Title	Use of IEC 61508 in nuclear applications regarding software reliability
Author(s)	Bäckström, Ola; Holmberg, Jan-Erik
Citation	11th International Probabilistic Safety Assessment and Management Conference & The Annual European Safety and Reliability Conference, pp. 10-TH2-4
Date	2012
Rights	Reprinted from 11th International Probabilistic Safety Assessment and Management Conference & The Annual European Safety and Reliability Conference. This article may be downloaded for personal use only

Use of IEC 61508 in Nuclear Applications Regarding Software Reliability

Ola Bäckström[a], Jan Erik Holmberg[b]
[a]Scandpower –Lloyds Register, Stockholm, Sweden
[b]VTT, Espoo, Finland

Abstract: IEC 61508 [1] is a standard for Functional safety of electrical/electronic/programmable electronic safety related systems. This standard defines concepts for analyses of these types of systems. On a general level the standard is defining the concept of Safety Integrity Levels, SILs. SIL sets the requirements on the process how the equipment is manufactured, tested etc. The same is relevant for the software being developed. To meet the requirements of a SIL – the software shall meet some predefined criteria.
The software reliability estimates in the current probabilistic safety assessments (PSA) for nuclear power plants (NPP) are often 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. The backgrounds for this figure is however not clear, as what it really means. Does it mean failure of an individual processor, CCF between different processers or failure of the whole system?
This paper discusses what type of methods presented in the IEC 61508 could be applicable in the PSA domain. Especially, is the use of SIL a way to justify the failure probability of the software used in NPP applications? In that case: What is the appropriate way of defining a function with SIL requirement?
In the paper a way forward for a future implementation of SIL methodology for software analysis as part of PSA within the nuclear domain is also discussed.

Keywords: IEC61508, Software reliability, PSA, nuclear power

1. INTRODUCTION

In Probabilistic Safety Assessment (PSA) studies for nuclear power plants (NPP), all relevant failures shall be considered to give an estimate of the frequency of core damage or releases to the environment. The purpose of the PSA is to give guidance regarding if the plant is sufficiently safe or not – and if there are parts of the plant that are more significant to safety than others.

Over the last 20 years the electro-mechanical relay technology has successively been phased out from the NPPs. The technology that has been used to replace them is programmable control systems. The modelling of programmable control systems in PSA is considered as a difficult task, especially with regard to the software aspect of the systems. The reason that something is difficult to treat probabilistically is however not a good argument to completely omit the software part in PSA. Software faults may cause system failures so they should be considered in some way but the question of this report is how.

It is obvious that software differs from hardware in many fundamental ways, which makes reliability models developed for hardware unsuitable for determining reliability of software. However, the basic question: "What is the probability that a safety system or a function fails when demanded" is fully feasible and well-formed question for all components or systems independently of the technology on which the systems are based [2]. A similar conclusion was made in the workshop on Workshop on Philosophical Basis for Incorporating Software Failures in a Probabilistic Risk Assessment [3].

Over a long period of time discussions have been on-going regarding how to estimate the probability that the software does not behave as originally intended – *software faults* (meaning a system failure caused by a software fault). How to derive such a probability is still an open debate. This paper outlines a method based on IEC 61508 to determine a graded approach for assigning a justifiable probability for software faults.

2. HARDWARE AND SOFTWARE FAILURES

In the reliability analysis and modelling of digital I&C systems, it is a common practice to distinguish between hardware and software failures. Hardware consists of the physical devices (e.g. microprocessors and communication links) while software is the computer programs and data stored in the devices. To what extent the reliability of a programmable system can be decomposed into a hardware and software parts and what are the possible interactions between them is a controversial issue, which however is left out of the scope of this paper. This paper will discuss software, and in the way software reliability is treated in PSA context and in the application of IEC standards.

The hardware part of a system can be usually decomposed in the smaller elements following the physical boundaries of the elements. For software, the manner of decomposition is not self-evident, but it may be anyway possible to identify parts of the software, i.e. the software functions. In the case of safety critical programmable systems in nuclear power plants (cat. A systems), the following kind of software components can be identified:

- Processing units
  - Operating system
  - Application specific software
  - Elementary functions
- Communication units
  - Communication firmware
  - Network specific communication patterns.

In principle, for each "software component" existence of faults may be assumed and the impact of faults to system may need to be considered.

3. REVIEW OF IEC 61508 FROM PSA PERSPECTIVE

3.1. General Description of IEC 61508

IEC 61508 is a standard for Functional safety of electrical, electronic, programmable electronic safety related systems. This standard is defining concepts for analyses of these types of systems. On a general level the standard is defining the concept of Safety Integrity Levels, SILs. SIL sets the requirements on the process how the equipment is manufactured, tested etc. The same is relevant for the software being developed. To meet a SIL level one has to, e.g.,

- Use hardware that meets the reliability requirements of the specific SIL level.
- Architectural design for safe failure fractions.
- Develop software according to quality graded requirements (high SIL means high requirements on the software process).

Table 1 shows the SIL levels and their corresponding assumed failure probability and failure rate.

Table 1. Safety Integrity Levels and their corresponding reliability targets. [1]

Safety integrity level (SIL)	**Probability of a dangerous failure on demand of the safety function**
4	≥ 10-5 to < 10-4
3	≥ 10-4 to < 10-3
2	≥ 10-3 to < 10-2
1	≥ 10-2 to < 10-1
Safety integrity level (SIL)	**Frequency of a dangerous failure of the safety function [h-1]**
4	≥ 10-9 to < 10-8
3	≥ 10-8 to < 10-7
2	≥ 10-7 to < 10-6
1	≥ 10-6 to < 10-5

3.2. Quantification of Software Reliability in the IEC 61508 Context

It is considered inappropriate (or at least very hard) to prove that a software system has a lower probability of failure than 1E-4 per demand (see for example [4] where it is stated that: *A review of nuclear sector standards ... shows that claims of lower than 10^{-4} probability of failure on demand for a computer based system are required to be treated with caution.* And further it is stated that: *Reliability claims for a single software based system important to safety of lower than 10^{-4} probability of failure (on demand or dangerous failure per year) shall be treated with extreme caution.*

It is, however, not fully clear how this shall be interpreted in a reactor protection system aspect, since there are multiple processors within one sub division and the software within each processor may differ.

Looking at IEC 61508-3, the standard avoids quantitative safety assessments on software. According to the IEC61508, quantification on software with the help of IEC 61508-3 cannot be done. It is, however, also stated that the standard does allow a claim to be made that the target failure measure associated with the function can be considered achieved when the requirements in the standard have been met.

In IEC 61508-4 it is stated that: *SIL characterizes the overall safety function, but not any of the distinct subsystems or elements that support that safety function. In common with any element, software therefore has no SIL in its own right. However, it is convenient to talk about "SIL N software" meaning "software in which confidence is justified (expressed on a scale of 1 to 4) that the (software) element safety function will not fail due to relevant systematic failure mechanisms when the (software) element is applied in accordance with the instructions specified in the compliant item safety manual for the element".*

4. SOFTWARE RELIABILITY DATA IN NUCLEAR PSA-STUDIES

Reviewing nuclear power plant PSA literature, it can be concluded that the software reliability estimates are engineering judgments. Sophisticated software reliability estimation methods described in academic literature are not applied in real industrial PSAs. The engineering judgement approaches can be divided into the following categories depending on the argumentation and evidence they use:

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

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 [5].

The only reliability model which is applied is constant unavailability (or per demand failure probability) and this is used to represent the failure probability (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 to mean the application software CCF or its meaning has not been specified. Software CCF is generally modelled between processors performing redundant functions, having same application software and being in same platform.

With regard to the reliability numbers used, it is difficult to trace back where they come from. The references indicate sort of engineering judgement but supporting argumentation lacks.

5. CCF TYPES IN SOFTWARE

In a nuclear application, the assignment of CCF for software is 100% crucial. Looking at a single processor there is no (or very little) doubt that the hardware failure mechanism will be the dominating factor. But seen

from the system the software faults may very well be the dominating factor – since if software produces an unexpected behaviour, it is likely that the same type of software will produce the same result. CCFs for software used in redundant channels can happen and will be important contributors to digital-system unreliability

When having redundancy, CCFs should be applied were similar/same software components are used. But what is really to be considered as identical software? In a nuclear application a train (redundancy) typically consists of several processors, and each processor handle some functions. The arrangement of functions controlled by the different processors within a train is typically performed so that the safety systems will still operate given that a CCF affect the processors with 100% identical software.

How to consider CCFs between processors where the software is not 100% identical, but they are built in the same software system, built with the same "bricks", using the same compiler, running on the same operating system etc.? Can these potential CCF causes be omitted? This is really the critical part and the part hard to estimate, regarding software CCF.

Roughly, some cases can be outlined:

- Same application, same platform
- Same application, different platforms
- Different applications, same platform
- Different application, different platforms

6. DISCUSSION AND JUSTIFICATION FOR THE PROPOSED METHOD

The SIL method is a graded quality approach to determine a reasonable probability of failure for a computer based system.

The failure probability for critical software faults ought to have lower failure probability than hardware. Thereby the software failure should at least have the same or lower failure probability than hardware. Therefore, for an individual processor the probability for software can typically be neglected (as normally done in different industries). In such cases, it is a justifiable approach to neglect the failure probability of software. This is NOT the case in nuclear applications regarding reactor protection system. According to IE61508, the standard does give a possibility to claim that the failure probability also for systematic faults (software) is justified – and that is the basis for the approach proposed in this document

The approach to use IEC61508 for software fault estimate can be controversial: There are documents that states that *you cannot use SIL to estimate software reliability* (see e.g. [6]).

According to [IE C61508] the purpose of the standard is: …*sets requirements for the avoidance and control of systematic faults, which are based on experience and judgement from practical experience gained in industry. Even though the probability of occurrence of systematic failures cannot in general be quantified the standard does, however, allow a claim to be made, for a specified safety function, that the target failure measure associated with the safety function can be considered to be achieved if all the requirements in the standard have been met.*

When a certain SIL level is met it can then be stated, see above, that the product will meet certain justified probabilistic criteria. This means, that the failure probability that is justified is also covers software. Quote from IEC 61508 …*it is convenient to talk about "SIL N software" meaning "software in which confidence is justified…"*

From a general perspective this is of course reasonable: The more stringent requirements put on the software, should lead to lower failure probability for critical software faults.

Altogether, this means that the probability for critical software faults should be lower than the failure probability of the product – if not – the product would not meet the justified failure probability. Therefore, it

ought to be reasonable to assume that the probability for critical software faults is at least – or rather better than – the same as the corresponding SIL.

7. CONCLUSION, PROPOSED METHOD

A necessary pre-requisite for the method is that the SIL level for the software has been proven. For a nuclear application this normally means that the software has been shown to be a category A application — and based on a deterministic approach therefore shown to meet the acceptance criteria. The Safety Integrity Level is then used to justify the probability or frequency of a critical software fault.

The failure modes of the software system need to be addressed carefully. The failure modes proposed are "All functions within a platform" or" Single function(s) within a platform".

It is imaginable that a platform can get "passive" and not send any signals at all – causing a global CCF at a real demand. It is however hard to imagine that the platform sends different spurious signals caused by a software fault from all units. Therefore, the spurious function needs to be addressed specifically.

In SIL evaluations the software reliability is considered better than hardware (if the requirements on the software meets the SIL requirements) – indicating that it is reasonable to assume that the failure probability for software is in the lower bound of the SIL (or better). This is however not justified at this stage.

It is also crucial to define the degree of diversification within the system. In case of the same application (function) in same platform, the probability of CCF can be assumed to be 1. In case the same application is implemented in different platforms, the probability of CCF may be assumed to be 1, too, depending however on the way the requirements specification and its implementation is done. An interesting case is the probability of CCF between different applications in same platform, which needs to be assessed case by case. Finally we have the case between different applications in different platforms. The probability of CCF should be low but it is not sure if CCF can be fully excluded.

Note that the CCF also needs to take the different failure modes into account (see discussion about failure per demand and spurious operation above).

8. COMMENT, WAY FORWARD

The method proposed in this paper provides a framework to use IEC 61508 and the SIL concept for the justification of software probability. It shall be noticed that the authors of this paper do not exclude the use of better (and practically usable) methods, should such be available. So far there has not been anything else available than the use of different types of engineering judgements without a clear basis in the nuclear PSA context.

The method discussed in this paper still leaves several open issues that need to be addressed so that a common practice can be applied. The issues identified are mainly:

- Given a SIL: Which probability limit is justifiable, the upper or the lower limit of each SIL?
- How to consider different failure modes appropriately? Can the probability be split in different failure modes? If yes, what is a justifiable method for that?
- Failure modes for initiating event need to be addressed further.
- The diversification analysis of a system, and estimation of CCF probabilities – how shall that be achieved?

Acknowledgements

The project, on which this paper is based, has been funded by SAFIR2014 (The Finnish Research Programme on Nuclear Power Plant Safety 2011–2014) and NPSAG (Nordic PSA Group) through the project 22-002.

References

[1] International Electrotechnical Commission, Function Safety of Electrical/Electronic/Programmable Safety-Related Systems, Parts 1-7, IEC 61508, various dates.

[2] Dahll, G., Liwång, B., and Pulkkinen, U., "Software-Based System Reliability," Technical Note, NEA/SEN/SIN/WGRISK(2007)1, Working Group on Risk Assessment (WGRISK) of the Nuclear Energy Agency, January 26, 2007.

[3] Chu, T.-L., Martinez-Guridi, G., Yue, M., Workshop on Philosophical Basis for Incorporating Software Failures in a Probabilistic Risk Assessment. Brookhaven National Laboratory, BNL-90571-2009-IR, 2009.

[4] 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, January 2010.

[5] Authén, S., Wallgren, E., Eriksson, S., Development of the Ringhals 1 PSA with Regard to the Implementation of a Digital Reactor Protection System, 10th International Probabilistic Safety Assessment & Management Conference, PSAM 10, Seattle, Washington, June 7-11, 2010, paper 213.

[6] Smith, D.J. Simpson, K.G.L. Safety Critical Systems Handbook Safety Critical Systems Handbook. A straightforward Guide to Functional Safety: IEC 61508 and Related Standards Including: Process IEC 61511, Machinery IEC 62061 and ISO 13849, 3rd edition, 2010.