Probabilistic modelling of common cause failures in digital I&C systems - Literature review

Tyrväinen, Tero

RESEARCH REPORT

VTT-R-00728-21

Probabilistic modelling of common cause failures in digital I&C systems - Literature review

Authors: Tero Tyrväinen

Confidentiality: VTT Public

Report's title Probabilistic modelling of common cause failures in digital I&C systems - Literature review	
Customer, contact person, address VYR	**Order reference** SAFIR 3/2021
Project name New developments and applications of PRA	**Project number/Short name** 128648/NAPRA
Author(s) Tero Tyrväinen	**Pages** 14/1
Keywords Common cause failure, digital I&C, probabilistic risk assessment, software failure	**Report identification code** VTT-R-00728-21

Summary

This report presents a state of the art review on probabilistic risk assessment of CCFs in digital instrumentation and control (I&C) systems of nuclear power plants. It covers a literature study and a questionnaire to Finnish nuclear power companies. Both software and hardware CCFs are in the scope of the report. There is relatively little literature addressing these CCFs. Concerning both software and hardware, lack of data is the main challenge, and there is need for data collection and method development activities.

Software CCF probabilities are usually based on either expert judgments or operating experience. Software reliability analysis methods in scientific literature do not usually address CCFs specifically. Only one method that focuses on software CCFs of a digital reactor protection system was found.

It is generally agreed that CCFs between identical redundant software modules can be modelled assuming full dependency. However, when there is some diversity present, the modelling is challenging. Some guidance exists for such cases nevertheless.

Hardware CCFs can be analysed according to normal CCF analysis principles. However, lack of data on digital I&C components often makes it necessary to use generic parameters or engineering judgment -based methods, which can lead to quite conservative results. Large and asymmetric CCF groups are one particularly challenging area related to digital I&C hardware.

Confidentiality	VTT Public

Written by Tero Tyrväinen, Research Scientist	**Reviewed by** Kim Björkman, Research Scientist

VTT's contact address
VTT Technical Research Centre of Finland Ltd, P.O. Box 1000, FI-02044 VTT, FINLAND

Distribution (customer and VTT)
SAFIR2022 RG2 members, VTT archive

Approval

Date:	23.9.2021
Signature:	DocuSigned by: Nadezhda Gotcheva E21E683840FD424...
Name:	Nadezhda Gotcheva
Title:	Research Team Leader

Contents

1. Introduction

Modelling of digital instrumentation and control (I&C) systems is one of the most challenging areas in probabilistic risk assessment (PRA) of nuclear power plants. The reasons for this are the lack of sufficient failure data on digital I&C systems, the nature of software failures and the complexity of the I&C systems. Software failures can be caused, for example, by design errors, requirements specification errors or programming errors, and therefore, their probabilities are difficult to estimate.

The majority of the risk related to digital reactor protection systems comes from common cause failures (CCFs), because those systems are designed to tolerate a single failure. Common cause failures of both hardware and software components are potentially important. The analysis of hardware and software CCFs is quite different because of the different nature of those failures. Software CCFs are particularly challenging to analyse, but there may not also be sufficient data to estimate CCF probabilities of large groups of hardware components.

As software failures are systematic in nature, it is very likely that redundant and identical software modules used for the same purpose fail at the same time, if one module fails. Typically, the beta-factor model with the beta parameter set to 1 is applied in that case. Cases where software modules are not identical or are used to process different signals are more challenging, because it may be too conservative to use beta-factor of 1 and it is very difficult to estimate the correlation between failures.

There is also not much data on hardware CCFs in digital I&C systems. Common cause component groups (CCCGs) may grow large if identical components are used for several different purposes. Lack of data is a problem particularly for large groups, and also large CCCGs can be difficult to handle in PRA models.

Literature related to software CCFs is surveyed in Section 2 and related to hardware CCFs in Section 3. In addition to the literature survey, a questionnaire has been prepared for Finnish nuclear power companies. Its results are summarised in Section 4. Section 5 concludes the report.

2. Software common cause failures

Software failures can be caused by faults in requirements specification, design or software implementation. Therefore, it is very likely that a failure occurs in identical redundant software modules at the same time, if it occurs in one. Because of this, software failures that are modelled in PRA are mostly CCFs.

In addition to a fault, a software failure requires also a trigger. A fault can lead to a failure only when such a trigger occurs that the fault impacts on the functioning of the software and the software functions incorrectly. Therefore, a CCF between software modules can occur only if there is a common trigger. The common trigger can be common signal trajectory (e.g. rare plant condition), human action, external event or temporal effect (IAEA 2009). To reduce CCFs, one has to therefore either reduce the number of common faults or probability of common trigger. Common faults can be reduces by diversity in the development of different (sub)systems, whereas common triggers can be reduced by diversity in the signal trajectories e.g. by diversity in inputs.

Software CCFs are discussed in general e.g. in (IAEA 2009), and methods have been developed to identify software CCFs (Bao et al. 2020). In this report, the focus is on PRA modelling of software CCFs.

2.1 Software failure probabilities in literature

Bäckström et al. (2015) prepared a literature review on software reliability analysis, which still covers most of the available references. Software failures have often been modelled in a very simple manner in PRA or not modelled at all, because there have not been sufficient data and methods to estimate failure probabilities. A presumably conservative probability, e.g. 1E-4, is often used without any detailed analysis. The view of Regulator Task Force on Safety Critical Software (2018) is that reliability claims smaller than 1E-4 should be treated with caution.

Operating experience is, of course, preferred in failure probability estimation, if sufficient data is available. In some studies, software failure probabilities have been estimated based on operating experience:

- According to (Bäckström et al. 2015), software CCF probability of 1E-6 was estimated for safety automation systems in Ringhals 1 nuclear power plant.
- Enzinna et al. (2009) estimated probability of 1E-7 for operating system software CCF, and probability 1E-5 for application software CCF based on operating experience.
- AREVA (2013) estimated operating system CCF failure probability of 1E-7 based on TXS operating experience from 44 units. A conservative 95th percentile value was used assuming that a failure causes the system to be unavailable for one hour. Beta-factor of 1 was used as the CCF parameter.
- Bickel (2008) estimated the failure rate of latent software design error as 1.18E-6/h, based on failure data from the first generation of digital reactor protection systems. The failure rate was however not estimated directly for PRA use. Other types of CCFs in digital reactor protection systems were also analysed based on observed failure events.

Sometimes, software failures are screened out from PRA, because software failure probabilities are considered negligible or there is no suitable method available to estimate the probabilities (Bäckström et al. 2015). Even if failure probabilities are not defined, sensitivity analyses on the failure probabilities can be performed, such as in Ringhals 2 PRA study (Bäckström et al. 2015).

Expert judgements are also sometimes used. Varde et al. (2003) assumed that the probabilities of software failures are 10% of the hardware failure probabilities in a PRA study of advanced pressurized water reactor (APR) 1400. They applied beta-factor of 0.03 to both hardware and software.

2.2 Methods

2.2.1 Nordic approach

A Nordic software reliability analysis method (Bäckström et al. 2015; Authen et al. 2016) divides software used in a digital reactor protection system into four failure cases (with some subcases) to be modelled in PRA. The failure cases are presented in Table 1. Possible fault locations are system software (SyS), functional requirements specification of an acquisition and processing unit (APU-FRS), application software of an acquisition and processing unit (APU-AS), functional requirements specification of voting unit (VU-FRS), application software of voting unit (VU-AS), and data communication software (DCS). All failure cases represent CCFs, except 4c.

Table 1: Generic software failure modes and effects (Authen et al. 2016).

Effects	**Software fault location**					
	SyS	**APU-FRS**	**APU-AS**	**VU-FRS**	**VU-AS**	**DCS**
Loss of complete system	Case 1					Case 1
Loss of one subsystem	Case 2a	Case 2a		Case 2a	Case 2a	Case 2b
Loss of one group of redundant APUs in one subsystem		Case 3a	Case 3a			
Loss of one group of redundant voters in one subsystem				Case 3b	Case 3b	
Loss of one function in all divisions of one subsystem		Case 4a	Case 4a	Case 4b	Case 4b	
Loss of one function in one division of one subsystem		Case 4c	Case 4c			

The quantification is divided between fatal system level failures (cases 1 and 2) and non-fatal application software failure (cases 3 and 4). Probabilities of fatal failures can be estimated based on operating experience as presented in (Authen et al. 2016). Probabilities of non-fatal application software failures can be estimated based on software complexity and level of verification and validation (V&V). Basic application software failure probability has tentatively been assumed as 1E-6, and this basic probability is scaled by a shaping factor determined in Table 2. The numbers have been selected by expert judgment, but they are in line with software failure probabilities used in some PRA models. Software complexity analysis can be performed using, e.g., the SICA method (Tyrväinen et al. 2016). The V&V levels correspond to safety integrity levels. The failure probability estimate can be updated based on operating data in a Bayesian manner, if suitable data is available.

Table 2: Shaping factor for application software failure probability (Authen et al. 2016).

		Complexity		
		High	Medium	Low
V&V	0	10000	1000	100
	1	1000	100	10
	2	100	10	1
	3	10	1	0.1
	4	1	0.1	0.01

2.2.2 Other methods

A number of software reliability analysis methods can be found from literature, though real PRA studies use mostly simpler expert judgment and operating experience methods (Bäckström et al. 2015). Chu et al. (2010) reviewed many software reliability methods found from literature considering their applicability to nuclear power plant PRA. The methods were divided into four categories: software reliability growth methods, Bayesian belief networks (BBNs), test-based methods and other methods. Only a few of the methods were developed for the reliability analysis of software systems in nuclear power plants, and none of the methods was designed to quantify different types of software failures, like the Nordic method presented in the previous section. For the review, desirable characteristics of a quantification method were formulated. One of the characteristics is the capability of the method to estimate CCF parameters. However, there is no mention of such capability on any of the methods. Practically all the methods actually consider the software system as one entity instead of dividing it into modules between which there could be a CCF.

Some newer references from the past decade include:

- Chu et al. (2017) have developed a statistical testing approach to estimate the failure on demand probability of software. Test cases are created based on PRA scenarios so that they represent realistic demand conditions. The scenarios are simulated using a thermal hydraulic model to generate input signals for the software.

- Chu et al. (2018) have developed a BBN model to estimate software reliability for nuclear applications. The BBN model covers the phases of software development lifecycle and it estimates the number of defects left after each phase, e.g. based on the quality of the development process. The number of defects left is assumed to correlate with the failure probability. The model was developed with help of expert elicitation.

- Based on the two previous methods, Cai et al. (2020) have developed a software reliability analysis method for software in a reactor protection system of a nuclear power plant. A BBN model that covers different software development phases is used to calculate the failure on demand probability of the software. The failure probability is used to calculate the number of tests required to meet the reliability target. Finally, scenarios from PRA model are used to generate the test cases for the software with help of a thermal hydraulic model.

- Smidts et al. (2011) have developed a method called Reliability Prediction System (RePS) to estimate software reliability based on selected metrics, which include defect density, test coverage, requirements traceability, function point analysis, bugs per line of code, cyclomatic complexity, cause and effect graphing, requirements specification change requests, fault-days number, capability and maturity model, completeness, and coverage factor.

- Guo et al. (2019) have developed a reliability growth model for safety-critical software in a reactor protection system. The model predicts the software reliability based on faults detected during software testing taking into account the severity of the faults. It is not presented how to apply the model in PRA context.

- Shin et al. (2017) present another test-based software reliability analysis method for software in a digital reactor protection system. The test cases are created based on investigation of probable internal states of the software.

- Yang and Sydnor (2012) developed a flow network model of software. Software failure probability was modelled using binomial distribution. Software failure rates were estimated by testing.

The methods do not specifically address CCFs.

2.3 Software common cause failures in presence of diversity

There seems to be a consensus that CCFs between identical redundant software modules can be modelled using beta-factor of 1. However, CCFs between non-identical software modules and CCFs between software modules in different (sub)systems are more difficult to analyse, because beta-factor of 1 is usually too conservative and it is difficult to estimate a lower value. Causes for such CCFs can be common functional requirements specification and use of same elementary functions (Authen et al. 2015).

There are several types of diversity that can decrease the dependencies between different software modules, e.g.: independent development teams, signal diversity, different development approaches, diversity of development tools, use of different programming languages and different algorithms. It can however be difficult to achieve and justify complete statistical independence, even if several diversity measures are used.

Functional diversity is one way that has been recognised to protect against software CCFs (IAEA 2009). It typically means that the same function can be actuated based on different plant parameters processed by different subsystems. Functional diversity is often assumed to eliminate application software CCFs between functionally diversified subsystems. For example, AREVA (2013) reasoned such assumption by different functional specifications, different sensed parameters and different signal trajectories. On the other hand, operating system CCFs between functionally diverse subsystems are usually modelled, e.g. in (AREVA 2013). The Nordic approach (Authen et al. 2016) described in Section 2.2.1 also assumes potential system software CCF between subsystems, but not application software CCF between subsystems.

EPRI (2012) reasons that inter-system software CCFs may need to be modelled with a beta-factor depending on the similarity and differences. Application software CCF between systems may not need to be modelled when there is functional and signal diversity. When similar platforms are used to implement similar functions activated at different plant conditions, a beta-factor of 0.001-0.1 is recommended. If the functions are very different, it is not necessary to model CCF, even if the platforms are similar. On the other hand, EPRI (2012) recommends a beta-factor value near 1 for intra-system CCFs even if there is diversity between channels. Failures originate most likely from functional specification, and if it is the same for different channels, diversity in equipment and software does not necessarily protect from CCFs.

In the PRA of APR1400 design, Korea Electric Power Corporation and Korea Hydro & Nuclear Power Co., Ltd (2018) modelled both operating system and application software CCFs between systems when software platforms were similar. No CCFs between different software platforms were modelled. Application software CCFs were modelled separately for different component types, such as bistables, local coincidence logic modules, group controllers and loop controllers. CCFs between those were not modelled.

Authen et al. (2016) presented an approach to estimate beta-factor for CCFs between similar but non-identical application software modules. The beta-factor depends on the number of identical inputs and whether the software modules belong to the same complexity category. The approach has however not been validated.

Littlewood and Rushby (2011) have studied a special case of software CCF between a complex system and a simple system that are used to actuate the same function. It is argued that a dependency should not be ruled out, because a demand impacts both systems, and some demands are more difficult than others. Both systems have a larger failure probability when a difficult demand comes, and therefore, the failure probabilities are dependent. A conservative mathematical model for failure probability estimation was developed utilising an assumption that the simple system is possibly perfect.

3. Hardware common cause failures

Digital I&C hardware modelling can be performed in a more traditional manner than software modelling, because hardware failures are random events, like failures of any mechanical components. Normal CCF analysis principles apply to digital I&C hardware (EPRI 2012).

More complex models than beta-factor, such as alpha-factor model or multiple Greek letter model (MGLM), can be used for hardware CCFs depending on data. The parameters should preferably be estimated based on operating data. If there is no data, use of generic parameters has been suggested in (EPRI 2012; Authen et al. 2015). Jockenhövel-Barttfeld et al. (2018) stated that generic parameters are conservative. Therefore, they are usable, but may lead to overly conservative results. Engineering judgments are also used (Authen et al. 2015).

Engineering judgment -based method from IEC 61508-6 (IEC 2010) has been applied e.g. for Ringhals nuclear power plant PRA studies according to (Authen et al. 2015). The analysis covers 37 questions that determine the beta-factor for the analysed components (Sun 2013). The questions concern e.g. separation, diversity, maturity, complexity, design reviews, maintenance procedures, diagnostic tests, competence of developers, and operating environment. For a group of two components, the beta-factor can range from 0.005 to 0.1 depending on the answers and the type of the components. The method gives smaller beta-factors for logic solvers than for other components. The beta-factors can also be adjusted for larger groups with specific multipliers.

Jockenhövel-Barttfeld et al. (2019) estimated alpha-factor parameters for hardware modules based on TELEPERM XS operating experience using impact vectors. There had been no CCF events. Therefore, it was important to take into account also potential CCFs, where only one division failed, but other divisions shared the same cause and could have been affected by the same coupling mechanism. The estimated CCF parameters were compared to generic parameters from (U.S. NRC 2016), and the estimated values were significantly smaller indicating that the generic parameters are very conservative. On the other hand, the estimated parameters were in line with values estimated using the IEC 61508-6 method (IEC 2010). Priority control modules had significantly smaller CCF parameters than input/output modules.

Recent DIGMAP project (Porthin et al. 2021) raised one problematic issue related to hardware CCFs, which is that a large number of identical hardware components is sometimes used in digital I&C systems of nuclear power plants. For example, in the fictive reference case of DIGMAP, there are in total 16 identical hardware components in analog input modules in two subsystems. Large groups like these are not only a challenge for parameter estimation, but also PRA modelling. Traditional alpha-factor and MGLM models are not well suited for large groups, because the number of CCF combinations is too large to be included in PRA explicitly. On the other hand, the beta-factor model might be too simplistic and conservative. The group in the DIGMAP case is not symmetric, i.e. all CCF combinations of the same size do not have the same impact, which increases the need for a finer model.

One question mentioned in (Authen et al. 2015) is whether same CCF parameters should be applied to undetected and detected failures. Jockenhövel-Barttfeld et al. (2018) comment that CCFs of detected failures are usually not modelled, because a detected failure is repaired relatively fast, and therefore, the time window for multiple failures to occur is short. Undetected failures, on the other hand, accumulate over a longer time period. IEC 61508-6 (IEC 2010) separates the beta-factor estimation of detected and undetected failures. Detected failures have smaller beta-factors in some cases depending on the diagnostic coverage and testing interval, but the order of the magnitude is the same. Liang et al. (2020) also used smaller CCF parameters for detected failures, though it is not explained where the values come.

4. Questionnaire

A questionnaire was prepared for the Finnish nuclear power companies on CCF modelling. The questions are presented in Appendix. Two organizations answered to the questionnaire. The results are summarised in the following, and the answers are treated anonymously.

4.1 Software common cause failures

In most of the nuclear power plant units, there are not many programmable systems, and software CCFs have very small impact on the core damage risk. In one unit, the reactor protection system is digital. For that unit, software is modelled as a global failure of all software in a platform, and also CCF between platforms. Software CCFs in the measurement system of the neutron flux and multifunctional relays of the power network have also been modelled in some units.

Different probability estimation approaches have been applied at different units. The Nordic software reliability analysis method is applied in one unit (Bäckström et al. 2015), i.e. operating experience combined with conservative engineering judgments. Generic values, statistics from the supplier and conservative expert judgments have also been used. One I&C supplier is said to apply IEC 61508-6 (IEC 2010).

Typically, CCFs are modelled between software in identical equipment. However, in some cases, CCFs between software from different suppliers have been modelled, because there might be similarities in the software at low level.

CCFs between different systems have been modelled in some cases. For example, a CCF between two subsystems has been modelled as presented in (Bäckström et al. 2015) [beta-factor of 0.01 is found in the reference].

Application software CCFs are assumed to be covered by hardware CCFs at least in one plant. Spurious signals have been evaluated in one unit, but not included in the PRA model due to small impact.

More operating experience should be collected to improve the estimates. A global CCF between I&C platforms is particularly seen as an important item to study. It is also suggested that more research could be performed on root causes of software CCFs and means to prevent CCFs. General software CCF method development is also seen as beneficial.

4.2 Hardware common cause failures

MGLM, beta-factor model and primitive parameters (Johanson et al. 2003) are used for hardware common cause failures. Generic parameter values and values found from literature have been used for CCF parameters. The I&C supplier of one plant uses the IEC 61508-6 method (IEC 2010) to estimate beta-factors.

CCFs between identical components are always modelled. The components that are modelled include at least processors, input modules, output modules, communication modules and measurement sensors.

CCF groups include typically four components. An answer indicated that there are larger groups, but did not specify how large the groups are.

There is difference in the CCF modelling between undetected and detected failures in some cases. As said in Section 3, IEC 61508-6 (IEC 2010) treats those in slightly different ways. In one unit, detected CCFs are not modelled.

Lack of operating experience is a challenge in hardware CCF estimation, and there are limitations in the CCF methods. Selection of the level of modelling detail is also not a trivial task. It is time-consuming to perform failure modes and effects analysis. Root causes and prevention of CCFs could also be studied more.

5. Conclusions

This report has presented a state of the art review on PRA modelling of CCFs in digital I&C systems. Both software and hardware CCFs have been in the scope of the report. There is relatively little literature addressing these CCFs, whereas there is plenty of literature on software reliability analysis in general.

There are some references on software CCF probabilities used in PRA studies. The estimates are usually based on either expert judgments or operating experience, but even the estimates based on operating experience seem quite uncertain. Many software reliability analysis methods can be found from literature, but they have mostly not been used in practise and do not specifically address CCFs, though software failures modelled in PRA are usually CCFs. The Nordic method (Bäckström et al. 2015; Authen et al. 2016) seems to be the only practical method that analyses software CCFs of a digital reactor protection system.

It is generally agreed that CCFs between identical redundant software modules can be modelled using beta-factor of 1. However, when there is some diversity present, the modelling is challenging. There is lack of methods to estimate CCF parameters between non-identical software modules. EPRI (2012) recommends a beta-factor of 0.001-0.1, when similar platforms are used to implement similar functions activated at different plant conditions. Functional diversity is often assumed to prevent application software CCFs between subsystems. On the other hand, operating system CCFs between functionally diverse subsystems are usually modelled.

Hardware CCFs can be analysed according to normal CCF analysis principles. However, lack of data on digital I&C components often makes it necessary to use generic parameters or engineering judgment -based methods, which can lead to quite conservative results. Large and asymmetric CCF groups are one particularly challenging area related to digital I&C hardware.

Data related to both software and hardware CCFs are sparse. Even though hardware CCF analysis is more mature than software CCF analysis, there is need for data collection and method development activities in both areas.

References

AREVA. 2013. AREVA design control document rev. 5 - Tier 2 chapter 19 - probabilistic risk assessment and severe accident evaluation. U.S. Nuclear Regulatory Commission. https://www.nrc.gov/docs/ML1326/ML13262A290.html

Authen, S., Bäckström, O., Holmberg, J.-E., Porthin, M. and Tyrväinen, T. 2016. Modelling of digital I&C, MODIG - Interim report 2015, NKS-361. Nordic nuclear safety research, Roskilde, Denmark.

Authen, S., Holmberg, J.-E., Tyrväinen, T. and Zamani, L. 2015. Guidelines for reliability analysis of digital systems in PSA context - Final report, NKS-330. Nordic nuclear safety research, Roskilde, Denmark.

Bao, H., Shorthill, T. and Zhang, H. 2020. Hazard analysis for identifying common cause failures of digital safety systems using a redundancy-guided systems-theoretic approach. Annals of Nuclear Energy, Vol. 148.

Bickel, J.H. 2008. Risk implications of digital reactor protection system operating experience. Reliability Engineering and System Safety, Vol. 93, 107-124.

Bäckström, O., Holmberg, J.-E., Jockenhövel-Barttfeld, M., Porthin, M., Taurines, A. and Tyrväinen, T. 2015. Software reliability analysis for PSA: failure mode and data analysis, NKS-341. Nordic nuclear safety research, Roskilde, Denmark.

Cai, Y., Wu, Y., Zhou, J., Liu, M. and Zhang, Q. 2020. Quantitative software reliability assessment methodology based on Bayesian belief networks and statistical testing for safety-critical software. Annals of Nuclear Energy, Vol. 145.

Chu, T.-L., Varuttamaseni, A., Baek, J.S., Yue, M., Kaser, T., Marts, G., Murray, P., Harwood, B., Johnson, N. and Li, M. 2017. Development of a statistical testing approach for quantifying safety-related digital system on demand failure probability, NUREG/CR-7234. U.S. Nuclear Regulatory Commission, Washington, DC.

Chu, T.-L., Varuttamaseni, A., Yue, M., Lee, S.J., Kang, H.G., Cho, J. and Yang, S. 2018. Developing a Bayesian belief network model for quantifying the probability of software failure of a protection system, NUREG/CR-7233. U.S. Nuclear Regulatory Commission, Washington, DC.

Chu, T.-L., Yue, M., Martinez-Guridi, G. and Lehner, J. 2010. Review of quantitative software reliability methods. Brookhaven National Laboratory.

Electric Power Research Institute. 2012. Modelling of digital instrumentation and control in nuclear power plant probabilistic risk assessment. Palo Alto, California.

Enzinna, B., Shi, L. and Yang, S. 2009. Software common-cause failure probability assessment. Proceedings of Sixth 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.

Guo, C., Zhou, S., Li, J., Chen, F., Li, D. and Huang, X. 2019. A novel software reliability growth model of safety-critical software considering fault severity classification. 4th International Conference on System Reliability and Safety, Rome, Italy, November 20-22, 2019.

International Atomic Energy Agency. 2009. Protecting against common cause failures in digital I&C systems of nuclear power plants, IAEA nuclear energy series No. NP-T-1.5. Vienna, Austria.

International Electrotechnical Commission. 2010. Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3, IEC 61508-6.

Jockenhövel-Barttfeld, M., Karg, S., Hessler, C. and Bruneliere, H. 2018. Reliability analyses of digital I&C systems within the verification and validation process. 14th International Conference on Probabilistic Safety Assessment and Management, PSAM14, Los Angeles, California, 16-21 September, 2018.

Jockenhövel-Barttfeld, M., Abusharkh, Y., Hessler, C. and Kollasko, H. 2019. Estimation of specific common cause factors for digital I&C modules in the PSA. PSAM 2019 topical: Practical use of probabilistic safety assessment, Stockholm, Sweden, 2-3 December, 2019.

Johanson, G., Hellström, P., Mankamo, T., Bento, J.-P., Knochenhauer, M. and Pörn, K. 2003. Dependency defence and dependency analysis guidance, SKI report 2004:04, Appendix 4.1 Model survey and review. Swedish Nuclear Inspectorate, Sweden.

Korea Electric Power Corporation and Korea Hydro & Nuclear Power Co., Ltd. 2018. APR1400 design control document tier 2, Chapter 19, Probabilistic risk assessment and severe accident evaluation, revision 3. U.S. NRC. https://www.nrc.gov/reactors/new-reactors/design-cert/apr1400/dcd.html

Liang, Q., Liu, M., Xiao, P., Guo, Y., Xiao, J. and Peng, C. 2020. Reliability assessment for a safety-related digital reactor protection system using event-tree/fault-tree (ET/FT) method. Science and Technology of Nuclear Installations, Vol 2020.

Littlewood, B. and Rushby, J. 2011. Reasoning about the reliability of diverse two-channel systems in which one channel is "possibly perfect". IEEE Transactions on Software Reliability, Vol. 38, No. 5, 1178-1194.

Porthin, M., Shin, S.M., Jaros, M., Sedlak, J., Picca, P., Quatrain, R., Demgne, J., Brinkman, H., Natarajan, V., Tyrväinen, T., Mueller, C. and Piljugin, E. 2021. WGRISK DIGMAP: Comparison of PSA modelling approaches for digital I&C. 12th Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies, NPIC&HMIT 2021, Virtual meeting, June 14-17, 2021.

Regulator Task Force on Safety Critical Software. 2018. Licensing safety critical software for nuclear reactors, Common position of international nuclear regulators and authorised technical support organisations. Bel V, BfE, CNSC, CSN, ISTec, KAERI, KINS, NSC, ONR, SSM, STUK.

Shin, S.M., Cho, J., Jung, W. and Lee, S.J. 2017. Test based reliability assessment method for a safety critical software in reactor protection system. 10th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies, NPIC&HMIT 2017, San Francisco, California, June 11-15, 2017.

Smidts, C.S., Shi, Y., Li, M., Kong, W. and Dai, J. 2011. A large scale validation of a methodology for assessing software reliability, NUREG/CR-7042. U.S. Nuclear Regulatory Commission, Washington, DC.

Sun, W. 2013. Determination of beta-factors for safety instrumented systems. Norwegian University of Science and Technology, Trondheim, Norway.

Tyrväinen, T., Bäckström, O., Holmberg, J.-E. and Porthin, M. 2016. SICA - A software complexity analysis method for the failure probability estimation. 13th International Conference on Probabilistic Safety Assessment and Management, PSAM13, Seoul, Korea, 2-7 October, 2016.

Varde, P.V., Choi, J.G., Lee, D.Y. and Han, J.B. 2003. Reliability analysis of protection system of advanced pressurized water reactor - APR 1400. Korea Atomic Energy Research Institute.

Yang, Y. and Sydnor, R. 2012. Reliability estimation for a digital instrument and control system. Nuclear Engineering and Technology, Vol. 44, No. 4, 405-414.

U.S. Nuclear Regulatory Commission. 2016. CCF parameter estimations 2015. Washington DC.

Appendix: Questionnaire to Finnish power companies

Software CCFs

1. Which software CCFs do you include in PRA model?
2. How do you estimate probabilities of software failures?
3. In which cases do you assume complete dependence (beta-factor of 1) between software modules?
4. Do you use beta-factor smaller than 1 for some software CCFs, or some other CCF model? For which software CCFs? Which models and parameters do you use?
5. If you have screened out software CCFs, what are the reasons for that?
6. Do you model CCFs between non-identical software modules? How and with what parameters? What is the reasoning for the modelling choices?
7. Do you model CCFs between software modules in different systems or subsystems? How and with what parameters? What is the reasoning for the modelling choices?
8. What could be improved in your software CCF modelling?
9. What aspects of software CCFs should be studied more? Do you have any suggestions for research?

Hardware CCFs

10. Which models do you use to model hardware CCFs in digital I&C systems?
11. Where do you get the parameter values for hardware CCFs in digital I&C systems? Can you provide details of the parameter estimation?
12. Are there cases where you do not model CCFs between identical hardware components in digital I&C systems, e.g. components in different systems? What is the reasoning for not modelling such CCFs?

13. Which are the hardware component types in digital I&C systems for which CCFs are modelled?

14. How large are the CCF groups for hardware components in digital I&C systems?

15. Is there difference between CCF modelling of detected and undetected failures of hardware components in digital I&C systems? What are the differences if there are any?

16. What are the main modelling challenges related to hardware CCFs in digital I&C systems? Do you have any suggestions for research?