Analysis and modelling of software in probabilistic safety assessment

HOLMBERG Jan-Erik[1], BÄCKSTRÖM Ola[2], and TYRVÄINEN Tero[3]

1. Risk Pilot AB, Rummunlyöjänkatu 11 G 45, FI-02600, Espoo, Finland (jan-erik.holmberg@riskpilot.fi)
2. Lloyd Register Consulting, Box 1288, SE-17225 Sundbyberg, Sweden (Ola.Backstrom@lr.org)
3. VTT: P.O.Box 1000, FI-02044 VTT, Finland (Tero.Tyrvainen@vtt.fi)

Abstract: Currently, no consensus approach is available for assessing safety and reliability of digital I&C at nuclear power plants. Due to the absence of a common method for modelling software failures in the probabilistic safety assessment (PSA), generic conservative common cause failure probabilities are usually used, which tend to be conservative and may ultimately prevent PSA results from providing proper risk insights. This paper presents a method for the quantification of software failures in a reactor protection system. The emphasis of the method is in the definition of the relevant software fault cases and related failure effects. Software fault cases are associated with different software modules, such as system software and application software modules. The approach for the reliability quantification is dependent on the type of module. The failure effects are divided into fatal failure and non-fatal failure of the processor. In the latter case, a specific I&C function is affected and the effect can be failure to actuate on demand or spurious actuation. To estimate the failure probability of a system software module operating experience may be used given that normal operation conditions correspond with transient conditions. For application software modules, indirect evidence needs to be used. The quantification is based on two main metrics: complexity of the application software and the degree of verification and validation of the software. The fractions between fatal vs. non-fatal failure as well as between failure to actuate and spurious actuation is based on expert judgement. The outlined quantification method offers a practical and justifiable approach to account for software failures that are usually ignored in current PSAs. Validation of the method will be a future activity.

Keyword: software reliability; probabilistic safety assessment; reactor protection system; nuclear safety

1 Introduction

Digital instrumentation and control (I&C) is becoming more and more common in nuclear power plants (NPPs). Turbine plant I&C and diverse other safety-related systems, which have minor role in probabilistic safety assessment (PSA) context, are already digital. Although quite a number of plants have received digital reactor protection systems (RPS) either as original equipment (*e.g.* China, France, Japan, United Kingdom) or in upgrade projects (*e.g.* Sweden, Switzerland, USA), most plants do not yet have digital reactor protection system. New-builds will have complete digital I&C.

Currently, no consensus approach is available in the NPP field for assessing safety and reliability of digital I&C and meeting related regulatory requirements. However, there is a tradition to try to find harmonised approaches for probabilistic safety assessment (PSA) and its applications. For areas of greater uncertainty, *e.g.*, analysis of digital I&C, the driver to find common approaches and guidelines is strong. Due to the absence of a common method for modelling software failures in the PSA, generic conservative software common cause failure (CCF) probabilities are usually used which tend to be conservative and may ultimately prevent PSA results from providing proper risk insights.

This paper presents a method for quantification of RPS software failures in a nuclear PSA context. The aim is to define a simple yet sufficient model which describes the software failure impacts and provides a quantification approach for the failures. The method has been developed in the Nordic DIGREL project,[1, 2] and builds partly on the work on taxonomy of failure modes of digital components for the purposes of PSA conducted by the international OECD/NEA Working Group RISK.[3]

Received date: January 5, 2015
(Revised date: February 12, 2015)

2 State-of-the-practice of modelling software in PSA

2.1 Background

This chapter gives an overview of the state-of-the-practice in software reliability analysis in PSA. 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 PSA 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 the software-based systems has on the dependence between the safety functions and the structure of accident sequences.

In spite of the unsolved issue of addressing software failures, there seems to be a consensus regarding some 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 [4]. A similar conclusion was made in the Workshop on Philosophical Basis for Incorporating Software Failures in a Probabilistic Risk Assessment [5]. As part of the open discussion, the panellists unanimously agreed that:

- software fails
- the occurrence of software failures can be treated probabilistically
- it is meaningful to use software failure rates and probabilities
- software failure rates and probabilities can be included in reliability models of digital systems.

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[4] and software metrics based methods[6,7]. These methods are reviewed in Ref. [8].

2.2 Software reliability estimation in PSA

In the context of PSA 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 [4,5] 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 PSAs for NPPs. Software failures are either omitted in PSA or modelled in a very simple way as common cause failure (CCF) related to the application software (AS) 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 PSA can be divided into the following categories depending on the argumentation and evidence they use [9]:

- 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).

2.2.1 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) exists.

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.[10] Accordingly, the “Common Position” document states that reliability claims “pfd < 1E-4” for a single software based system important to safety shall be treated with extreme caution.[11] The basis for such a assumption is due to the fact that demonstrating lower probabilities, *e.g.*, by statistical testing is very laborious.

2.2.3 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 approach is used extensively in PSA, *e.g.*, in human reliability analysis. But 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 [12].

In one protection system reliability analysis study, it was assumed that the contribution from software failure to total failure probability is 10% of the hardware failure probabilities.[13] 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) [14] 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 [15]: 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.

2.2.4 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 the reliability estimation.

In a Swedish PSA, the contribution of software CCF to the unavailability of a safety system was assessed based on operational experience [16]. 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 [17], 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.

2.3 Conclusions on software reliability in PSA

Generally, only common cause failures are modelled in PSA. 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 PSA made for an automation renewal project, where four different levels of software failures were 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 [18].

It is difficult to trace back where the reliability numbers used in PSA come from — even in the case of using operating experience. The references indicate a sort of engineering judgement but lacks supporting argumentation.

3 Failure modes taxonomy

3.1 WGRISK/DIGREL task group work

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 this field. One of the recommendations was to develop a taxonomy of failure modes of digital components for the purposes of probabilistic safety assessment (PSA), resulting in a follow-up task group called DIGREL.[19]

The WGRISK/DIGREL failure modes taxonomy [3] is based on a hierarchical definition of five levels of abstraction for a nuclear power plant safety automation: 1) system level, 2) division level, 3) I&C unit level, 4) I&C unit module level, 5) basic component level. This structure corresponds to a typical reactor protection system architecture. See Fig. 1.

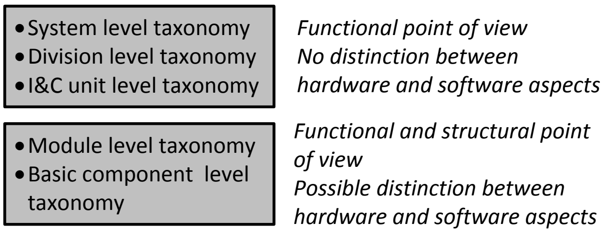
• System level taxonomy
• Division level taxonomy
• I&C unit level taxonomy
Functional point of view
No distinction between hardware and software aspects
• Module level taxonomy
• Basic component level taxonomy
Functional and structural point of view
Possible distinction between hardware and software aspects

Fig.1 Levels of abstraction and points of view in the failure modes taxonomy. [3]

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 I&C 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 or I&C unit the fault is located).
- Failure effect: 1) Fatal failure (generation of outputs ceases, outputs are set to specified, supposedly safe values), 2) Non-fatal failure (generation of outputs continues with possibly wrong output values).
- Detection situation: On-line detection, off-line detection, revealed only by demand, spurious actuation.

The combination of fault location, failure 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,
- Loss of a specific I&C function (no actuation when demanded),
- Spurious I&C 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 PSA, at least for protection systems.

3.2 Example architecture

In order to define an approach to analysis and modelling of digital I&C, an example design has been considered in DIGREL. The architecture of the safety I&C is presented in Fig. 2. The protection system is divided into two subsystems, called RPS-A and RPS-B. The two subsystems enable diversification of safety functions the whole path from sensors to actuators. The four divisions (1 to 4) are identical.

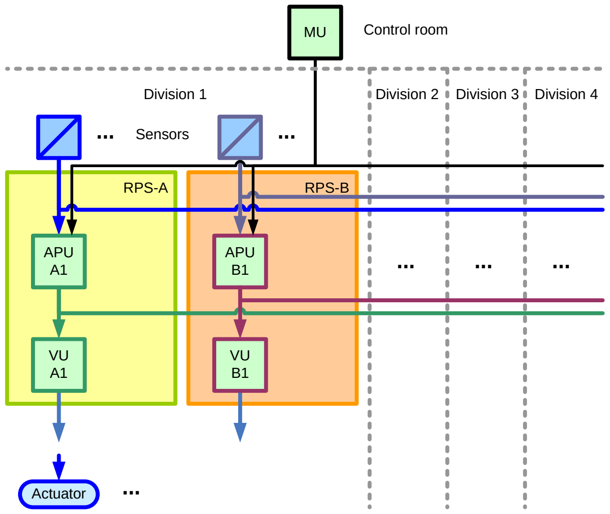

Fig.2 Architecture of the example reactor protection system. APU = Acquisition and processing unit, VU = Voting unit, MU = Processor unit for operator.[1]

The example reactor protection system is designed with fault tolerant features, which provides means to detect failures and mark faulty signals, *e.g.* self-surveillance, dynamic self-test, open circuit monitoring, cross channel comparison etc. Fault processing is implemented in the design of the hardware circuits and the software logic, and it can be defined on a case-by-case basis how the logic shall react if invalid input signals are present, and how output signals shall be set in case of faulty logic signals.

In general, the following applies for detected failures of the example I&C protection system:

- Detected failure in input signals, in intra I&C unit signal processing or in inter I&C unit signal exchange will cause corresponding signals to be replaced by a default value of 0 or 1.
- Complete, or fatal, failure of an I&C unit, *e.g.* processor failure or power supply failure, will cause all output channels of the I&C unit to 0 and controlled actuators will go to the predefined fail-safe state.

3.3 Software failure modes and effects

The approach to handle software failures is based on the postulation of software faults in different software modules and the consideration of a limited but representative number of end effects for the software module failures. The following software modules are considered:

- System Software (SyS), which is generic to the system (platform).
- Application software (AS) modules, which is specific to the application function implemented in APU or VU.
- Elementary function (EF) blocks (or library functions) used in the design of application software modules.
- Data communication software, which is the operating system of the data communication units.
- Data link configuration, which is specific to the network (*e.g.* RPS-A or RPS-B).
- Proprietary software in hardware modules (other than the processor module).

In principle a fault can be postulated in any of the software modules listed above, and consider all the theoretically possible failure effects in the I&C units. It is, however, sufficient to distinguish between fatal and non-fatal failures and to consider the relevant CCF cases. This consideration leads to the cases listed in Tables 1 and 2.

In system fault trees, software module faults can be modelled parallel to hardware module failures, which have the same failure effect.

Table 1 Screening of software modules failure cases

Case	Description
1	Fatal failure causing loss of all subsystems that have the same System Software (SyS), *e.g.*, RPS-A and RPS-B in the example architecture.
2a	Fatal failure causing loss of processing units in one subsystem, *e.g.*, RPS-A or RPS-B. The whole subsystem stops running and outputs are set to 0.
2b	Fatal failure in communication modules of one subsystem (RPS-A or RPS-B). The voting units (VU) run and take default values. Fatal failure of communication modules in both subsystems is omitted due to diversity and separation of the communication between subsystems.
3	Fatal failure causing failure of redundant set of I&C units, *i.e*, a set of acquisition and processing units (APU) or a set of voting units (VU) in one subsystem.
4	Non-fatal failure associated with an application software module. Failure effect can be a failure to actuate the function or a spurious actuation. The fault can be in the APUs or VUs.

Table 2 Software modules failure cases and failure effects

	Software fault case												
	1	2a		2b		3				4			
I&C unit	RPS-A&B	RPS-A	RPS-B	RPS-A	RPS-B	APU-A	VU-A	APU-B	VU-B	APU-A	VU-A	APU-B	VU-B
APU-A	**0**	**0**		**d**		**0**				**f**			
VU-A	**0**	**0**		**d**			**0**				**f**		
APU-B	**0**		**0**		**d**			**0**				**f**	
VU-B	**0**		**0**		**d**				**0**				**f**

0 = fatal failure of the unit, outputs goto 0
d = communication lost, outputs goto default values
f = non-fatal failure of the unit, specific I&C functions are affected (no actuation or spurious actuation)

4 Software reliability quantification

Suitable software reliability quantification method depends on the type of software modules. In the example, the modelling and quantification of software was simplified into the four cases described in Section 3.3.

System software failures cover the cases 1 and 2a. Data communication unit software faults are covered by case 2b.

Each application software module needs to be considered specifically and the failure belongs to the case 3 or 4.

Faults in elementary function blocks can be included in the AS failures. This is based on the judgement that faults in EF modules are unlikely. Faults in AS are mainly caused by wrong use of EF modules.

Proprietary software faults can be included in the hardware modules failures.

4.1 Use of operating experience for system software and data communication software failure rates

The failure cases 1, 2a and 2b (see Section 3.3) should preferably be estimated for the system in question from the operational history. The main challenge is to find historical events that have caused a complete fatal failure of the whole system.

According to the analysis of the I&C system vendor AREVA GmbH, the following order of magnitude could be estimated for the different cases (numbers presented here are not exactly those which were estimated from the vendor data, but they reflect the order of magnitude that can be estimated from the available data):[2]

- Case 1: fatal failure of all subsystems with the same system software, pfd ~ 1E-7. There is no experience from such events. It can be assumed to be a fraction of Cases 2a and 3, depending on the degree of diversity between the subsystems (RPS-A vs. RPS-B).
- Case 2a: fatal failure of one subsystem, pfd ~ 1E-6. There is no experience from such events. Failure rate is estimated using Bayesian approach with Jeffreys non-informative prior distribution yielding the posterior mean value 0.5/T for the failure rate, where T is the observation time. The value for pfd corresponds to 24 h mission time.
- Case 2b: fatal failure in data communication unit software of one subsystem, pfd ~ 1E-5 (some occurred events may be classified in this category, though no complete CCF has been observed).

4.2 Estimation of application software module failure probabilities

4.2.1 General assumptions

There are several AS modules on each I&C unit (APU and VU). A fault in one application software, which causes a fatal failure of the processor affects also the other application software modules running on the same processor (case 3). Hence, a fatal failure can affect the other processes – but only in the configuration that the information output stops.

A non-fatal failure in one application software module (case 4) can produce an incorrect output (no actuation when demanded or spurious actuation). If there is a strict separation between the system and application software, a non-fatal failure does not affect the other processes running in the same processor.

4.2.2 Indirect evidence for AS failure probability

In the proposed quantification method, indirect evidence is applied for the failure probability estimates of application software modules using the metrics “Complexity” and “Verification & Validation” (V&V). In the case of AS modules in RPS, they all belong to same V&V category. The idea of the method is however that it could be applied to systems which have lower safety class and V&V requirements than RPS. The V&V metric would then make variation between software modules in different systems.

Regarding complexity, three categories are assumed to be sufficient (high-medium-low). It is assumed that failure probabilities differ by factor 10 between the categories, *i.e.*,

$$\text{pfd(high)} = 10\cdot\text{pfd(medium)} \qquad (1)$$

$$\text{pfd(medium)} = 10\cdot\text{pfd(low)}. \qquad (2)$$

Reference values for pfd have been searched from literature [2]. The range is large, and preferably operating experience may be used to determine justifiable reliability numbers. For RPS it can be hard to find enough real demand data. However, data from other I&C systems can be used, too, as long as the platform is same. This will be a future task.

Meaning of high/medium/low has been studied by comparing typical logic diagrams for AS modules in RPS. It is apparent that most of them are in the “low” and “medium” category, and “high” is a rare exception. For the categorisation purposes, some complexity metrics have been compared.[2] The metrics take into account number of elementary function blocks (or library functions), types of function blocks (*e.g.* with or without memory), complexity of interconnections between function blocks and number of inputs and outputs in the analysed diagram, which are considered as indicators for complexity. As an assessment method, this seems to be practical, but the complexity categorisation principle needs to be validated in future.

It should be noted that the “application software module” can be defined in various manner, which should be taken into account when assessing complexity and analysing operating experience. The largest meaningful definition for an AS module is an I&C function, which usually consists of several sub-modules. The I&C function level of abstraction may miss the dependences between the functions (due to common sub-modules). The sub-module level of abstraction may lead to impractically high number of basic events in PSA. Experiment with a full-scale PSA for an NPP with digital safety I&C is needed to find an appropriate definition for an AS module.

4.2.3 Split fractions

The probability pfd above covers both fatal and non-fatal failure modes of AS modules as well as “failure to actuate” and “spurious actuation” failure modes, see Fig. 3. At the moment only engineering judgements are available to judge the split fractions of the AS module failure modes. In the future, the aim is to search for as representative data as possible to get supporting evidence for the split fractions.

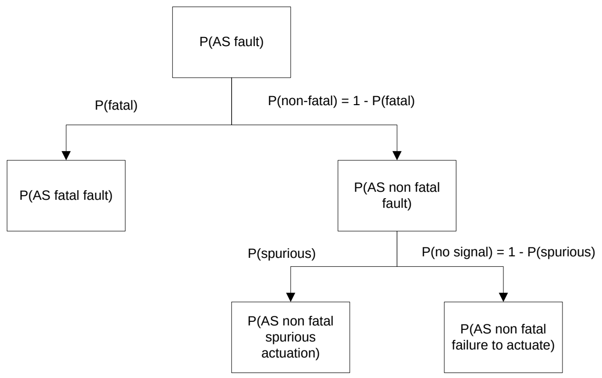
P(AS fault)
P(fatal)
P(non-fatal) = 1 - P(fatal)
P(AS fatal fault)
P(AS non fatal fault)
P(spurious)
P(no signal) = 1 - P(spurious)
P(AS non fatal spurious actuation)
P(AS non fatal failure to actuate)

Fig. 3 Split fractions of the AS module failure probabilities [2]. (P(AS fault) = pfd).

4.2.4 CCF between related AS modules

Generally, the AS modules performing identical function in redundant divisions are assumed to be identical software modules. The conditional probability of CCF is assumed to be 1, given a fault in an AS module.

In addition to CCF between identical AS modules, it is worth considering CCF between related AS modules. At least two kinds of relationships can form a potential to CCF:

- use of same elementary functions
- common functional requirements specifications.

With regard to the fault coupling by elementary functions, it is in principle possible to assume fault in an elementary function module, which would be then a common fault for more than one AS module. However, the position of this method is that faults in elementary functions are practically eliminated due to their being part of the rigorous verification and validation of the system software. It is instead more likely that an AS fault is caused by a wrong usage of complex elementary function. Thus, the main attention should be paid on the assessment of use of complex elementary functions, and the fault coupling can be associated with the coupling via common functional requirements specifications.

With regard to the fault coupling by functional requirements specifications, the interesting case is possible CCF between two AS modules implementing same I&C function in diverse subsystems. There are numerous such examples in the current way designing safety I&C, since it is an overall requirement that the actuation of safety function should be accomplished by two different process parameters (*e.g.*, temperature and pressure of primary circuit). The structure of the actuation logic is often same for the two “diverse” functions and they are defined in the same functional requirements specification. Conservatively, the conditional probability of CCF could be assumed to be 1. Optimistically, no dependency is assumed. Realistic assessment is somewhere between. Future work is needed to find a justifiable approach to assess the degree of diversity.

5 Evaluation with the example PSA

In DIGREL, an existing simplified PSA model has been complemented with fault tree models for a four-redundant distributed protection system in order to study and demonstrate the effect of design features and modelling approaches. The model has been used to test the effect of different levels of modelling detail, CCF modelling, fail-safe principle and voting logic. The example PSA-model represents a fictive boiling water reactor (BWR), which has four-redundant safety systems [1].

Generally, same failure modes (types 1, 2a, 2b, 3, 4a and 4b) are considered in the model and the same failure probabilities are used as suggested in Ref. [1]. The fault trees for I&C have been structured in hierarchical manner starting from the actuator down to measurements. The model of the digital I&C currently consists of 680 fault trees pages, 460 basic events and 100 hardware CCF groups. Software faults are modelled with a total of 44 CCF basic events.

The results with the example PSA model show that software faults have a significant impact on the overall result. Software faults in total have a fractional contribution of about 9%.[1] Fractions of different types of software faults (cases 1, 2, 3 and 4 of Tables 1 and 2) are dependent on the assumed probabilities and the design. It can, however, be concluded that software faults in general have a non-negligible effect on the results and should be considered in a digital I&C PSA. Quantification of software faults and the assessment of the degree of diversity between subsystems can therefore be significant from the overall PSA results point of view. The evaluations with the example model also showed that the failure mode “spurious actuation” has some impact and should not be ignored in PSA.

6 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 PSA, computer-based systems are mostly analyzed simply and conventionally. The conventional failure modes and effects analysis and fault tree modelling are utilized. The survey of literature and PSA shows that software failures are either omitted in PSA or modelled in a very simple way as CCF related to the application software of operating system. It is difficult to find 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, a failure modes taxonomy was developed jointly by PSA and I&C experts. 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 PSA studies.

In the Nordic DIGREL project, a method for the quantification of software failures has been developed. The emphasis of the method is on the quantification of the failure probability of an application software module, which can lead to the functional failure modes: failure to actuate on demand a specific instrumentation and control (I&C) function or spurious actuation of a specific I&C function.

The quantification of the application software module is based on two main metrics, complexity of the application software and the degree of verification and validation of the software. Common cause failures and different failure modes are covered by the method. Operational data may be used for software reliability quantification but collecting and using it is challenging and requires more research. The outlined quantification method offers a practical and justifiable approach to account for software failures that are usually ignored in current PSAs.

The results with the example PSA show that software faults have a significant impact on the overall result. Quantification of software faults, consideration of different failure modes and the assessment of the degree of diversity between subsystems can therefore be significant from the overall PSA results point of view.

Nomenclature

APU	Acquisition and processing I&C unit
AS	Application software module
BBN	Bayesian belief network
BWR	Boiling water reactor
CCF	Common cause failure
DIGREL	Guidelines for reliability analysis of digital systems in PSA context
EF	Elementary function
I&C	instrumentation and control
MU	Main control room I&C unit for operators
NKS	Nordic nuclear safety research
NPP	nuclear power plant
pfd	probability of failure per demand
PSA	Probabilistic safety assessment
RPS	Reactor protection system
SIL	Safety integrity level
SyS	System software module
V&V	Verification and validation
VU	Voting I&C unit
WGRisk	Working Group on Risk Assessment (OECD/NEA)

Acknowledgement

The work has been financed by NKS (Nordic nuclear safety research), SAFIR2014 (The Finnish Research Program 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. AREVA GmbH has provided insights and data for the software reliability estimation. The taxonomy work is based on contributions from the WGRISK/DIGREL task group members. 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 paper possible.

References

[1] AUTHÉN, S., HOLMBERG, J.-E., LANNER, L., and TYRVÄINEN, T.: Guidelines for reliability analysis of digital systems in PSA context - Phase 4 Status Report, NKS-302, Roskilde: Nordic nuclear safety research (NKS), 2014.

[2] BÄCKSTRÖM, O., HOLMBERG, J.-E., JOCKENHÖVEL-BARTTFELD, M., PORTHIN, and M., TAURINES, A.: Software reliability analysis for PSA, NKS-304, Roskilde: Nordic nuclear safety research (NKS), 2014.

[3] Failure modes taxonomy for reliability assessment of digital I&C systems for PRA, report prepared by a task group of OECD/NEA Working Group RISK, Paris: OECD/NEA/CSNI, 2014.

[4] 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.

[5] CHU, T.-L., MARTINEZ-GURIDI, G., YUE, M., SAMANTHA, P., VINOD, G., and LEHNER, J.: Workshop on Philosophical Basis for Incorporating Software Failures in a Probabilistic Risk Assessment, BNL-90571-2009-IR, New York: Brookhaven National Laboratory, 2009.

[6] 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.

[7] 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.

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

[9] HOLMBERG, J.-E.: Software reliability analysis in probabilistic risk analysis, Nuclear Safety and Simulation, 2012, 3(4): 281–291.

[10] Nuclear power plants – Instrumentation and control systems important to safety – Classification of instrumentation and control functions, IEC 61226. Second edition, Geneva: International Electrotechnical Commission, 2005.

[11] 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.

[12] 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.

[13] 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, Daejeon: Korea Atomic Energy Research Institute, 2003.

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

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

[16] 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.

[17] 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.

[18] JÄNKÄLÄ, K.: “Reliability of New Plant Automation of Loviisa NPP,” Proc. DIGREL seminar Development of best practice guidelines on failure modes taxonomy for reliability assessment of digital I&C systems for PSA, October 25, 2011, VTT-M-07989-11, Espoo: VTT, 2011.

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