nordic nuclear safety research), SAFIR2014 (The Finnish Research Programme on Nuclear Power Plant Safety 2011–2014) and the members of the Nordic PSA Group: Forsmark, Oskarshamn Kraftgrupp, Ringhals AB and Swedish Radiation Safety Authority. NKS conveys its gratitude to all organizations and persons who by means of financial support or contributions in kind have made the work presented in this report possible.

HARMONICS is co-funded by the European Commission, the UK C&I Nuclear Industry Forum (CINIF) and the consortium organisations. The project consortium has five partners: VTT Technical Research Centre of Finland, Électricité de France (EDF), Institute for Safety Technology (ISTeC) from Germany, Adelard LLP from UK and Strålsäkerhetsmyndigheten (SSM) from Sweden. The public website address of the project is http://harmonics.vtt.fi. The author acknowledges the contributions of the other HARMONICS project members.

Nomenclatures

BBN Bayesian belief network
BN Bayesian network
CCF common-cause failures
CEMSIS Cost Effective Modernisation of Systems Important to Safety
CSRM Context-based Software Risk Model
DIGREL Guidelines for reliability analysis of digital systems in PSA context
FMEA failure modes and effects analysis
HARMONICS Harmonised Assessment of Reliability of Modern Nuclear I&C Software
I&C instrumentation and control
NPP nuclear power plant
pfd probability of failure per demand
PRA Probabilistic Risk Analysis
SRGMs Software Reliability Growth Models
V&V verification and validation
WGRisk Working Group on Risk Assessment

References

[1] DAHLL, G., LIWÅNG, B., and PULKKINEN, U.: Software-Based System Reliability. Technical Note, NEA/SEN/SIN/WGRISK(2007)1, Paris: Working Group on Risk Assessment (WGRISK) of the Nuclear Energy Agency, 2007.

[2] CHU, T.-L., MARTINEZ-GURIDI, G., and YUE, M.: Workshop on Philosophical Basis for Incorporating Software Failures in a Probabilistic Risk Assessment. Brookhaven National Laboratory, BNL-90571-2009-IR, 2009.

[3] SMIDTS, C., and LI, M.: Software Engineering Measures for Predicting Software Reliability in Safety Critical Digital Systems, NUREG/GR-0019, Washington D.C.: United States Nuclear Regulatory Commission, 2000.

[4] SMIDTS, C., and LI, M.: Preliminary Validation of a Methodology for Assessing Software Quality, NUREG/CR-6848, Washington D.C.: United States Nuclear Regulatory Commission, 2004.

[5] CHU, T.-L., YUE, M., MARTINEZ-GURIDI, G., and LEHNER, J.: Review of Quantitative Software Reliability Methods, BNL-94047-2010, Brookhaven National Laboratory, 2010.

[6] ZHANG, Y.: Reliability Quantification of Nuclear Safety-Related Software. PhD Thesis in Nuclear Engineering, Cambridge: Massachusetts Institute of Technology, 2004.

[7] BISHOP, P.G., and BLOOMFIELD, R.E.: A Conservative Theory for Long Term Reliability Growth Prediction". In IEEE Trans. Reliability, 1996, 45(4): 550–560.

[8] HAAPANEN, P., HELMINEN A., and PULKKINEN U.: Quantitative reliability assessment in the safety case of computer-based automation systems. STUK-YTO-TR 202. Helsinki: STUK, 2004.

[9] EOM, H.-S., PARK, G.-Y., KAG, H.-G., and JANG, S.-C.: Reliability Assessment Of A Safety-Critical Software By Using Generalized Bayesian Nets, Proc.6th American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies, NPIC&HMIT 2009, Knoxville, Tennessee, April 5–9, 2009.

[10] YAU, M., and GUARRO, S.: Application of Context-based Software Risk Model (CSRM) to Assess Software Risk Contribution in Constellation Project PRAs, Proc. 10th International Probabilistic Safety Assessment & Management Conference, PSAM 10, Seattle, Washington, June 7–11, 2010, paper 186.

[11] GUARRO, S.: Risk-Informed Safety Assurance and Probabilistic Assessment of Mission-Critical Software-Intensive Systems, NASA Technical Paper AR 07-01; JSC-CN-19704, ASCA, Inc., Redondo Beach, CA, 2007.

[12] VESELY, W., STAMATELATOS, M., DUGAN, J., FRAGOLA, J., MINARICK III, J., and RAILSBACK, J.: Fault Tree Handbook with Aerospace Applications, Washington D.C.: NASA, 2002.

[13] ENZINNA, B., SHI, L., and YAN, S.: Software Common-Cause Failure Probability Assessment," Proc.6th American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies, NPIC&HMIT 2009, Knoxville, Tennessee, April 5–9, 2009.

[14] BÄCKSTRÖM, O., and HOLMBERG, J.-E.: Use of IEC 61508 in Nuclear Applications Regarding Software Reliability, Proc. of 11th International Probabilistic Safety Assessment and Management Conference & The Annual European Safety and Reliability Conference, 25–29.6.2012, Helsinki, paper 10-Th2-4.

[15] AUTHÉN, S., BJÖRKMAN, K., HOLMBERG, J.-E., and LARSSON, J.: Guidelines for reliability analysis of digital systems in PSA context — Phase 1 Status Report," NKS-230, Roskilde: Nordic nuclear safety research, 2010.

[16] KANG, H. G., and JANG, S.-C.: Issues And Research Status For Static Risk Modeling Of Digitalized Nuclear Power Plants, Proc.6th American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies, NPIC&HMIT 2009, Knoxville, Tennessee, April 5–9, 2009.

[17] Nuclear power plants – Instrumentation and control systems important to safety – Classification of instrumentation and control functions, IEC 61226.

Second edition. International Electrotechnical Commission, Geneva, 2005.

[18] Licensing of safety critical software for nuclear reactors – Common position of seven European nuclear regulators and authorized technical support organisations, SSM Report 2010:01, Stockholm: SSM, 2010.

[19] VARDE, P. V., CHOI, J. G., LEE, D. Y., and HAN, J. B.: Reliability Analysis of Protection System of Advanced Pressurized Water Reactor-APR 1400, KAERI/TR-2468/2003, Korea Atomic Energy Research Institute, 2003.

[20] Functional safety of electrical/electronic/programmable electronic safety-related systems. IEC 61508, second edition. Geneva: International Electrotechnical Commission, 2010.

[21] Estimating Failure Rates in Highly Reliable Digital Systems. EPRI TR-1021077, Palo Alto: Electric Power Research Institute, Inc., 2010. Limited distribution.

[22] AUTHÉN, S., WALLGREN, E., and ERIKSSON, S.: Development of the Ringhals 1 PSA with Regard to the Implementation of a Digital Reactor Protection System," Proc. 10th International Probabilistic Safety Assessment & Management Conference, PSAM 10, Seattle, Washington, June 7–11, 2010, paper 213.

[23] Recommendations on assessing digital system reliability in probabilistic risk assessments of nuclear power plants, NEA/CSNI/R(2009)18, Paris: OECD/NEA/CSNI, 2009.

[24] AUTHÉN, S., GUSTAFSSON, J., and HOLMBERG, J.-E.: Guidelines for reliability analysis of digital systems in PSA context — Phase 2 Status Report, NKS-261, Roskilde: Nordic nuclear safety research (NKS), 2012.

[25] HOLMBERG, J.-E., AUTHÉN, S., and AMRI, A.: Development of best practice guidelines on failure modes taxonomy for reliability assessment of digital I&C systems for PSA, Proc. of 11th International Probabilistic Safety Assessment and Management Conference & The Annual European Safety and Reliability Conference, 25–29.6.2012, Helsinki, paper 10-Th4-1.

[26] Safety Management Requirements for Defence Systems, Ministry of Defence Standard 00-56 Issue 4, June 2007.

[27] BISHOP, P. G., BLOOMFIELD, R., GUERRA, S., and THUY, N.: Safety justification frameworks: integrating rule-based, goal-based and risk informed approaches, Proc. of the 8th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies, NPIC & HMIT 2012, San Diego, July, 2012, LaGrange Park: American Nuclear Society, USA.

[28] Cost Effective Modernisation of Systems Important to Safety. Work Package 0. Final Public Synthesis Report (first issue), 2004. http://www.cemsis.org/

[29] HOLMBERG, J.-E., BISHOP, P., GUERRA, S., and THUY, N.: Safety case framework to provide justifiable reliability numbers for software systems, Proc. of 11th International Probabilistic Safety Assessment and Management Conference & The Annual European Safety and Reliability Conference, 25–29.6.2012, Helsinki, paper 10-Th2-2.

[30] PEARL, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Representation and Reasoning Series (2nd printing ed.). San Francisco: Morgan Kaufmann. 2007.

[31] GRAN, B. A.: Assessment of programmable systems using Bayesian belief nets. Safety Science, 2002, 40: 797–812.

[32] FENTON, N., NEIL, M., and MARQUEZ, D.: Using Bayesian networks to predict software defects and reliability. JRR161 IMechE 2008 Proc. IMechE Vol. 222 Part O: J. Risk and Reliability.

[33] CHU, T.L., YUE, M., VARUTTAMASENI, A., KIM, M.C., EOM, H.S., SON, H.S., and AZARM, A.: Applying Bayesian belief network method to quantifying software failure probability of a protection system, Proc. of the 8th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies, NPIC & HMIT 2012, San Diego, 22–26.7.2012, LaGrange Park: American Nuclear Society: 296–307.

[34] BOUISSOU, M., MARTIN, F. and OURGHANLIAN, A.: Assessment of a Safety-Critical System Including Software: A Bayesian Belief Network for Evidence Sources. 1999 Proceedings Annual Reliability and Maintainability Symposium.

[35] The SERENE Method Manual, Task 5.3 Report. EC Project SERENE (Safety and Risk Evaluation Using Bayesian Nets) No. 22187, 1999.